2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Intelligent Visual Servoing for Nonholonomic Mobile Robots

Carlos López-Franco[1], Michel López-Franco[2], Edgar N. Sanchez[2] and Alma Y. Alanis[1]

***Abstract*— In this work the authors present a visual servoing approach based on particle swarm optimization (PSO-PBVS). The PSO-PBVS algorithm overcomes the traditional PBVS approach by keeping the target features inside the image plane during the servoing. The PSO-PBVS approach allows to define the 3D trajectory in the task space, in the same way as the PBVS approach. In addition, an intelligent control is used to estimate the currents for each motor, and ensure that the motors provide the desired velocities.**

I. Introduction

Visual servoing refers to the use of computer vision information to control the motion of a robot. In this work, the vision information is obtained from a vision sensor mounted on the robot. There are two main approaches in visual servoing, the image based visual servoing (IBVS) which computes the robot velocity directly from the image features, and the position based visual servoing (PBVS) which uses 3D features, in this paper we use a PBVS approach [1]. The advantage of the PBVS with respect to the IBVS is that a 3D trajectory can be defined in the task space. However, one of the problems of PBVS is that the image features can be lost, and if the minimum required features are lost then the visual servoing task will fail.

To overcome the problems of PBVS we propose the use of particle swarm optimization (PSO) combined with the PBVS, we call this approach PSO-PBVS. The objective of the PSO-PBVS is to minimize the distance of the current camera pose to a desired camera pose by estimating the angular velocity ω that drives the robot to the desired pose.

The main contribution of the proposed approach is a visual servoing algorithm where the control reference can be defined in the Cartesian space, similarly to PBVS, but without its disadvantages. The proposed approach does not uses a Jacobian matrix (which requires inversion), and also thanks to the PSO algorithm the proposed approach will not loose the target features.

Once that the velocities v, ω of the robot are known, we require to convert them to corresponding motor current. This task will be solved by an intelligent controller, which is implemented using a recurrent neural network. The intelligent controller guarantees that the desired velocities (v, ω) are provided by the motors of the robot.

This work is organized as follows: section II presents an introduction to the visual servoing problem. Section III gives a brief introduction to the particle swarm optimization. Section IV introduces the proposed PBVS approach based on PSO. In section V the neural control of the mobile robot is presented. The simulations and experimental results are presented in sections VI and VII respectively. Finally, the conclusions are given in section VIII.

[1]CUCEI, Universidad de Guadalajara, e-mail: carlos.lopez@cucei.udg.mx, almayalanis@gmail.com
[2]CINVESTAV, Unidad Guadalajara, Jalisco 45015, Mxico. sanchez@gdl.cinvestav.mx

II. Visual based control

The use of visual feedback to control a robot is commonly termed *visual servoing* or *visual control* [1], [2]. In this work the visual data is acquired from a stereo vision system that is mounted directly on the mobile robot, in which case motion of the robot induces camera motion.

The visual control objective is to minimize an error $e(t)$ defined as [3]

$$e(t) = s(t) - s^* \tag{1}$$

where $s(t)$ denote the features extracted from the current pose, and s^* denote the features extracted from the desired pose.

In this paper we consider a nonholonomic mobile robot moving on a plane. The kinematic model of the unicycle robot is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{2}$$

where the inputs v and ω represent the driving velocity and the steering velocity respectively.

The objective of visual servoing is estimate the inputs v and ω from the current pose to the desired pose using a vision sensor.

III. Particle Swarm Optimization

Particle swarm optimization (PSO) is a bio-inspired optimization technique introduced by Kennedy and Eberhart [4]. PSO is inspired by the collective behaviors of animals (such as birds) and it is widely applied to continuous and discrete optimization problems.

Consider a general unconstrained optimization problem, with the following objective function

$$f(x_1, x_2, \ldots, x_N) \tag{3}$$

where $f : \mathbb{R}^N \rightarrow [R]$, and N is the dimensionality of the search space.

The PSO algorithm starts with a population of candidate solutions encoded as particles in the search space. Particles move on the search space according to rules inspired by a flock of birds. The particle i is represented as a vector $\vec{x}_i = (x_1, x_2, \ldots, x_N)$. The velocity of a particle is represented

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

as $\vec{v}_i = (v_1, v_2, \ldots, v_N)$. Then PSO searches for the optimal solution by updating the position and velocity of each particle according to the following equations

$$\begin{aligned} \vec{v}_i(t+1) &= \eta\vec{v}_i(t) + \phi_1\rho_1(\vec{p}_i(t) - \vec{x}_i(t)) \\ &\quad +\phi_2\rho_2(\vec{p}_g(t) - \vec{x}_i(t)) \quad (4) \\ \vec{x}_i(t+1) &= \vec{x}_i(t) + \vec{v}_i(t+1) \quad (5) \end{aligned}$$

The parameter η represents the inertia weight [5], and it determines how much percentage of the velocity should be retained from the previous step to the next step. The parameter η can be found using an adaptive algorithm [6]. The parameters ϕ_1 and ϕ_2 are two constant values, the parameters ρ_1 and ρ_2 are two random values, uniformly distributed in [0,1].

For each iteration $\vec{p}_i$ is the best position found by particle i, which is computed using the objective function. The $\vec{p}_g$ is the best position found by the global population.

IV. Position Based Visual Servoing using Particle Swarm Optimization

There are two classical approaches of visual servoing, IBVS and PBVS. Where in the former the control inputs are expressed in the image space, and in the later the control inputs are expressed in the Cartesian space. The advantage of IBVS is that the target remains on the field of view, the disadvantage is that a Cartesian trajectory can not be defined, and therefore we are unable to define a trajectory in the task space only on the image space. In contrast PBVS, allows to define the Cartesian trajectory, and therefore the control is achieved in the task space, however since there is no control in the image, it is impossible to ensure that the target will always remain in the camera field of view during the servoing [7].

The motivation of using PSO is to overcome the PBVS problems, but with its advantages over IBVS.

A. PBVS model formulation

In this section we describe the PBVS task, which will be solved using the PSO algorithm.

The pose is represented using a transformation matrix given by

$${}^a\mathbf{T}_b = \begin{bmatrix} {}^a\mathbf{R}_b & {}^a\mathbf{t}_b \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (6)$$

The homogeneous matrices can be composed as

$${}^a\mathbf{T}_c = {}^a\mathbf{T}_b\, {}^b\mathbf{T}_c \quad (7)$$

A 3D point $\mathbf{x} = [x, y, z]^\top$ is defined as $\mathbf{X} = [x, y, z, 1]^\top$ in homogeneous coordinates. A point $\mathbf{X}$ relative to frame $\mathcal{F}_b$, can be defined with respect to another frame $\mathcal{F}_a$ with

$${}^a\mathbf{X} = {}^a\mathbf{T}_b\, {}^b\mathbf{X} \quad (8)$$

In our approach we define the following frames the robot frame $\mathcal{F}_r$, the current camera frame $\mathcal{F}_c$, the desired camera frame $\mathcal{F}_c^*$ and the object frame $\mathcal{F}_o$. These frames will be related with the following homogeneous matrices ${}^w\mathbf{T}_r$, ${}^w\mathbf{T}_c$, ${}^w\mathbf{T}_c^*$ and ${}^w\mathbf{T}_o^*$, which relate the world frame with the camera frame, desired camera frame, object frame respectively. We also define the transformation of the camera frame with respect to the robot frame ${}^c\mathbf{T}_r$, which is constant since the camera is fixed to the robot.

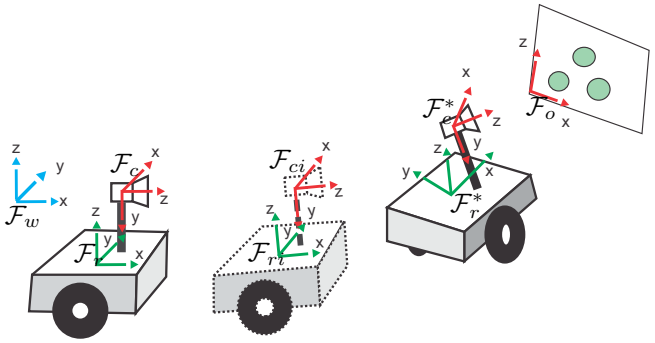

Fig. 1. Visual servoing task

At the beginning of the task we move the robot to the desired pose, and then we take the desired image, in this pose the features are

$${}^{c*}\mathbf{X}^* = {}^w\mathbf{T}_{c*}^{-1}\, {}^w\mathbf{X} \quad (9)$$

Then the robot is moved to an initial pose, the 3D features are defined with respect to the current camera as

$${}^c\mathbf{X} = {}^w\mathbf{T}_c^{-1}\, {}^w\mathbf{X} \quad (10)$$

Using the features ${}^{c*}\mathbf{X}$ and ${}^c\mathbf{X}$ we can compute the transformation between them, that is

$${}^c\mathbf{X}^* = {}^c\mathbf{T}_{c*}\, {}^{c*}\mathbf{X}^* \quad (11)$$

B. PSO-PBVS

The unicycle robot requires two velocities v and ω, which are the linear and angular velocities. The velocity v is set to a constant, which can decrease proportionally to the desired target. The velocity ω is estimated using the proposed algorithm. In the PSO algorithm each particle contains a feasible ω value.

For each particle i a feasible pose is estimated using the corresponding ω_i, with

$${}^c\mathbf{T}_{ci}(v,\omega) = {}^c\mathbf{T}_r\, {}^r\mathbf{T}_{ri}(v,\omega)\, {}^c\mathbf{T}_r^{-1} \quad (12)$$

where ${}^c\mathbf{T}_r$ relates the camera coordinate frame with the robot frame, and since the camera is fixed to the robot then this transformation is constant. The transformation ${}^r\mathbf{T}_{ri}(v,\omega_i)$ computes the new robot pose based on the velocity v and the particle velocity ω_i with

$${}^r\mathbf{T}_{ri} = \begin{bmatrix} \mathbf{R}_z(\omega_i) & \mathbf{R}_z^{-1}(\omega_i)\mathbf{p}(v,\omega_i) \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (13)$$

where $R_z(\omega_i)$ is a rotation about the z-axis with an angle of ω_i, and the vector $\mathbf{p}(v,\omega_i) = (v\cos\omega_i, v\sin\omega_i, 0)^\top$.

For each feasible pose ${}^c\mathbf{T}_{ci}$ the 3D features are projected into the image plane to test if the image features u_i are inside the boundaries of the image. The image features can be computed with

$$u_i = \left[\begin{array}{cc} K & \mathbf{0}_{3\times 1} \end{array}\right] {}^c\mathbf{T}_{ci}^{-1}\, {}^cX \tag{14}$$

If the features u_i are outside the valid boundaries then the proposed pose is discarded. On the contrary, if the features u_i are valid image features then the particle can be considered as a valid pose. The fitness function is defined as

$$f(\omega_i) = \|{}^c\mathbf{T}_{ci}(v,\omega_i)\mathbf{X}_0 - {}^c\,\mathbf{T}_{c*}\mathbf{X}_0\| \tag{15}$$

where $\mathbf{X}_0 = (0,0,0,1)^\top$. The previous equation minimize the Euclidean distance between the proposed robot pose and the desired pose, the PSO algorithm will choose the best angular velocity ω_i that minimizes it.

We can note that PSO-PBVS does not require the inversion of any matrix (e.g. Jacobian or iteraction matrix), and therefore it does not have singularities as conventional PBVS.

V. Neural Control of the Mobile Robot

The mobile robot has two actuated wheels, and its dynamics can be expressed in the following state-space model [17], [18], [19]

$$\begin{aligned} \dot\chi_1 &= J(\chi_1)\chi_2 \\ \dot\chi_2 &= M^{-1}(-C(\dot\chi_1)\chi_2 - D\chi_2 - \tau_d + NK_T\chi_3) \\ \dot\chi_3 &= L_a^{-1}(u - R_a\chi_3 - NK_E\chi_2) \end{aligned} \tag{16}$$

where each subsystem is defined as

$$\begin{aligned} \chi_1 &= [\chi_{11},\chi_{12},\chi_{13}]^T \\ \chi_2 &= [\chi_{21},\chi_{22}]^T \\ \chi_3 &= [\chi_{31},\chi_{32}]^T \end{aligned}$$

with

$$\begin{aligned} J(\chi_1) &= 0.5r\begin{bmatrix} \cos(\chi_{13}) & \cos(\chi_{13}) \\ \sin(\chi_{13}) & \sin(\chi_{13}) \\ R^{-1} & -R^{-1} \end{bmatrix} \\ M &= \begin{bmatrix} m_{11} & m_{12} \\ m_{12} & m_{11} \end{bmatrix} \\ C(\chi) &= 0.5R^{-1}r^2m_cd\begin{bmatrix} 0 & \dot\chi_{13} \\ -\dot\chi_{13} & 0 \end{bmatrix} \\ D &= \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix} \\ m_{11} &= 0.25R^{-2}r^2(mR^2+I) + I_w \\ m_{12} &= 0.25R^{-2}r^2(mR^2-I) \\ m &= m_c + 2m_w \\ I &= m_cd^2 + 2m_wR^2 + I_c + 2I_m \\ \tau &= [\tau_1,\tau_2]^T \\ \tau_d &= [\tau_{d1},\tau_{d2}]^T \end{aligned}$$

where $\chi_{11} = x$, $\chi_{12} = y$ are the coordinates of P_0 and $\chi_{13} = \theta$ is the heading angle of the mobile robot, $\chi_{21} = v_1$, $\chi_{22} = v_2$ represent the angular velocities of right and left wheels, respectively and $\chi_{31} = i_{a1}$, $\chi_{32} = i_{a2}$ represent motor currents of right and left wheels, respectively. R is half of the width of the mobile robot and r is the radius of the wheel, d is the distance from the center of mass P_c of the mobile robot to the middle point P_0 between the right and left driving wheels. m_c and m_w are the mass of the body and the wheel with a motor, respectively. I_c, I_w, and I_m are the moment of inertia of the body about the vertical axis through P_c, the wheel with a motor about the wheel axis, and the wheel with a motor about the wheel diameter, respectively. The positive terms d_{ii}, $i = 1,2$, are the damping coefficients. $\tau \in \mathbb{R}^2$ is the control torque applied to the wheels of the robot. $\tau_d \in \mathbb{R}^2$ is a vector of disturbances including unmodeled dynamics. $K_T = diag[k_{t_1}, k_{t_2}]$ is the motor torque constant, $i_a = [i_{a_1}, i_{a_2}]$ is the motor current vector, $u \in \mathbb{R}^2$ is the input voltage, $R_a = diag[r_{a_1}, r_{a_2}]$ is the resistance, $L_a = diag[l_{a_1}, l_{a_2}]$ is the inductance, $K_E = diag[k_{e_1}, k_{e_2}]$ is the back electromotive force coefficient and $N = diag[n_1, n_2]$ is the gear ratio. Here, $diag[\cdot]$ denotes the diagonal matrix. Model (16) is discretized using the Euler methodology.

A. Neural Identification Design

To obtain a discrete-time neural model for the electrically driven nonholonomic mobile robot (16) we employ the identifier developed in [7], with $n = 7$ (the number of states) and trained with EKF, and is defined as follows

$$\begin{aligned} x_{1,k+1} &= w_{11,k}S(\chi_{11,k}) + w_{12,k}S(\chi_{12,k}) + \\ &\quad w'_{1_{11}}\chi_3 + w'_{1_{12}}\chi_4 \\ x_{2,k+1} &= w_{21,k}S(\chi_{11,k}) + w_{22,k}S(\chi_{12,k}) + \\ &\quad w'_{1_{21}}\chi_3 + w'_{1_{22}}\chi_4 \\ x_{3,k+1} &= w_{31,k}S(\chi_{11,k}) + w_{32,k}S(\chi_{12,k}) + \\ &\quad w'_{1_{31}}\chi_3 + w'_{1_{31}}\chi_4 \\ x_{4,k+1} &= w_{41,k}S(\chi_{11,k}) + w_{42,k}S(\chi_{12,k}) + \\ &\quad w_{43,k}S(\chi_{21,k}) + w_{44,k}S(\chi_{31,k}) + w'_2\chi_6 \\ x_{5,k+1} &= w_{51,k}S(\chi_{11,k}) + w_{52,k}S(\chi_{12,k}) + \\ &\quad w_{53,k}S(\chi_{22,k}) + w_{54,k}S(\chi_{32,k}) + w'_2\chi_7 \\ x_{6,k+1} &= w_{61,k}S(\chi_{11,k}) + w_{62,k}S(\chi_{12,k}) + \\ &\quad w_{63,k}S(\chi_{21,k}) + w_{64,k}S(\chi_{31,k}) + w'_3u_{11} \\ x_{7,k+1} &= w_{71,k}S(\chi_{11,k}) + w_{72,k}S(\chi_{12,k}) + \\ &\quad w_{73,k}S(\chi_{22,k}) + w_{74,k}S(\chi_{32,k}) + w'_3u_{12} \end{aligned} \tag{17}$$

where x_1 and x_2 identify the x and y coordinates, respectively; x_3 identifies the robot angle; x_4 and x_5 identify the angular velocities of right and left wheels, respectively; finally, x_6 and x_7 identify the motor currents, respectively. The NN training is performed on-line, and all of its states are initialized in a random way. The RHONN parameters are heuristically selected as:

$$
\begin{array}{lll}
P_1(0) = 1 \times 10^8 & R_1(0) = 1 \times 10^4 & Q_1(0) = 5 \times 10^5 \\
P_2(0) = 1 \times 10^2 & R_2(0) = 5 \times 10^4 & Q_2(0) = 5 \times 10^5 \\
P_3(0) = 1 \times 10^8 & R_3(0) = 1 \times 10^4 & Q_3(0) = 5 \times 10^5 \\
P_4(0) = 1 \times 10^2 & R_4(0) = 1 \times 10^1 & Q_4(0) = 1 \times 10^1 \\
P_5(0) = 1 \times 10^2 & R_5(0) = 1 \times 10^1 & Q_5(0) = 1 \times 10^1 \\
P_6(0) = 1 \times 10^2 & R_6(0) = 1 \times 10^3 & Q_6(0) = 1 \times 10^3 \\
P_7(0) = 1 \times 10^2 & R_7(0) = 1 \times 10^3 & Q_7(0) = 1 \times 10^3
\end{array}
$$

It is important to consider that for the EKF-learning algorithm the covariances are used as design parameters [20], [21]. The neural network structure (17) is determined heuristically in order to minimize the state estimation error.

B. Control Synthesis

In order to facilitate the controller synthesis, we rewrite neural network (17) in a block structure form as

$$
\begin{aligned}
x_{1,k+1} = \begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ x_{3,k+1} \end{bmatrix} &= w_{1,k}\varphi_1(\chi_{1,k}) + w_{1,k}\chi_{2,k} \\
x_{2,k+1} = \begin{bmatrix} x_{4,k+1} \\ x_{5,k+1} \end{bmatrix} &= w_{2,k}\varphi_2(\chi_{1,k}, \chi_{2,k}) + w_{2,k}\chi_{3,k} \\
x_{3,k+1} = \begin{bmatrix} x_{6,k+1} \\ x_{7,k+1} \end{bmatrix} &= w_{3,k}\varphi_3(\chi_{1,k}, \chi_{2,k}, \chi_{3,k}) \\
&\quad + w_{3,k}u_k
\end{aligned} \tag{18}
$$

with $\chi_{1,k}, \chi_{2,k}, \chi_{3,k}, \varphi_1, \varphi_2, \varphi_3, w_{1,k}, w_{2,k}, w_{3,k}, w_{1,k}, w_{2,k}$ and $w_{3,k}$ of appropriated dimension according to (18).

The goal is to force the state $x_{1,k}$ to track a desired reference signal $\chi_{1\delta,k}$. This is achieved by designing a control law as described in [7]. First the tracking error is defined as

$$z_{1,k} = x_{1,k} - \chi_{1\delta,k}$$

Then using (18) and introducing the desired dynamics for $z_{1,k}$ results in

$$
\begin{aligned}
z_{1,k+1} &= w_{1,k}\varphi_1(\chi_{1,k}) + w_{1,k}\chi_{2,k} - \chi_{1\delta,k+1} \\
&= K_1 z_{1,k}
\end{aligned} \tag{19}
$$

where $K_1 = diag\{k_{1_1}, k_{2_1}, k_{3_1}\}$ with $|k_{1_1}|, |k_{2_1}|, |k_{3_1}| < 1$. The desired value $\chi_{2\delta,k}$ for the pseudo-control input $\chi_{2,k}$ is calculated from (19) as

$$
\begin{aligned}
\chi_{2\delta,k} &= (w_{1,k})^{-1}(-w_{1,k}\varphi_1(\chi_{1,k}) + \chi_{1\delta,k+1} \\
&\quad + K_1 z_{1,k})
\end{aligned} \tag{20}
$$

At the second step, we introduce a new variable as

$$z_2 = x_{2,k} - \chi_{2\delta,k}$$

Then using (18) and introducing the desired dynamics for $z_{2,k}$ results in

$$
\begin{aligned}
z_{2,k+1} &= w_{2,k}\varphi_2(\chi_{1,k}, \chi_{2,k}) + w_{2,k}\chi_{3,k} - \chi_{2\delta,k+1} \\
&= K_2 z_{2,k}
\end{aligned} \tag{21}
$$

where $K_2 = diag\{k_{1_2}, k_{2_2}\}$ with $|k_{1_2}|, |k_{2_2}| < 1$. The desired value $\chi_{3\delta,k}$ for the pseudo-control input $\chi_{3,k}$ is calculated from (21) as

$$
\begin{aligned}
\chi_{3\delta,k} &= (w_{2,k})^{-1}(-w_{2,k}\varphi_2(\chi_{1,k}, \chi_{2,k}) + \chi_{2\delta,k+1} \\
&\quad + K_2 z_{2,k})
\end{aligned} \tag{22}
$$

At the third step, we introduce a new variable as

$$z_3 = x_{3,k} - \chi_{3\delta,k}$$

Taking one step ahead, we have

$$
\begin{aligned}
z_{3,k+1} &= w_{3,k}\varphi_3(\chi_{1,k}, \chi_{2,k}, \chi_{3,k}) \\
&\quad + w_{3,k}u_k - \chi_{3\delta,k+1}
\end{aligned} \tag{23}
$$

where u_k is defined as

$$
\begin{aligned}
u_k &= -\frac{1}{2}\left(R(z_k) + g^T(x_k)Pg(z_k)\right)^{-1} \\
&\quad \times g^T(x_k)P(f(x_k) - x_{\delta,k+1})
\end{aligned} \tag{24}
$$

where the controllers parameters are selected heuristically as

$$w'_{1,k} = \begin{bmatrix} \cos(x_3) & \cos(x_3) \\ \sin(x_3) & \sin(x_3) \\ R^{-1} & R^{-1} \end{bmatrix},$$

and where P, $w'_{2,k}$, $w'_{3,k}$ are defined as 2×2 identity matrices.

VI. Simulation Results

This section presents the simulation results of the proposed visual servoing scheme. The simulations have been performed using Matlab-Simulink.

The physical parameters of the mobile robot in the simulations were defined as

$$
\begin{array}{ll}
R = 0.75m & I_m = 0.0025kgm^2 \\
d = 0.3m & R_a = diag[2.5, 2.5]\Omega \\
r = 0.15m & L_a = diag[0.048, 0.048]H \\
m_c = 30kg & K_E = diag[0.02, 0.02]V/\,(rad/s) \\
m_w = 1kg & N = diag[62.55, 62.55] \\
I_c = 15.625kgm^2 & K_T = diag[0.2613, 0.2613]Nm/A \\
I_w = 0.005kgm^2 & d_{m1} = d_{m2} = 0.5N
\end{array}
$$

In the simulation the robot moves under the action of the proposed controller, the controller uses as references a linear velocity $v = 0.2m/s$ and the angular velocities estimated by the PSO-PBVS algorithm. In the simulation the initial pose of the robot is $[0 \;\; 0 \;\; 0]^\top$, and the desired pose is $[1.3156 0.4788 \; 0.3491]^\top$.

The sampling time of the simulation was T= 0.01s. Simulations results are presented as follows: In Fig. 2 we present the camera motion in the 3D space, computed by the proposed PSO-PBVS approach. Fig. 3 shows the identification performance for x-axis, y-axis and θ angle. Fig. 5 shows the trajectory tracking results. In Fig. 5 we present the tracking errors. In Fig. 6 we show the applied control signal for the left and right wheels. Fig. 7 presents the current identification for simulation in left and right wheels. Finally, Fig. 8 shows the angular velocity identification for simulation in left and right wheels.

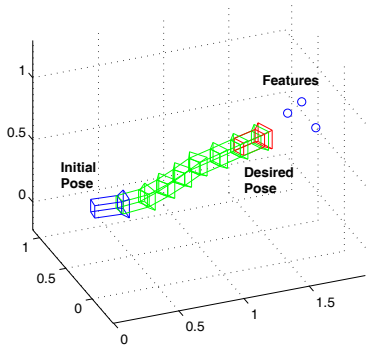

Fig. 2. Camera motion estimated by the proposed PSO-PBVS approach.

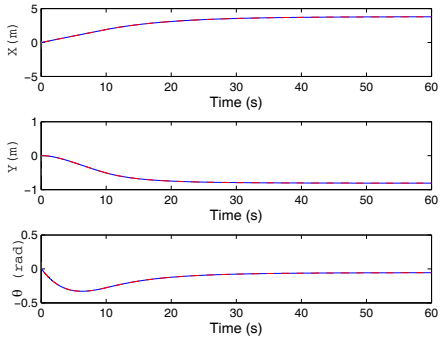

Fig. 3. x-axis identification (top), y-axis identification (middle) and θ angle (bottom), plant signal in solid line and neural signal in dashed line.

VII. Experimental Results

We have accomplished the presented methods in our mobile robot platform (9). The robot is equipped with a Kinect sensor, which is a sensor that provides vision data, i.e. an image, but it also provides the depth of this data. Therefore, we can retrieve the features 3D pose with respect to the current camera pose.

The target object containts three circles which are segmented using an hsv segmentation, then their centroids are used as features for the the task. The transformation between the current pose and the desired pose ${}^{c}\mathbf{T}_{c*}$, is computed

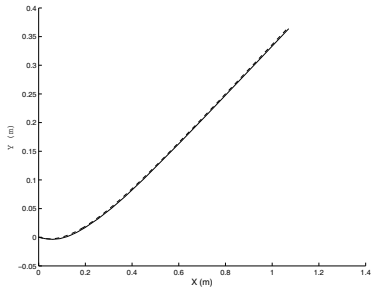

Fig. 4. Trajectory tracking result for simulation (plant signal in solid line and neural signal in dashed line).

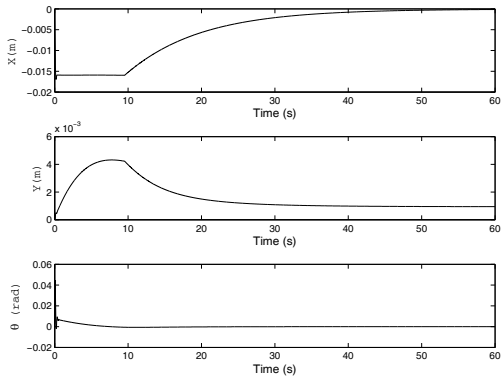

Fig. 5. Tracking errors, x-axis (top), y-axis (middle) and θ angle (bottom).

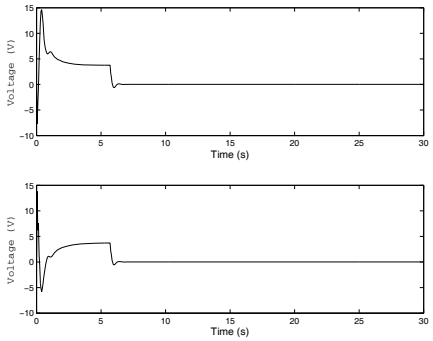

Fig. 6. Applied control signal for the left and right wheels respectively.

directly from the the features defined with respect to the current pose ${}^{c}\mathbf{X}$ and the features defined with respect to the desired pose the ${}^{c*}\mathbf{X}$, using [22]. The visual servoing task is shown in Fig. 10, the figure shows the trajectory computed with the proposed algorithms.

VIII. Conclusions

In this paper a new PBVS approach has been presented, namely the PSO-PBVS algorithm. This approach uses particle swarm optimization (PSO) to solve the visual servoing task. The PSO-PBVS allows to control the 3D trajectory of the robot, while keeping the visibility of the 3D features. In

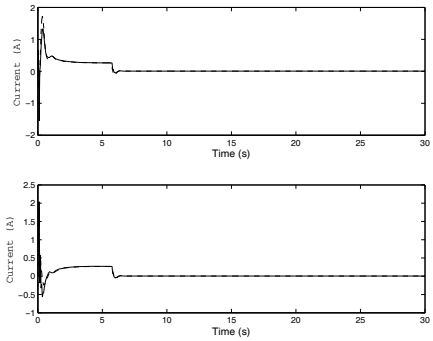

Fig. 7. Current identification for simulation in left and right wheels, respectively (plant signal in solid line and neural signal in dashed line).

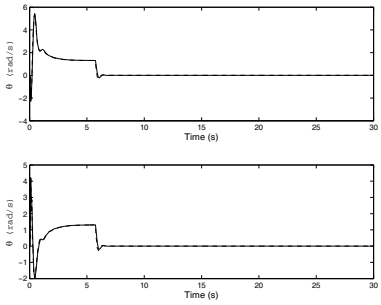

Fig. 8. Angular velocity identification for simulation in left and right wheels, respectively (plant signal in solid line and neural signal in dashed line).

Fig. 9. Differential drive robot, with kinect sensor

addition, since the algorithm does not require the inversion of any matrix (e.g. Jacobian or iteraction matrix), then it does not have singularities as conventional PBVS.

With the linear velocity and the angular velocity estimated by PSO-PBVS, an intelligent controller is used to estimate the currents for each motor, and also to ensure that the motors provide the desired velocities. The intelligent controller uses a discrete-time recurrent high order neural network (RHONN), trained with an EKF algorithm.

ACKNOWLEDGMENT

The authors want to thank CONACYT for supporting this work through projects CB-156567 and CB-103191.

Fig. 10. Trajectory of the robot during the experiment.

REFERENCES

[1] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, 1996.

[2] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 3, pp. 313–326, Jun. 1992.

[3] F. Chaumette and S. Hutchinson, "Visual servo control, part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 82–90, 2006.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[5] R. E. Y. Shi, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1998, pp. 69–73.

[6] K. Suresh, S. Ghosh, D. Kundu, A. Sen, S. Das, and A. Abraham, "Inertia-adaptive particle swarm optimizer for improved global search," in *Eight International Conference on Intelligent Systems Design and Applications (ISDA)*, 2008.

[7] M. Lopez-Franco, E. Sanchez, A. Alanis, and C. Lopez-Franco, "Discrete time neural control of a nonholonomic mobile robot integrating stereo vision feedback," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, Aug 2013, pp. 1–8.

[8] G. A. Rovithakis and M. A. Chistodoulou, *Adaptive Control with Recurrent High -Order Neural Networks*. pringer Verlag, Berlin, Germany, 2000.

[9] R. Grover and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley and Sons N. Y., 1992.

[10] A. Y. Alanis, E. N. Sanchez, A. G. Loukianov, and G. Chen, "Discrete-time output trajectory tracking by recurrent high-order neural network control," in *Decision and Control, 2006 45th IEEE Conference on*, Dec 2006, pp. 6367–6372.

[11] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," 1989.

[12] D. E. Kirk, *Optimal Control Theory: An Introduction*. Dover Publications, Apr. 2004. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0486434842

[13] F. L. Lewis and V. L. Syrmos, *Optimal Control*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1995.

[14] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. New York, NY, USA: Academic Press, 1995.

[15] A. Al-Tamimi, F. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 943 –949, aug. 2008.

[16] T. Ohsawa, A. Bloch, and M. Leok, "Discrete hamilton-jacobi theory and discrete optimal control," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, dec. 2010, pp. 5438 –5443.

[17] T. Das and I. Kar, "Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots," *Control Systems Technology, IEEE Transactions on*, vol. 14, no. 3, pp. 501 – 510, may 2006.

[18] K. Do, Z. Jiang, and J. Pan, "Simultaneous tracking and stabilization of mobile robots: an adaptive approach," *Automatic Control, IEEE Transactions on*, vol. 49, no. 7, pp. 1147 – 1151, july 2004.

[19] B. S. Park, S. J. Yoo, J. B. Park, and Y. H. Choi, "A simple adaptive control approach for trajectory tracking of electrically driven nonholonomic mobile robots," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 5, pp. 1199 –1206, sept. 2010.

[20] L. A. Feldkamp, D. V. Prokhorov, and T. M. Feldkamp, "Simple and conditioned adaptive behavior from kalman filter trained recurrent networks," *Neural Netw.*, vol. 16, no. 5-6, pp. 683–689, Jun. 2003. [Online]. Available: http://dx.doi.org/10.1016/S0893-6080(03)00127-8

[21] S. Haykin, *Kalman Filtering and Neural Networks*. John Wiley and Sons N. Y., 2001.

[22] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, jun 1994, pp. 935 –938.