Image-based Visual Servoing for Automatic Control of Balancing Beam

Saša Marković*, Josip Musić* i Mirjana Bonković*

* Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Split, Croatia
smarkovi@fesb.hr, imusic@fesb.hr, mirjana@fesb.hr

Abstract - While uncalibrated image based visual servoing is traditionally used in robotics, in the paper we propose its application on the problem of automatic control of balancing beam with two thrusters, one at each end. Classical approach to the problem is application of PID regulator or LQR (linear quadratic regulator). Our proposal has several important advantages over these classical approaches: knowledge of exact mathematical model is not needed, no issues with observability and controllability, no need for regulator parameter identification, no calibration is needed. In the paper we propose application of Broyden and population algorithms within image based visual servoing framework for control and stabilization of balancing beam and test it in simulation environment. Obtained results are analyzed and compared to results obtained with PID. Several improvements and possible future research directions are proposed and discussed.

I. INTRODUCTION

Visual servoing in robotics [1,2,3,4,5] has received a lot of attention in the last decade since associated theory has reached a level of development where excellent results can be achieved even for most complex systems/problems. Particularly interesting category of visual servoing is uncalibrated model free image-based visual servoing due to its low cost and simplicity [4,5]. Here, the term image-based signifies that error signal is not defined in 3D world coordinate frame but rather directly in image space/plane while term model free indicates that no kinematic or dynamic model of the system under consideration is needed. The final term, uncalibrated indicates that no camera calibration is needed. Specified advantages, which are possible only with estimation of total Jacobian matrix (which includes both the image and system Jacobian matrix), significantly facilitate practical applications of visual servoing. Nevertheless, real-time Jacobian matrix estimation is usually extremely difficult and computationally demanding task due to matrix nonlinearity, high dimensionality, time-varying elements and ill-conditioning [5,6]. Thus, several characteristic approaches to the problem, usually based on optimization techniques with least square methods, were developed. These include Broyden method [1,2,3,4,5,6,7], Piepmeyer method [3,5], Kalman filter and more recently particle filter [5] and population-based method [4,5]. Due to iterative nature of the algorithms only knowledge of Jacobian matrix from the previous time instance as well as current measurements (i.e. image coordinates and system states) is required to estimate Jacobian matrix for current time instance. It should be noted that calibration and/or dynamic model would improve accuracy of visual servoing but at the same time some of previously mentioned desired properties (guarantying simplicity and low-cost) would be lost. All of previously mentioned characteristics of uncalibrated model free image-based visual servoing make it appropriate for application in automatic control of number of systems which include some of the classical control problems. One of them is beam and ball problem with its nonlinear and unstable open-loop characteristics. Since the ball cannot be controlled directly but only by means of beam rotation this problem is often considered benchmark problem [8,9] for testing of control algorithms and is thus appropriate and important model for teaching control engineering. Thus it is surprising that only several such applications can be found in available literature [8,9,10] with the remark that in some of them [9] computer vision was only used as a sensor for some other control scheme with calibration for added precision. Classical approach to solving this problem is to either mathematically model the system and then linearize it around working point applying linear quadratic regulator (LQR) [11] or using PID regulator [8] and associated parameter estimation algorithms which do not guaranty optimality of the solution in respect to measurement error. Additional control schemes like fuzzy control [12], state observers [13] and phase lead compensation [14] are also used.

Since the developed control scheme and visual servoing algorithms are intended for education purposes, in the paper a simplified and slightly altered control problem is considered in order to be more accessible to undergraduate control engineering students. The problem is now formulated as stabilization of balancing beam (without the ball) in the desired orientation which can be static or dynamically changing. The beam was built with two thrusters (one at each end of the beam) as can be seen in figure 1. Thrusters act as actuators and provide necessary forces to achieve control effect. But before the control scheme can be implemented, simulation trials should be carried out in order to verify applicability of visual servoing to the control problem as well as test for various situations in a secure environment and controlled conditions. This is the main focus of the paper. We note that because of very simple plant it is not expected that visual servoing will produce better results when compared to classical approaches. Full advantages of visual servoing can be expected for more complex systems. The remainder of the article is organized as follows.
Section 2 describes Jacobian matrix estimation algorithms as well as standard PID regulator. Section 3 presents developed simulation model using object oriented approach with Modelica and Dymola. Section 4 summarizes and discusses obtained simulation results while section 5 contains conclusions and directions for possible future research and implementation.

II. ALGORITHMS

The visual servoing problem can be solved using quasi-Newton methods [4,6], which consider at each iteration the linear model of form:

\[ L_k(x, B_k) = F(x_k) + B_k(x - x_k) \]

where \( F(x_k) \) is a continuously differentiable function, and \( B(x-x_k) \) is the function estimate at previous instance.

A. Broyden algorithm

Broyden [7] proposed the most successful class of quasi-Newton methods based on the secant equations, imposing the linear model to exactly match the nonlinear function at iterates \( x_k \) and \( x_{k-1} \) i.e.

\[ L_{k+1}(x_k) = F(x_k) \]
\[ L_{k+1}(x_{k+1}) = F(x_{k+1}) \]

Subtracting these two equations and defining the difference as \( y_k = F(x_{k+1}) - F(x_k) \) and \( s_k = x_{k+1} - x_k \) as the step representing the difference between two consecutive points at which the function has been observed, we obtain the classical secant equation:

\[ y_k = B_{k+1}s_k \]

If the dimension \( n \) of \( y_k \) is strictly greater than 1, there are an infinite number of matrices satisfying (3). The “least-change secant update”, proposed by Broyden, includes following steps:

- minimize \( \|B_{k+1} - B_k\| \)
- use constraint expressed with (3)

which, after some mathematical manipulations, lead to

\[ B_{k+1} = B_k + \eta \left( \frac{y_k - B_k s_k}{s_k^T s_k} \right) s_k^T \]

Parameter \( \eta \in (0,1) \) is introduced to control Jacobian update rate and to prevent singularity of matrix \( B \) [4]. Broyden method has proved successful in number of applications but is sensitive to noise [1,2,3,4,5,6].

B. Population-based algorithm

Recently, “population based” generalization of Broyden update, has been recommended [4,5] in order to improve robustness and stability where at each iteration, the finite population of iterates \( x_0, \ldots, x_{k+1} \) is maintained. The method also belongs to quasi-Newton framework, with \( B_{k+1} \) computed as

\[ B_{k+1} = \arg\min_{J} \left\{ \sum_{i=0}^{k} \omega_i \| F(x_i) - B_i s_i \|^2 + \| J - B_{k+1}J \|^2 \right\} \]

where \( B_{k+1}^0 \in \mathbb{R}^{n \times n} \) is an a priori approximation of \( B_{k+1} \). The role of the second term in equation (5) is to overcome the under-determination of the least-square problem based on the first term and also to control the numerical stability of the method. The matrix contains weights associated with the arbitrary term \( B_{k+1}^0 \), and the weights \( \omega_i \in \mathbb{R}^+ \) are associated with the previous iterates. Equation (5) can be rewritten in matrix form as

\[ b_{k+1} = \arg\min_{j} \left\{ J(S_j J_m)^T \left( \begin{array}{cc} \Omega & 0 \\ 0 & I \end{array} \right) - (s_{k+1}^T J_x B_{k+1}^0 \left( \begin{array}{cc} \Omega & 0 \\ 0 & I \end{array} \right)^{-1} \right) \right\} \]

where \( \Omega \in \mathbb{R}^{k \times k} \) is a diagonal matrix with weights \( \omega_i \) on the diagonal for \( i=0, \ldots, k \). Now the following update equation is obtained

\[ B_{k+1} = B_{k+1}^0 + (Y_{k+1} - B_{k+1}^0 S_{k+1}) \Omega^{-1} S_{k+1}^T (\Gamma^2 + S_{k+1} \Omega^2 S_{k+1}^T)^{-1} \]

where \( Y_{k+1} = (y_0, y_1, \ldots, y_k) \) and \( s_{k+1} = (s_0, s_1, \ldots, s_k) \). Update equation (7) exhibits good properties for \( B_{k+1}^0 = B_{k+1} \) and its local convergence has been proved [4]. The weight \( \omega_i \) captures the relative importance of each iteration in the population, and the matrix \( \Gamma \) captures the importance of the arbitrary terms defined by \( B_{k+1}^0 \) for the identification of the linear model. The weights have to be finite and \( \Gamma \) must be such that \( \Gamma^2 + S_{k+1} \Omega^2 S_{k+1}^T \) is safely positive definite. To ensure this, we have found that it should be defined through simulations as a small positive constant [4].
Population method exhibit faster convergence and a greater robustness than quasi-Newton methods (e.g. Broyden).

C. PID regulator

PID controller is one of the oldest control schemes. It is used to achieve desired system response in terms of steady state error, overshoot, time of overshoot and/or rise time. Its effects on measurement error (difference between desired and actual beam angle value) are threefold: proportional (P), integration (I) and derivational (D). It should be noted that in the paper the term PID refers to ideal parallel realization. Ideal means that D term is ideal which cannot be realized practically due to noise issues, while parallel refers to the fact that angle error signal is simultaneously fed to all three terms and the final output is their summation. The equation describing behavior of PID used in the paper is

\[ u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt} \tag{8} \]

where \( u(t) \) is PID output, \( e(t) \) is error signal, \( K_p \) proportional constant, \( K_i \) integration constant and \( K_d \) derivative constant. There are several methods which help in determining these constants (ultimate cycle method, Cohen-Coon method and Ziegler-Nichols method) but none of them can guarantee optimal selection of parameters thus human intervention is needed based on the knowledge of effects which individual PID terms have on system response. In the paper we adopted the following approach: proportional term was increased to the level that steady state error was small enough while at the same time keeping system stable. Then integration term was corrected to further eliminate the error and decrease rise time. Finally, derivative constant is corrected to reduce and/or eliminate overshoot and settling time.

III. SIMULATION

A. Modelica-Dymola model of balancing beam

Modelica-Dymola modeling and simulation environment was chosen for this experiment because it does not require knowledge of mathematical model to build a simulation model. This can be achieved because Modelica is object oriented language used for acasual modeling, as opposed to some other widely used simulation environments (like Simulink) that use casual modeling. In acasual modeling state equations are written in neutral form without consideration for computational order, while in casual modeling computational order needs to be considered. Commercially available programming environment based on Modelica language which enables simple and easy graphical user interface based programming is Dymola from Dynasim. Dymola offers wide range of features such as: acasual modeling, reusability, hierarchical structure of models, symbolic and numerical solving of system equations, algebraic loop solving, hybrid system modeling, 3D animation of modeled system, etc. Important feature of Dymola is Dymola-Simulink interface. The DymolaBlock in Simulink is shielded around an S-function MEX block i.e. the interface to the C-code generated by Dymola for the Modelica model. This enables us to take advantage of both programming environments: modeling in Modelica-Dymola without the need to develop mathematical model and simulation in Simulink where full potential of Matlab can be used.

Designed Modelica-Dymola model of balancing beam can be seen in figure 2 where world icon is not a part of physical model, but rather the requirement of Modelica. BodyBox1 and BodyBox2 present the beam, while BodyBox3 is necessary for connecting these parts to revolute joint which is in turn connected to BodyBox4 which presents the vertical pillar on which the beam system is suspended. In order to know the system state at given time instance two sensors were added to the model (although it should be noted that the same result can be achieved with only one sensor). This is illustrated in figure 3. Upper sensor measures relative angle of the balancing beam in respect to horizontal, while lower sensor measures angular velocity. These quantities present balancing beam model output. It should be noted that same quantities are readily available in real world system through potentiometer measurements which is attached to the revolute bearing (figure 1).

![Figure 2 Modelica-Dymola balancing beam model](image)

![Figure 3 Modelica-Dymola model with angle and angular rate sensors](image)

Since system under consideration has thrusters as actuators for realization of desired control actions it was necessary to include resulting force into the model as is shown in figure 4.
In simulation force acting on only one end of the beam was used since it could obtain both the positive and negative values. In real world system (figure 1) two thrusters (one mounted at each end) are used since they can produce only positive forces. Necessary blocks were added in order to ensure that force acting on the beam is always perpendicular to it so to correspond to real world situation (top of figure 4). In practical application there exists a nonlinear relation between force exerted by thrusters and propeller which needs to be considered. In simulations we assumed that the force control loop is achieved. Also camera model shown in figure 5 was added so that desired angle (input to the model) was transformed into x and y coordinates of beam’s end. In the figure this is represented by yellow square. Note that two such squares are present in the figure: upper one executes described operation for actual beam position while the lower one executes it for desired position. This is also the case for all other squares in the figure. Next, red square transforms beam’s end coordinates into camera coordinate frame using homogenous matrix. Translation of 1.5 m along z axis and constant rotation of 15° around x axis is assumed for this step. Finally, green square transforms coordinates from camera coordinate frame into image plane coordinate frame using known camera parameters such as focus (data from real camera were used). Final model thus has 2 inputs (desired angle and force control signal) and 6 outputs (current angle and angular velocity, desired and actual x and y coordinates in image plane) as can be seen in figure 5.

B. Simulink model for control application

Modelica-Dymola programming environment offers possibility of exporting built model in the Simulink-Matlab environment where more in-depth analysis and data presentation can be achieved. Thus Simulink model with imported Dymola-Modelica was constructed as can be seen from figure 6. The figure contains five distinct parts highlighted by different colors. Blue part named “Desired angle” contains source blocks which produce static or dynamic angle trajectory depending on manual switch position. This angle defined in degrees is then transformed by simple gain into radians which are required for proper functioning of other simulation blocks. Red square labeled “Dymola model” contains imported Modelica-Dymola model described in preceding section and presented in figure 5. Green markings on figure 6 highlight visual servoing part of simulation. Here, feature vector containing current angle value as well as current and desired x and y image plane coordinates is fed via multiplexer into two blocks (Level-2 M file S-Functions), one containing Broyden algorithm code and other population-based method. Although both M-files are executed simultaneously during simulation only one output selected with multiport switch is forwarded to proceeding blocks for calculation. Output from selected M-file is estimated Jacobian matrix (in our case it is actually a 2x1 vector). This estimation is inverted and multiplied with error vector containing difference between actual and desired image plane coordinates of balancing beam end. Output from this part of simulation is (required) angular velocity for next time step. Required angular velocity is then fed into yellow part of simulation scheme which is responsible for transforming it into required force acting on balancing beam. In order to achieve this in simulation, simplified mathematical model based on Euler-Lagrange mechanics was derived

\[ F = \frac{m \cdot g \cdot d}{l} \sin \phi - \frac{l}{l} \alpha \] (9)

where \( m \) is total beam mass (without thrusters), \( g \) gravity constant, \( d \) distance of beam’s center of mass from rotation axis, \( l \) half the length of the beam, \( I \) is beam’s moment of inertia, \( \phi \) current angle in respect to Earth surface, \( \alpha \) is angular acceleration and F applied force. Since angular acceleration is not readily available from the model we used the following equation

\[ \alpha = \frac{\dot{\phi}_2 - \dot{\phi}_1}{\Delta t} \] (10)
where $\Delta t$ is simulation time step, $\dot{\phi}_2$ is desired angular velocity (output from visual servoing) and $\dot{\phi}_1$ is current angular velocity (Modelica-Dymola output). It should be noted that in real application simplified mathematical model might not be sufficient but system identification can be applied. Generated force signal is inverted in purple square because of negative feedback and is multiplied with constant gain (P controller).

For PID controller the same scheme could not be used due to different Modelica-Dymola model (no need for camera model) and no need for angular velocity/force conversion. Simulation scheme for PID regulator is depicted in figure 7 containing three main parts. Part in blue square is the same as in case of visual servoing while red square highlights ideal parallel PID regulator with angle error as input and force control signal as output. Green square pertains to Modelica-Dymola model which is different from visual servoing case and is depicted in figure 4. Please note that angular velocity output still exists in Dymola model but is not used in simulation.

IV. RESULTS AND DISCUSSION

Simulation experiments were achieved for two cases: static and dynamic target position. For each of the two cases Broyden and population algorithms were applied as well as standard PID controller. Broyden method parameter $\eta$ was set to 1, while population size was set to 50. PID regulator was tuned so that its response was quick: $K_p$ was set to 30, $K_I$ to 3 and $K_D$ to 5. It should be noted that in visual servoing case desired target position was defined as coordinates in camera image plane while for PID this could not be done (since no camera is required) thus target position was defined as balancing beam angle. For comparison reasons angle results will be presented for both the static and dynamic case. Comparison of obtained results for static case are presented in figure 8.

As can be seen from the figure all of the methods successfully converge to desired target position ($-10^\circ$) from the initial position ($15^\circ$). Visual servoing methods are somewhat slower in their response with 0.235° steady state error for Broyden and 0.246° for population method but with no overshoot. On the other hand, PID response is faster with somewhat smaller steady state error (0.057°) when compared to visual servoing methods but has overshoot of 12.4% which might present a problem for certain applications. PID regulator could have been tuned differently to have slower response with smaller overshoot or even leaving out the D term which would altogether eliminate the overshoot (but it would then be PI controller). This might be a valid approach in practical
implementation since D term cannot be implemented in ideal form and is sensitive to noise prompting the use of filtering algorithms such as low-pass filtering or median filtering. Figure 9 presents the dynamic target case. From the figure conclusion that PID performs better than visual servoing methods can be made.

![Figure 9 Simulation results for dynamic target](image)

But that would be only a partial conclusion. As was concluded for static case PID controller exhibits faster response time thus it better (or more quickly) tracks the target orientation then visual servoing. Also because of quick response and overshoot in static case it exhibits more sensitive behavior to abrupt changes in desired angle signal. If figure is analyzed in more detail it can be seen that visual servoing methods also track target with good agreement but with almost constant lag. This again is in agreement with static case were asymptotical response was observed. Improved response might be obtained if instead of simple gain (P term) in feedback loop (purple square in figure 6) some other block (PD term) was used or speed of desired trajectory lowered. Better insight into the issue can be provided by figure 10 which presents same results but in camera image plane. In the figure time is implicitly included thus lag induced error is not visible. Visual servoing methods performance now looks better since it tracks desired trajectory very closely so that they can be hardly distinguished (for better visibility additional markers with colors corresponding to appropriate methods are included: x for population and dot for referent signal).

![Figure 10 Enlarge portion of image plane for dynamic target tracking in visual servoing](image)

It should be noted that some of other standard approaches like LQR might exhibit significantly degraded performance then PID since they rely on model linearized around working point which is constantly changing. Also although Broyden and population algorithm results are very similar with population method having slightly better response but with significantly higher computational requirements we expect significantly larger degradation of Broyden performance (due to noise sensitivity) then population based method in practical application as was observed in [4].

V. CONCLUSIONS

In the paper application of model free uncalibrated visual servoing was applied to a problem of balancing beam stabilization and trajectory following. PID controller was used for comparison purposes as standard solution. Obtained results validated the proposed approach with good agreement with desired signals both in static and dynamic case (where certain amount of constant lag was present). In static case errors were as follows: 0.057° for PID regulator, 0.235° for Broyden algorithm and 0.246° for population based method. Future development of the approach might include application of other estimation algorithms like Kalman and Particle filter in the visual servoing loop especially for more complicated systems under automatic control, application to real world balancing beam and extension to ball and beam problem.

LITERATURE