INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Advanced Robotic Systems

Visual Trajectory-Tracking Model-Based Control for Mobile Robots

Regular Paper

Andrej Zdešar[1,*], Igor Škrjanc[1] and Gregor Klančar[1]

1 University of Ljubljana, Faculty of Electrical Engineering, Laboratory of Modelling, Simulation and Control, Slovenia
* Corresponding author E-mail: andrej.zdesar@fe.uni-lj.si

Received 1 Jun 2012; Accepted 12 Jun 2013

DOI: 10.5772/56757

Abstract In this paper we present a visual-control algorithm for driving a mobile robot along the reference trajectory. The configuration of the system consists of a two-wheeled differentially driven mobile robot that is observed by an overhead camera, which can be placed at arbitrary, but reasonable, inclination with respect to the ground plane. The controller must be capable of generating appropriate tangential and angular control velocities for the trajectory-tracking problem, based on the information received about the robot position obtained in the image. To be able to track the position of the robot through a sequence of images in real-time, the robot is marked with an artificial marker that can be distinguishably recognized by the image recognition subsystem.
Using the property of differential flatness, a dynamic feedback compensator can be designed for the system, thereby extending the system into a linear form. The presented control algorithm for reference tracking combines a feedforward and a feedback loop, the structure also known as a two DOF control scheme. The feedforward part should drive the system to the vicinity of the reference trajectory and the feedback part should eliminate any errors that occur due to noise and other disturbances etc. The feedforward control can never achieve accurate reference following, but this deficiency can be eliminated with the introduction of the feedback loop. The design of the model predictive control is based on the linear error model. The model predictive control is given in analytical form, so the computational burden is kept at a reasonable level for real-time implementation. The control algorithm requires that a reference trajectory is at least twice differentiable function. A suitable approach to design such a trajectory is by exploiting some useful properties of the Bernstein-Bézier parametric curves.
The simulation experiments as well as real system experiments on a robot normally used in the robot soccer small league prove the applicability of the presented control approach.

Keywords Visual Servoing, Trajectory-Tracking, Two DOF Control, Differential Flatness, Error Model Predictive Control

1. Introduction

Visual servoing (VS) is a technique which uses an image sensor in a feedback loop for motion control of a robot. The field of VS combines robotics, machine vision and control theory. An extensive overview of the VS applications and methodology was given by Corke [1, 2], Chaumette and Hutchinson [3, 4], Kragic and Christensen [5], and Chang [6] etc. Visual servoing can be found in a variety of applications: cooperative movement of mobile soccer robots [7], navigation of autonomous mobile robots

[8, 9], docking of autonomous water surface vehicles [10], helicopter and quadrocopter hovering and guidance [11–14], autonomous landing of aeroplanes [15–17], attitude control of satellites [18], grasping and movement of objects [19, 20], and keeping the relative view in a dynamic environment [21] etc.

The VS approaches are normally divided into three main groups [1]: position-based visual servoing (PBVS), image-based visual servoing (IBVS) and hybrid visual servoing. The methods differ in the definition of the control error. In the PBVS case, the control error is defined as the difference between the desired and current pose in the world (Euclidean) space. As opposed to the classical approach of the PBVS, the IBVS (implemented in this paper) defines the control error between the desired and current pose directly in the image coordinate frame (in pixels). The PBVS usually generates better motion, but if the system is not precisely calibrated it may not be able to eliminate the steady-state error. The IBVS can usually achieve an error-free motion, but the generated motion may not be optimum or may sometimes even produce unnecessary or even undesirable motion. The control algorithms that try to combine the useful properties of both approaches are called hybrid VS methods [22]. The VS algorithms may further be divided based on the configuration of the camera and the robot: eye-in-hand and eye-to-hand configuration; and the number of cameras: one camera, two cameras (stereo configuration), multiple cameras (more than two). Some visual servoing approaches take advantage of some special structural properties that can be used implicitly in the design of the control algorithm, e.g., planar surfaces [23]. In this paper we consider an eye-to-hand configuration with one camera. The mobile robot is observed by an overhead camera that is placed at arbitrary inclination with respect to the ground plane. In other words, the arbitrary positioned camera is used to provide information about the position of the mobile robot that moves on a flat surface.

A camera is considered to be an inexpensive and non-invasive sensor, and the information about the environment it provides is extremely rich compared to the other sensors, e.g., laser or ultrasonic distance sensors. This makes the camera an extremely appealing sensor for a broad range of applications. The huge amount of data the images can provide represent a challenge, particularly when it comes to extracting the relevant information for the specific task in real-time. A classical approach to the design of visual servoing consists of image acquisition, image segmentation and classification, high-level reasoning and decision-making, movement planning and, finally, execution of appropriate actions [1]. Although some approaches for visual servoing describe the task on a high (abstract) level [24], the modern approaches consider the action generation directly on the acquired image features to reduce the computational burden. The main idea is that the complex image signal can be described by a relatively small set of image features (like SIFT [25]), and these features are then used in the controller for the calculation of the appropriate actions [8, 18]. However, the classical high-level approach is useful in a process when the system is learning a new task, but when the task is learned, the system should carry it at low level to speed up the execution. In this paper, the machine vision is given a task of measuring the position of the robot in the image frame. To simplify and speed up the segmentation of the robot from the surrounding environment, the robot can be equipped with a marker (special colour or pattern) that can be indistinguishably detected by the image recognition system [7, 13].

The problem of designing a controller for trajectory-tracking (smooth movement along the predefined path) is one of the fundamental problems in robotics. Another fundamental problem in robotics is posture stabilization (movement from point-to-point), but we do not discuss this here. Over the years many different approaches have been developed to tackle the trajectory-tracking problem [26–29]. Some methods for trajectory-tracking are also able to accommodate the path on-line for obstacle avoidance [30, 31]. A good overview of the path planning methods can be found in [32].

The main emphasis of this paper is on the following topics:

- The trajectory-tracking task is supposed to be given in the image space, and hence the IBVS is adopted as the control scheme.
- The design of the control algorithm takes into account arbitrary inclination of the camera with respect to the ground plane.
- The system states are estimated from the delayed measurements using a Kalman filter.
- The states of the robot needed in the control law are all estimated in the image frame (no explicit conversion to the world frame is made).
- The trajectory-tracking control law is developed in the model predictive framework. The entire control algorithm is designed in the discrete space for optimum performance on a digital computer.
- For the path planning, the use of parametric curves under perspective projection is studied.
- The overall control algorithm was experimentally tested for robustness.

This paper is structured as follows. Section 2 gives an overview of the system along with all the mathematical equations describing the mobile robot and camera. This is followed by section 3 which presents the design of the controller. The system is linearized with the introduction of a non-linear compensator, which is described in section 3.1. In section 3.2 the model predictive control for trajectory-tracking is presented. Since all the states required by the control algorithm are not directly measurable, a state observer in the form of a Kalman filter is needed, which is presented in section 3.3. In section 4 the approach for designing a trajectory based on Bernstein-Bézier splines is presented. Afterwards, section 5 presents experimental results. Finally in section 6, some conclusions are drawn and ideas for future work are presented.

2. System overview

If not specified differently, we use small bold letters (e.g., $\boldsymbol{x}$) for column vectors, and big bold letters for matrices (e.g., $\boldsymbol{X}$). If needed, we use the subscript $(\cdot)_w$ to denote the world coordinate frame, $(\cdot)_p$ for the picture coordinate frame, and $(\cdot)_c$ for the camera coordinate frame. To describe the position and orientation of an object in a plane, we use generalized coordinates $\boldsymbol{q}_w^T = [x\ y\ \varphi]$ and $\boldsymbol{q}_p^T = [u\ v\ \theta]$ with respect to the world and image (picture) frame, respectively. For denoting just a point in a plane we use $\boldsymbol{p}_w^T = [x\ y\ 1]$ and $\boldsymbol{p}_p^T = [u\ v\ 1]$ with respect to the world and image coordinate frame, respectively. Note that we distinguish between bold and regular face symbols, and that we may use the same symbol to denote points in homogeneous and non-homogeneous coordinates interchangeably.

The system consists of a two-wheeled differentially driven mobile robot and a camera that observes the robot from an inclined angle with respect to the ground plane, on which the robot can move freely. Next, we describe the kinematic model of the robot and projective transformation of the camera.

2.1. Mobile robot kinematics

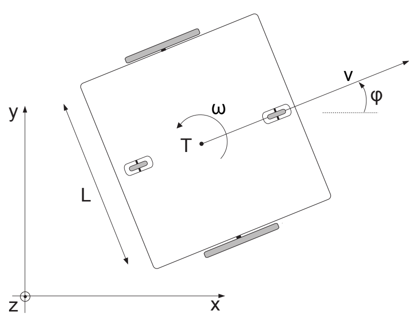


Figure 1. Two-wheeled differentially driven mobile robot.

The robot's architecture is shown in Figure 1. The kinematic motion equations of a two-wheeled differentially driven mobile robot are the same as those of a unicycle [32]. The kinematic model has a non-integrable constraint:

$$\boldsymbol{A}(\boldsymbol{q}_w)\dot{\boldsymbol{q}}_w = \dot{x}\sin\varphi - \dot{y}\cos\varphi\,, \tag{1}$$

which results from an assumption that the robot cannot slip in a lateral direction. The $\boldsymbol{A}(\boldsymbol{q}_w)$ in (1) is the constraint matrix defined over the generalized coordinates $\boldsymbol{q}_w$. Expressing all the achievable velocities of the mobile robot as a linear combination of the vector fields $\boldsymbol{s}_i(\boldsymbol{q}_w)$ that span the null space of the matrix $\boldsymbol{A}(\boldsymbol{q}_w)$ yields the *first order kinematic model* [32]:

$$\dot{\boldsymbol{q}}_w(t) = \left[\boldsymbol{s}_1\ \boldsymbol{s}_2\right]\begin{bmatrix}\nu(t)\\ \omega(t)\end{bmatrix} = \begin{bmatrix}\cos(\varphi(t)) & 0\\ \sin(\varphi(t)) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\nu(t)\\ \omega(t)\end{bmatrix}. \tag{2}$$

2.2. Camera

The transformation between the world point $\boldsymbol{p}_w^T = [x\,y\,z\,1]$ and the corresponding point in the image frame $\boldsymbol{p}_p^T = [u\ v\ 1]$ (in pixels) can be described by a pinhole camera model [33]:

$$\lambda \boldsymbol{p}_p = \boldsymbol{S}\left[\boldsymbol{R}\ \boldsymbol{t}\right]\boldsymbol{p}_w\,,\ \boldsymbol{S} = \begin{bmatrix}\alpha & \gamma & u_c\\ 0 & \beta & v_c\\ 0 & 0 & 1\end{bmatrix}, \tag{3}$$

where the λ is a scalar weight, the matrix $\boldsymbol{S}$ holds intrinsic camera parameters, while the matrix $\boldsymbol{R} = [\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{r}_3] \in \mathbb{R}^3 \times \mathbb{R}^3$ and vector $\boldsymbol{t} \in \mathbb{R}^3$ describe the camera orientation and position, respectively. The intrinsic camera parameters contained in the matrix $\boldsymbol{S}$ are: the scaling factors α and β in horizontal and vertical direction, respectively; the optical axis centre (u_c, v_c); and the skew γ. The model is non-linear because of the dependence $\lambda = \lambda(\boldsymbol{p}_w)$.

If the world points are confined to a common plane, the relation (3) simplifies. Without loss of generality, we can assume that the plane spans the axis vectors x and y $(z = 0)$:

$$\lambda \boldsymbol{p}_p = \boldsymbol{S}\left[\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{t}\right]\boldsymbol{p}_w = \boldsymbol{H}\boldsymbol{p}_w\,, \tag{4}$$

where we have taken advantage of the notation by redefining the $\boldsymbol{p}_w^T = [x\ y\ 1]$. The matrix $\boldsymbol{H} \in \mathbb{R}^3 \times \mathbb{R}^3$ is known as *homography* and presents the mapping between the points in the world frame and the corresponding points in the image plane. In a special configuration, when the image frame of the camera is aligned with the world frame, the homography takes a special form:

$$\boldsymbol{H} = \begin{bmatrix}s_u & 0 & t_u\\ 0 & s_v & t_v\\ 0 & 0 & 1\end{bmatrix}, \tag{5}$$

which means that the coordinates of a point in the world plane are just scaled and translated to the image: $u = s_u x + t_u$ and $v = s_v x + t_v$; where the tuples of values (s_u, t_u) and (s_v, t_v) represent the scaling and translation factor in the horizontal and vertical direction, respectively. Such a canonical configuration can greatly simplify the design of the in plane object tracking, and it is the configuration used in mobile soccer small league [7]. Denoting $\boldsymbol{H}^T = [\boldsymbol{h}_1\ \boldsymbol{h}_2\ \boldsymbol{h}_3]$, from the equation (4) follows:

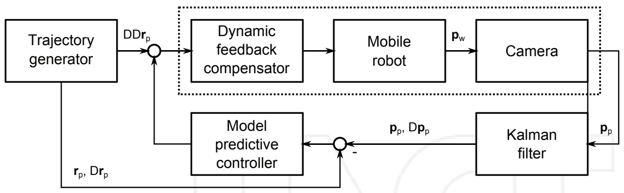
$$u = \frac{\boldsymbol{h}_1^T\boldsymbol{p}_w}{\boldsymbol{h}_3^T\boldsymbol{p}_w} \quad \text{and} \quad v = \frac{\boldsymbol{h}_2^T\boldsymbol{p}_w}{\boldsymbol{h}_3^T\boldsymbol{p}_w}\,. \tag{6}$$

Note that the denominator in the equations (6) is equal to the factor $\lambda = \boldsymbol{h}_3^T\boldsymbol{p}_w$ in the equation (4).

The whole system given with the equations (2) and (6) is multi-variable and non-linear. Furthermore, the system (2) is non-holonomic which makes the visual servoing non-trivial since Brockett's theorem (1983) shows that no linear time-invariant controller can control it [2].

3. Control design

The design of the control law is divided into several subsystems. First, it can be shown that the system under consideration, given in the equations (2) and (6), is a differentially flat system with respect to the position of the robot in the image reference frame [27]. Therefore, the system can be linearized by a non-linear dynamic compensator. Based on the obtained linear model, a model predictive control for trajectory-tracking is derived. Since some states needed in the controller are not directly measurable, they must be estimated, and this is achieved with the use of a Kalman filter. The overall control scheme is shown in Figure 2.



Trajectory generator
DD$\mathbf{r}_p$
Dynamic feedback compensator
Mobile robot
$\mathbf{p}_w$
Camera
Model predictive controller
$\mathbf{p}_p$, D$\mathbf{p}_p$
Kalman filter
$\mathbf{p}_p$
-
$\mathbf{r}_p$, D$\mathbf{r}_p$

Figure 2. Control scheme. (Operator D denotes the differentiation.)

3.1. *Dynamic feedback linearization*

How the property of differential flatness can be used in the design of trajectory-tracking and posture stabilization controllers for mobile robots has already been shown in [34]. The control method was later extended to a case where the camera is observing the mobile robot from an arbitrary inclination [26, 27]. Here, we give a brief summary of the approach for dynamic feedback linearization of the system.

A general guideline to obtaining a dynamic feedback compensator is to successively differentiate the system outputs until the system inputs appear in a non-singular way [34]. At some stage, an introduction of integrators on some of the inputs may be necessary to avoid subsequent differentiation of the original inputs.

Let us find the first derivative of the equation (6) with respect to time:

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \frac{1}{\boldsymbol{p}_w^T(\boldsymbol{h}_3\boldsymbol{h}_3^T)\boldsymbol{p}_w} \begin{bmatrix}\boldsymbol{p}_w^T(\boldsymbol{h}_3\boldsymbol{h}_1^T - \boldsymbol{h}_1\boldsymbol{h}_3^T)\\ \boldsymbol{p}_w^T(\boldsymbol{h}_3\boldsymbol{h}_2^T - \boldsymbol{h}_2\boldsymbol{h}_3^T)\end{bmatrix} \begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}^T \begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \boldsymbol{F}\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}, \quad (7)$$

where $\boldsymbol{F} \in \mathbb{R}^2 \times \mathbb{R}^2$, since the matrix $\boldsymbol{F}$ gives the relation between the derivatives of non-homogeneous vectors. Taking into account the equation (2), the relation (7) becomes

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \boldsymbol{F}\begin{bmatrix}\cos(\varphi)\\ \sin(\varphi)\end{bmatrix}\nu\,. \quad (8)$$

All the inputs have not yet appeared in the equation (8), so we need to continue with the differentiation. However, another differentiation of the equation (8) would differentiate the system input ν, so we need to introduce a new state $\xi = \nu$ before continuing. The second derivative is then:

$$\begin{bmatrix}\ddot{u}\\ \ddot{v}\end{bmatrix} = \dot{\boldsymbol{F}}\begin{bmatrix}\cos(\varphi)\\ \sin(\varphi)\end{bmatrix}\xi + \boldsymbol{F}\left(\begin{bmatrix}-\xi\sin(\varphi)\\ \xi\cos(\varphi)\end{bmatrix}\dot{\varphi} + \begin{bmatrix}\cos(\varphi)\\ \sin(\varphi)\end{bmatrix}\dot{\xi}\right)\,. \quad (9)$$

In the equation (9) the other system input (the angular velocity $\omega = \dot{\varphi}$) appeared, so the process of differentiation can be finished. The equation (9) can be rewritten into a short matrix form:

$$\begin{bmatrix}\ddot{u}\\ \ddot{v}\end{bmatrix} = \boldsymbol{\alpha} + \boldsymbol{T}\begin{bmatrix}\dot{\xi}\\ \omega\end{bmatrix} = \begin{bmatrix}u_1\\ u_2\end{bmatrix} = \boldsymbol{u}\,. \quad (10)$$

It can be shown that in a case where the tangential velocity differs from zero $\nu = \xi \neq 0$, and the matrix $\boldsymbol{F}$ is invertible, the matrix $\boldsymbol{T}$ is also invertible, so the dynamic compensator can be expressed from the equation (10) as follows:

$$\begin{bmatrix}\dot{\xi}\\ \omega\end{bmatrix} = \boldsymbol{T}^{-1}(\boldsymbol{u} - \boldsymbol{\alpha})\,, \quad \xi(0) = \nu_0\,, \quad \nu = \xi\,. \quad (11)$$

The elements of the matrix $\boldsymbol{F}$ depend on the world point $\boldsymbol{p}_w$, but since the world point can be expressed in terms of the image point $\boldsymbol{p}_p$ according to the equation (4), the matrix can be said to depend on the image point: $\boldsymbol{F} = \boldsymbol{F}(\boldsymbol{p}_p)$. Similarly, since the angle φ can be expressed in terms of the values in the image frame:

$$\tan\varphi = \frac{(h_{21} - vh_{31})\dot{u} - (h_{11} - uh_{31})\dot{v}}{(h_{12} - uh_{32})\dot{v} - (h_{22} - vh_{32})\dot{u}}\,, \quad (12)$$

all the terms on the right-hand side of the equation (11) can also be expressed in terms of the values in the image frame: $\boldsymbol{\alpha} = \boldsymbol{\alpha}(\boldsymbol{p}_p, \dot{\boldsymbol{p}}_p)$ and $\boldsymbol{T} = \boldsymbol{T}(\boldsymbol{p}_p, \dot{\boldsymbol{p}}_p, \xi)$. This is an important observation, because it enables us to estimate all the states, needed in the design of the control law, in the image frame directly.

Since the number of differentiations of all the outputs $(2+2)$ equals to the order of the original system plus the number of added integrators during the process of differentiation $(3+1)$, the overall extended system can be written in the following form:

$$\dot{\boldsymbol{x}}_i(t) = \boldsymbol{A}_c\boldsymbol{x}_i(t) + \boldsymbol{B}_c\boldsymbol{u}_i(t)\,, \quad \boldsymbol{y}_i(t) = \boldsymbol{C}_c\boldsymbol{x}_i(t)\,, \quad (13)$$

for $i \in \{1,2\}$, where $\boldsymbol{x}_1^T = [u\ \dot{u}]$, $\boldsymbol{x}_2^T = [v\ \dot{v}]$, $\boldsymbol{y}_1 = [u]$ and $\boldsymbol{y}_2 = [v]$, and the state-space matrices are:

$$\boldsymbol{A}_c = \begin{bmatrix}0&1\\0&0\end{bmatrix}, \quad \boldsymbol{B}_c = \begin{bmatrix}0\\1\end{bmatrix}, \quad \boldsymbol{C}_c = \begin{bmatrix}1&0\end{bmatrix}\,. \quad (14)$$

With respect to the new inputs $\boldsymbol{u}$ and flat outputs $\boldsymbol{p}_p$, the extended system is not only linear but also without input-output cross-coupling. The system is represented with two uni-variable subsystems, each subsystem consists of two integrators connected in series — the scheme known as *chain form*.

The feedback part of the controller is designed for each of the subsystems, indexed $i = \{1,2\}$ in the equation (13), separately. Since both subsystems in the equation (13) have the same form, we omit the writing of the subscript index i.

We are to develop the controller for the extended system as a sum of the feedforward and feedback actions $\boldsymbol{u} = \boldsymbol{u}_F + (-\boldsymbol{u}_B)$. We assume the reference signal has the same dynamics as the subsystem in the equation (13): $\dot{\boldsymbol{x}}_r = \boldsymbol{A}_c\boldsymbol{x}_r + \boldsymbol{B}_c\boldsymbol{u}_r$, which in our case means that the reference signal must be at least twice differentiable function. We select the feedforward control signal to be equal to the input signal of the reference model: $\boldsymbol{u}_F = \boldsymbol{u}_r$. The method of constructing such a reference signal is shown in section 4. Then, the model of the error $\boldsymbol{e} = \boldsymbol{y}_r - \boldsymbol{y}$ has the same form as the subsystem in the equation (13):

$$\dot{\boldsymbol{\epsilon}}(t) = \boldsymbol{A}_c\boldsymbol{\epsilon}(t) + \boldsymbol{B}_c\boldsymbol{u}_B(t)\,, \tag{15}$$

$$\boldsymbol{e}(t) = \boldsymbol{C}_c\boldsymbol{\epsilon}(t)\,, \tag{16}$$

where the state-error vector has been introduced: $\boldsymbol{\epsilon} = \boldsymbol{x}_r - \boldsymbol{x}$.

In the development of the error model predictive controller, the discrete equivalent of the error model (15) is needed. By expanding the step-invariant transformation equations into a Taylor series [35, p. 52], the following discrete error model with the sample time T_s is obtained:

$$\boldsymbol{\epsilon}(k+1) = \boldsymbol{A}\boldsymbol{\epsilon}(k) + \boldsymbol{B}\boldsymbol{u}_B(k)\,, \tag{17}$$

$$\boldsymbol{e}(k) = \boldsymbol{C}\boldsymbol{\epsilon}(k)\,, \tag{18}$$

where all the matrices are:

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}\,, \quad \boldsymbol{B} = \begin{bmatrix} \frac{T_s^2}{2} \\ T_s \end{bmatrix}\,, \quad \boldsymbol{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}\,. \tag{19}$$

3.2. Error model predictive control

The predictive control can be formulated as an optimization problem where we search for the control signal that minimizes some penalty function over finite prediction horizon $h > 0$ [28, 31]:

$$J(\boldsymbol{u}_B, k) = \sum_{i=0}^{h-1} \boldsymbol{\varepsilon}^T(k, i+1)\boldsymbol{Q}(i+1)\boldsymbol{\varepsilon}(k, i+1) + \\ + \boldsymbol{u}_B^T(k+i|k)\boldsymbol{R}(i)\boldsymbol{u}_B(k+i|k)\,, \tag{20}$$

where we have defined the states' error $\boldsymbol{\varepsilon}(k,i) = \boldsymbol{\epsilon}_r(k+i|k) - \boldsymbol{\epsilon}(k+i|k)$; and the matrices $\boldsymbol{Q}(i+1)$ and $\boldsymbol{R}(i)$ are positive-definite for $i = 0, 1, \ldots, h-1$.

At the current time step k an estimate of the error based on the model (17) at the future time moment $k+h$ can be obtained provided known current states and inputs until the future time $k+h-1$:

$$\boldsymbol{\epsilon}(k+h|k) = \boldsymbol{\Lambda}_h(k,0)\boldsymbol{\epsilon}(k|k) + \\ + \sum_{i=1}^{h-1} \boldsymbol{\Lambda}_h(k,i)\boldsymbol{B}(k+i-1|k)\boldsymbol{u}_B(k+i-1) + \\ + \boldsymbol{B}(k+h-1|k)\boldsymbol{u}_B(k+h-1)\,, \tag{21}$$

where $\boldsymbol{\Lambda}_h(k,i) = \boldsymbol{A}(k+h-1|k)\ldots\boldsymbol{A}(k+i+1|k)\boldsymbol{A}(k+i|k)$. All the states over the prediction horizon h can be described with an extended system:

$$\bar{\boldsymbol{\epsilon}}(k) = \bar{\boldsymbol{A}}(k)\boldsymbol{\epsilon}(k|k) + \bar{\boldsymbol{B}}(k)\bar{\boldsymbol{u}}_B(k)\,, \tag{22}$$

where the predicted states are gathered in the augmented vector $\bar{\boldsymbol{\epsilon}}(k) \in \mathbb{R}^{n \cdot h}$:

$$\bar{\boldsymbol{\epsilon}}^T(k) = \left[\boldsymbol{\epsilon}^T(k+1|k)\ \boldsymbol{\epsilon}^T(k+2|k)\ \ldots\ \boldsymbol{\epsilon}^T(k+h|k)\right]\,, \tag{23}$$

and the unknown control inputs in $\bar{\boldsymbol{u}}_B(k) \in \mathbb{R}^{m \cdot h}$:

$$\bar{\boldsymbol{u}}_B^T(k) = \left[\boldsymbol{u}_B^T(k|k)\ \boldsymbol{u}_B^T(k+1|k)\ \ldots\ \boldsymbol{u}_B^T(k+h-1|k)\right]\,. \tag{24}$$

The matrices $\bar{\boldsymbol{A}} \in \mathbb{R}^{n \cdot h} \times \mathbb{R}^n$ and $\bar{\boldsymbol{B}} \in \mathbb{R}^{n \cdot h} \times \mathbb{R}^{m \cdot h}$ are defined as follows:

$$\bar{\boldsymbol{A}}^T(k) = \left[\boldsymbol{A}^T(k|k)\ \boldsymbol{A}^T(k|k)\boldsymbol{A}^T(k+1|k)\ \ldots\ \boldsymbol{\Lambda}_h^T(k,0)\right]\,, \tag{25}$$

$$\bar{\boldsymbol{B}}(k) = \begin{bmatrix} \boldsymbol{B}(k|k) & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{A}(k+1|k)\boldsymbol{B}(k|k) & \boldsymbol{B}(k+1|k) & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Lambda}_h(k,1)\boldsymbol{B}(k|k) & \boldsymbol{\Lambda}_h(k,2)\boldsymbol{B}(k+1|k) & \cdots & \boldsymbol{B}(k+h-1|k) \end{bmatrix}\,. \tag{26}$$

Stacking the reference error states over the prediction horizon h into a vector

$$\bar{\boldsymbol{\epsilon}}_r^T(k) = \left[\boldsymbol{\epsilon}_r^T(k+1|k)\ \boldsymbol{\epsilon}_r^T(k+2|k)\ \ldots\ \boldsymbol{\epsilon}_r^T(k+h|k)\right]\,, \tag{27}$$

the penalty function (20) can be rewritten:

$$J(\bar{\boldsymbol{u}}_B, k) = \bar{\boldsymbol{\varepsilon}}^T(k)\bar{\boldsymbol{Q}}\bar{\boldsymbol{\varepsilon}}(k) + \bar{\boldsymbol{u}}_B^T(k)\bar{\boldsymbol{R}}\bar{\boldsymbol{u}}_B(k)\,, \tag{28}$$

where

$$\bar{\boldsymbol{\varepsilon}}(k) = \bar{\boldsymbol{\epsilon}}_r(k) - \bar{\boldsymbol{\epsilon}}(k)\,, \tag{29}$$

$$\bar{\boldsymbol{Q}} = \bigoplus_{i=1}^{h} \boldsymbol{Q}(i)\,, \qquad \bar{\boldsymbol{R}} = \bigoplus_{i=0}^{h-1} \boldsymbol{R}(i)\,. \tag{30}$$

The operator $\bigoplus$ denotes the direct sum. We have the ability to choose the reference error vector $\bar{\boldsymbol{\epsilon}}_r(k)$. One suggestion comes from [28], where the reference error is set to exponentially decrease according to the dynamics defined by the reference model $\boldsymbol{A}_r \in \mathbb{R}^n \times \mathbb{R}^n$:

$$\bar{\boldsymbol{\epsilon}}_r(k) = \left[\boldsymbol{A}_r^T\ \boldsymbol{A}_r^{2^T}\ \ldots\ \boldsymbol{A}_r^{h^T}\right]^T \boldsymbol{\epsilon}(k|k)\,. \tag{31}$$

Another option is to choose $\bar{\boldsymbol{\epsilon}}_r(k) = \boldsymbol{0}$, which is a special case of the equation (31) when $\boldsymbol{A}_r = \boldsymbol{0}$. This option demands from the controller that actual trajectory precisely follows the reference trajectory without filtering the error over the prediction horizon h.

With a search for the minimum of the penalty function $\frac{\partial J(\bar{\boldsymbol{u}}_B, k)}{\partial \bar{\boldsymbol{u}}_B(k)} = 0$ the optimum inputs to the system are obtained:

$$\bar{\boldsymbol{u}}_B(k) = \left(\bar{\boldsymbol{B}}^T(k)\bar{\boldsymbol{Q}}\bar{\boldsymbol{B}}(k) + \bar{\boldsymbol{R}}\right)^{-1} \bar{\boldsymbol{B}}^T(k)\bar{\boldsymbol{Q}}\left(\bar{\boldsymbol{\epsilon}}_r(k) - \bar{\boldsymbol{A}}(k)\boldsymbol{\epsilon}(k|k)\right)\,. \tag{32}$$

According to the receding horizon control strategy, at the time instant k only the first m inputs are applied to the system: $\boldsymbol{u}_B(k) = [\boldsymbol{I}\ \boldsymbol{0}\ \ldots\ \boldsymbol{0}]\bar{\boldsymbol{u}}_B(k)$, and at the next time step the whole procedure is repeated again. Because the control law was analytically derived, the need for the time-consuming minimization of the cost function (20) was eliminated, and thus the implementation in real-time systems is possible.

3.3. States estimation

We assume that the image recognition system can provide us with the information about the robot position given in the image frame, however, all the states required for the control are not directly measurable from the image. The unmeasurable states must be estimated, and because we have written the system in a linear form, a Kalman filter can be used. In visual servoing systems, delays are always present, so here we present a design of a Kalman filter that can take delays into account [36].

We assume the outputs of the system (18) are disturbed by a normally distributed white noise with zero mean and covariance matrix $\boldsymbol{N}$. The states of the system are assumed to be disturbed by a white noise with zero mean and covariance matrix $\boldsymbol{V}$, filtered by $\boldsymbol{F}$. Let the delay be equal to $d > 0$ sample times. In the current moment k, a new measurement of the error $\boldsymbol{e}(k_d)$ is obtained, which was valid at the delayed sample time $k_d = k - d$. The correction of the posterior $\hat{\boldsymbol{\epsilon}}(k_d|k)$ with covariance $\hat{\boldsymbol{P}}(k_d|k)$ can be made from the estimation $\boldsymbol{\epsilon}(k_d|k-1)$ with covariance $\boldsymbol{P}(k_d|k-1)$:

$$\hat{\boldsymbol{\epsilon}}(k_d|k) = \boldsymbol{\epsilon}(k_d|k-1) + \boldsymbol{L}(k_d|k)(\boldsymbol{e}(k_d) - \boldsymbol{C}(k_d|k)\boldsymbol{\epsilon}(k_d|k-1))\,, \tag{33}$$

$$\hat{\boldsymbol{P}}(k_d|k) = \boldsymbol{P}(k_d|k-1) - \boldsymbol{L}(k_d|k)\boldsymbol{C}(k_d|k)\boldsymbol{P}(k_d|k-1)\,, \tag{34}$$

$$\boldsymbol{L}(k_d|k) = \boldsymbol{P}(k_d|k-1)\boldsymbol{C}^T(k_d|k)(\boldsymbol{C}(k_d|k)\boldsymbol{P}(k_d|k)\boldsymbol{C}(k_d|k) + \boldsymbol{N}(k_d|k))^{-1}\,. \tag{35}$$

According to the error model (18), the predictions from the delayed step to the current step can be made:

$$\boldsymbol{\epsilon}(k|k) = \hat{\boldsymbol{A}}(k)\hat{\boldsymbol{\epsilon}}(k_d|k) + \hat{\boldsymbol{B}}(k)\hat{\boldsymbol{u}}_B(k)\,, \tag{36}$$

where we have gathered the last d control signals into augmented vector $\hat{\boldsymbol{u}}_B^T(k) = [\boldsymbol{u}_B^T(k_d)\ \boldsymbol{u}_B^T(k_d+1)\ \dots\ \boldsymbol{u}_B^T(k-1)]$. The matrices $\hat{\boldsymbol{A}} \in \mathbb{R}^n \times \mathbb{R}^n$ and $\hat{\boldsymbol{B}} \in \mathbb{R}^n \times \mathbb{R}^{m\cdot d}$ are defined as follows:

$$\hat{\boldsymbol{A}}(k) = \boldsymbol{\Lambda}_d(k,0)\,, \tag{37}$$

$$\hat{\boldsymbol{B}}(k) = [\boldsymbol{\Lambda}_d(k,1)\boldsymbol{B}(k_d|k)\ \dots\ \boldsymbol{\Lambda}_d(k,d-1)\boldsymbol{B}(k-2|k)\ \boldsymbol{B}(k-1|k)]\,, \tag{38}$$

where $\boldsymbol{\Lambda}_d(k,i) = \boldsymbol{A}(k-1|k)\dots\boldsymbol{A}(k_d+i+1|k)\boldsymbol{A}(k_d+i|k)\ \forall\ i < d$. Before we continue to the next step, we estimate the error in the next delayed step:

$$\boldsymbol{\epsilon}(k_d+1|k) = \boldsymbol{A}(k_d|k)\hat{\boldsymbol{\epsilon}}(k_d|k) + \boldsymbol{B}(k_d|k)\boldsymbol{u}_B(k_d)\,, \tag{39}$$

$$\boldsymbol{P}(k_d+1|k) = \boldsymbol{A}(k_d|k)\hat{\boldsymbol{P}}(k_d|k)\boldsymbol{A}^T(k_d|k) + \boldsymbol{F}\boldsymbol{V}(k_d|k)\boldsymbol{F}^T\,. \tag{40}$$

When the Kalman filter is used, it is important that we have accurate estimation of the covariance matrices, otherwise the Kalman filter may return state estimations that are either overconfident or too pessimistic [37].

In multitasking and/or distributed systems it may not be easy to establish sampling time with the constant period which can lead to large errors when we use the equation (36). If the model (18) is known for time-dependent sampling times, the problem can be solved by measuring the time between the samples [13, 36].

4. Trajectory generation

In the development of the control law in section 3.1 we have assumed the reference signal comes from the space of twice differentiable functions. One way of determining the reference trajectory is in the form of a parametric curve (dependent on time) [27, 31, 32]. A convenient way of constructing a reference trajectory is with the use of the Bernstein-Bézier (BB) parametric curves [31]. The BB curves are completely determined with a set of control points, and the number of the control points determine the order of the BB curve. High-order BB curves can be numerically unstable, but it usually suffices to use just low-order curves, since an approximation of an arbitrary curve can be achieved with gluing together more low-order BB curves to obtain a BB spline.

A general D-dimensional BB curve $\boldsymbol{r}^T = [r_1\ r_2\ \dots\ r_D]$ of order $b \in \mathbb{N}$ in parametric form is defined as follows:

$$\boldsymbol{r}(\lambda) = \sum_{i=0}^{b} B_{i,b}(\lambda)\boldsymbol{p}_i\,, \quad B_{i,b}(\lambda) = \binom{b}{i}\lambda^i(1-\lambda)^{b-i}\,, \tag{41}$$

where the parameter λ represents the normalized time which takes the real values from the interval $[0,\ 1]$ and it is related to the relative time t with the linear equation $t = \lambda\Lambda$, where Λ is the time it takes to reach from the beginning ($\lambda = 0$) to the end ($\lambda = 1$) of the curve. The BB curve (41) is of class $\mathcal{C}^\infty$ (smooth curve), and it is completely defined by a set of control points $\{\boldsymbol{p}_i^T = [p_{1,i}\ p_{2,i}\ \dots\ p_{D,i}]\}_{i=0,1,\dots,b}$ that form a Bézier polygon. The BB curve is always bounded by the convex envelope of the Bézier polygon. The derivative of the BB curve with respect to relative time t of order $d \geq 0$ is as follows:

$$\frac{\partial^d \boldsymbol{r}(t)}{\partial t^d} = \frac{1}{\Lambda^d}\frac{\partial^d \boldsymbol{r}(\lambda)}{\partial \lambda^d}\,. \tag{42}$$

4.1. Bernstein-Bézier curve at end points

The BB curve (41) always begins ($\lambda = 0$) at the first control point $\boldsymbol{p}_0$ and ends ($\lambda = 1$) at the last control point $\boldsymbol{p}_b$; the so-called end point interpolation property. All the other control points in general do not lie on the BB curve. Next, we show how the other control points influence the behaviour of the curve at the end points. At the end points, the following limits can be derived for $d \geq 0$, $b \geq d$:

$$\lim_{\lambda\leftarrow 0}\frac{\partial^d \boldsymbol{r}(\lambda)}{\partial \lambda^d} = \binom{b}{d}d!\sum_{i=0}^{d}(-1)^i\binom{d}{i}\boldsymbol{p}_{d-i}\,, \tag{43}$$

$$\lim_{\lambda\to 1}\frac{\partial^d \boldsymbol{r}(\lambda)}{\partial \lambda^d} = \binom{b}{d}d!\sum_{i=0}^{d}(-1)^i\binom{d}{i}\boldsymbol{p}_{b-i}\,. \tag{44}$$

The derivative of order d at the end points is dependent only on the d first or last control points, respectively.

Let us confine ourselves to a two-dimensional space ($\boldsymbol{r}^T = [r_x\ r_y]$, $\boldsymbol{p}_i^T = [p_{x,i}\ p_{y,i}]$). In this case the orientation of the tangent on the curve is determined by the equation:

$$\varphi(t) = \arctan\frac{\dot{r}_y(t)}{\dot{r}_x(t)} + C\pi\,, \quad C \in \mathbb{Z}\,. \tag{45}$$

The tangential velocity along the curve is calculated as follows:

$$v(t) = \sqrt{\dot{r}_x^2(t) + \dot{r}_y^2(t)}\,, \tag{46}$$

and from the differentiation of the (45), the angular velocity along the curve is obtained:

$$\omega(t) = \frac{\dot{r}_x(t)\ddot{r}_y(t) - \ddot{r}_x(t)\dot{r}_y(t)}{\dot{r}_x^2(t) + \dot{r}_y^2(t)}\,. \tag{47}$$

We consider only the relations at the starting point of the BB curve (41) ($\lambda = 0$), since the relations at the end point can be obtained in the same way. Taking into account the equations (42) and (43), the tangential velocity (46) at the starting point of the BB curve can be expressed in terms of the first two control points:

$$v_0 = \lim_{t\leftarrow 0} v(t) = \frac{b}{\Lambda}\sqrt{(p_{x,1} - p_{x,0})^2 + (p_{y,1} - p_{y,0})^2}\,. \tag{48}$$

We assume the tangential velocity never reaches zero $v(t) \neq 0$. The orientation (45) at the starting point is then given as:

$$\varphi_0 = \lim_{t \leftarrow 0} \varphi(t) = \arctan \frac{p_{y,1} - p_{y,0}}{p_{x,1} - p_{x,0}} + C\pi\,, \quad C \in \mathbb{Z}\,, \tag{49}$$

and the angular velocity (47) at the starting point as:

$$\omega_0 = \lim_{t \leftarrow 0} \omega(t) = \frac{b-1}{\Lambda} \frac{(p_{x,1}-p_{x,0})p_{y,2}-(p_{y,1}-p_{y,0})p_{x,2}+p_{x,0}p_{y,1}-p_{x,1}p_{y,0}}{(p_{x,1}-p_{x,0})^2+(p_{y,1}-p_{y,0})^2}\,. \tag{50}$$

The relation (50) is the equation of a line that is parallel to the line that passes through the control points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$. When the point $\boldsymbol{p}_2$ lies on the line passing through the points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$, the angular velocity ω_0 becomes zero. It can be shown that the lines perpendicular to the line that is passing through the points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ represent the lines of constant tangential acceleration $\dot{v}$ at the starting point as seen from another equation of a line:

$$a_0 = \lim_{t \leftarrow 0} \dot{v}(t) = \frac{b-2}{\Lambda^2} \frac{(p_{x,2}-2p_{x,1}+p_{x,0})(p_{x,1}-p_{x,0})+(p_{y,2}-2p_{y,1}+p_{y,0})(p_{y,1}-p_{y,0})}{(p_{x,1}-p_{x,0})^2+(p_{y,1}-p_{y,0})^2}\,. \tag{51}$$

When the following condition is met $\boldsymbol{p}_2 = 2\boldsymbol{p}_1 - \boldsymbol{p}_0$, the angular velocity ω_0 and tangential acceleration a_0 are both zero.

The equations (48) to (51) can be expressed more conveniently in terms of the wanted properties at the starting point of the curve:

$$\boldsymbol{p}_0 = \begin{bmatrix} p_{x,0} & p_{y,0} \end{bmatrix}^T\,, \tag{52}$$

$$\boldsymbol{p}_1 = \frac{\Lambda v_0}{b} \begin{bmatrix} \cos\varphi_0 & \sin\varphi_0 \end{bmatrix}^T + \boldsymbol{p}_0\,, \tag{53}$$

$$\boldsymbol{p}_2 = \frac{\Lambda^2}{b(b-1)} \begin{bmatrix} \cos\varphi_0 & -v_0\sin\varphi_0 \\ \sin\varphi_0 & v_0\cos\varphi_0 \end{bmatrix} \begin{bmatrix} \dot{v}_0 \\ \omega_0 \end{bmatrix} + 2\boldsymbol{p}_1 - \boldsymbol{p}_0\,. \tag{54}$$

The minimum order of the BB curve required to design a curve with the desired values of the properties (48) to (51) at the beginning and end of the curve, independently, is five. In the case of a higher-order curve, the additional central control points can be used to change the shape of the curve without changing the desired properties at the curve boundaries. For example, these additional control points can be positioned in a way that achieves an obstacle-free path [31].

To summarize, the first control point defines the curve starting position, the first two control points define the orientation φ_0 and tangential velocity v_0, and the first three control points define the angular velocity ω_0 and tangential acceleration a_0. Hence, the first three control points $(\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2)$ can be positioned by specifying the starting position, orientation, tangential and angular velocity, and tangential acceleration. In the same way the last three control points $(\boldsymbol{p}_b, \boldsymbol{p}_{b-1}, \boldsymbol{p}_{b-2})$ define the curve at the end point.

4.2. Bernstein-Bézier spline

To define an arbitrary curve one could increase the order of the BB curve, but this may introduce numerical instability. A better approach for defining an arbitrary path is to compose it from a set of connected low-order curves. This also allows us to define additional properties at the joint points as shown in section 4.1. The connections should be carried out in way that achieves continuity of the curve and its derivatives up to some order. We consider three cases for joining two consecutive BB curves, demanding the continuity of the curve, continuity of the first order derivative and also continuity of the second order derivative. We describe how all these demands are related to the so-called tangential and angular velocities of the curve at the junctions. We denote two consecutive curves with $\boldsymbol{r}_j(\lambda)$ and $\boldsymbol{r}_{j+1}(\lambda)$, respectively. Here we allow the relative time to be dependent on the curve part $t_j = \Lambda_j\lambda$ which gives us an additional degree of freedom. We also assume that the relative time t is reset to zero at the beginning of each BB curve.

To achieve continuity of a spline the following condition has to be met:

$$\lim_{\lambda \to 1} \boldsymbol{r}_j(\lambda) = \lim_{\lambda \leftarrow 0} \boldsymbol{r}_{j+1}(\lambda)\,, \tag{55}$$

which yields the condition for selecting the first control point in the next curve part:

$$\boldsymbol{p}_{0,j+1} = \boldsymbol{p}_{b,j}\,. \tag{56}$$

This means that the curve is continuous, but the orientation of the tangent coming into the junction may or may not be the same as the one leaving the junction, $\varphi_{b,j} \neq \varphi_{0,j+1}$. The same fact also applies to the tangential velocity $v_{b,j} \neq v_{0,j+1}$, the angular velocity $\omega_{b,j} \neq \omega_{0,j+1}$, and the tangential acceleration $a_{b,j} \neq a_{0,j+1}$.

To achieve continuity of the BB spline up to the first order in addition to the condition (55) the following condition has to be met:

$$\frac{1}{\Lambda_j} \lim_{\lambda \to 1} \frac{\partial \boldsymbol{r}_j(\lambda)}{\partial\lambda} = \frac{1}{\Lambda_{j+1}} \lim_{\lambda \leftarrow 0} \frac{\partial \boldsymbol{r}_{j+1}(\lambda)}{\partial\lambda}\,, \tag{57}$$

which yields an additional condition for selecting the second control point in the next curve part:

$$\boldsymbol{p}_{1,j+1} = \left(\frac{\Lambda_{j+1}}{\Lambda_j} + 1\right)\boldsymbol{p}_{b,j} - \frac{\Lambda_{j+1}}{\Lambda_j}\boldsymbol{p}_{b-1,j}\,. \tag{58}$$

Assume that the tangential velocity at the end point of the previous curve is not zero, $v_{b,j} \neq 0$. Then, in addition to the continuous position, the orientation and tangential velocity are also continuous at the junction point, $\varphi_{b,j} = \varphi_{0,j+1}$ and $v_{b,j} = v_{0,j+1}$; but the angular velocity and tangential acceleration may still not be continuous at the junction point.

To achieve continuity of a spline up to the second order in addition to the conditions (59) and (57) the following condition has to be met:

$$\frac{1}{\Lambda_j^2} \lim_{\lambda \to 1} \frac{\partial^2 \boldsymbol{r}_j(\lambda)}{\partial\lambda^2} = \frac{1}{\Lambda_{j+1}^2} \lim_{\lambda \leftarrow 0} \frac{\partial^2 \boldsymbol{r}_{j+1}(\lambda)}{\partial\lambda^2}\,, \tag{59}$$

which yields an additional condition for selecting the third control point in the next curve part:

$$\boldsymbol{p}_{2,j+1} = \left(\frac{\Lambda_{j+1}}{\Lambda_j} + 1\right)^2 \boldsymbol{p}_{b,j} - 2\frac{\Lambda_{j+1}}{\Lambda_j}\left(\frac{\Lambda_{j+1}}{\Lambda_j} + 1\right)\boldsymbol{p}_{b-1,j} + \frac{\Lambda_{j+1}^2}{\Lambda_j^2}\boldsymbol{p}_{b-2,j}\,. \tag{60}$$

In this case, in addition to the continuity of the position, orientation and tangential velocity, the angular velocity and tangential acceleration are also continuous at the junction point, $\omega_{b,j} = \omega_{0,j+1}$ and $a_{b,j} = a_{0,j+1}$.

4.3. Bernstein-Bézier curve under projective transformation

Since the perspective transformation of homography (4) is a non-linear transformation, the BB curve (41) cannot be transformed through homography from the world to image frame, or vice versa, in terms of the control points. However, the curve and also the curve's derivatives can be transformed from one frame to another, and the homography transformation does not break apart the continuity of the curve to the specific order, since the continuity and smoothness are properties of the curve that are invariant under homography transformation.

Since the Bernstein basis polynomials $B_{i,b}(\lambda)$ form a partition of unity: $\sum_{i=0}^{b} B_{i,b}(\lambda) = 1$, the equation (41) does not change even if the points are defined in homogeneous coordinates. Applying the homography transformation (4) to the two-dimensional BB curve in the world frame $\boldsymbol{r}_w^T(\lambda) = [r_x(\lambda)\ r_y(\lambda)\ 1]$, defined with the control points $\boldsymbol{p}_i^T = [p_{x,i}\ p_{y,i}\ 1]$, the BB curve in the image frame $\boldsymbol{r}_p^T(\lambda) = [r_u(\lambda)\ r_v(\lambda)\ 1]$ is as follows:

$$\begin{aligned} r_u(\lambda) &= \frac{\sum_{i=0}^{b} B_{i,b}(\lambda)\boldsymbol{h}_1^T\boldsymbol{p}_i}{\sum_{i=0}^{b} B_{i,b}(\lambda)\boldsymbol{h}_3^T\boldsymbol{p}_i}, \\ r_v(\lambda) &= \frac{\sum_{i=0}^{b} B_{i,b}(\lambda)\boldsymbol{h}_2^T\boldsymbol{p}_i}{\sum_{i=0}^{b} B_{i,b}(\lambda)\boldsymbol{h}_3^T\boldsymbol{p}_i}. \end{aligned} \tag{61}$$

It is clear that the rational expression (61) cannot be written in a polynomial form like (41), so the transformed BB curve (rational BB curve) cannot be written in terms of the control points in the transformed frame.

Two distinctive approaches can be used in the design of the reference trajectory. The reference trajectory can be designed in terms of the control points in the world plane, and then transformed into the image frame according to the equation (61). This approach requires knowledge of the homography matrix $\boldsymbol{H}$, and if the homography matrix is not exactly known, there will be an inherent error that the controller will not be able to eliminate, since the control error is defined in the image space. An alternative is to define the trajectory in the image frame directly. The equations in section 4.1 and section 4.2 stay exactly the same, except that they are all defined in the image frame, and therefore all the values are expressed in pixels instead of metres. This is the preferred way, since the control error is also defined in the image space, and therefore there is no inherent error. In the previous approach the trajectory would need to be designed in the world plane, then transformed into the image plane and then inspected in the image frame to see if the transformation is accurate enough for the given task. In other words, no matter which approach is selected for generation of the trajectory, the trajectory should always be inspected in image space, but in the case where the trajectory is designed in the image space, this step is done inherently.

5. Experiments

We first tested the presented control algorithm in a simulation environment. The camera was positioned to some arbitrary location in space, in a way that the ground plane was in the camera's field of view. We defined the reference trajectory with the fifth order BB spline, and demanded from the trajectory to have continuous derivatives up to the second order — which means that the tangential and angular velocity are both continuous functions. We selected the time length of all the BB curves in the spline to be $\Lambda = 5\ s$.

To the outputs of the system a normally distributed white noise with zero mean and a variance $\sigma_n = 2\,\text{px}^2$ was added. The sampling time was selected to be $T_s = 0.1$ s. The system was supposed to have a delay of two samples. The initial tangential velocity in the dynamic feedback compensator was set to the tangential velocity at the beginning of the reference trajectory in the world space. The error model predictive controller was initialized with the following values:

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}, \quad \boldsymbol{R} = [0.001], \quad \boldsymbol{A}_r = \begin{bmatrix} 0.65 & 0 \\ 0 & 0.65 \end{bmatrix}. \tag{62}$$

The Kalman filter was initialized with the following values of the covariance matrices:

$$\boldsymbol{N} = [3], \quad \boldsymbol{V} = \boldsymbol{P} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.5 \end{bmatrix}. \tag{63}$$

The robot was displaced from the starting location of the reference trajectory (non-zero initial condition).

With the selected parameters, we made several experiments. For the control performance assessment, we defined several performance evaluation functions:

$$\begin{aligned} SSE &= \sqrt{\sum_{k=1}^{K} e_u^2(k) + e_v^2(k)}, \\ SAE &= \sum_{k=1}^{K} \sqrt{e_u^2(k) + e_v^2(k)}, \end{aligned} \tag{64}$$

$$\begin{aligned} SSdU_c &= \sqrt{\sum_{k=2}^{K} (c(k) - c(k-1))^2}, \\ SAdU_c &= \sum_{k=2}^{K} |c(k) - c(k-1)|, \end{aligned} \tag{65}$$

where $e_u(k) = r_u(k) - u(k)$ and $e_v(k) = r_v(k) - v(k)$, and c can be either v or ω. The criterion functions SSE and SAE evaluate the trajectory-tracking error, while the cost functions $SSdU_c$ and $SAdU_c$ evaluate the control effort.

Since the control algorithm demands knowledge of the homography (the mapping between the points in the image and world plane), we first evaluated the robustness of the proposed algorithm to an imprecise estimation of the homography matrix. The bold quadrilateral in Figure 3(b) shows the boundaries of the plane that are visible in the image, and the other ten thin quadrilaterals represent the estimated visible area in the image — each quadrilateral represents one homography. The performance of the tracking algorithm for all the inaccurately estimated homography matrices is depicted in the other three figures in Figure 3, and in table Table 1 some statistics about the control performance for all the cases is gathered.

	SSE px	SAE px	$SSdU_v$ m s^{-1}	$SAdU_v$ m s^{-1}	$SSdU_\omega$ s^{-1}	$SAdU_\omega$ s^{-1}
Min	129.1	1094	0.1203	1.671	10.16	111.6
Mean	151.5	1505	0.183	2.085	11.87	120.2
Max	200.4	2519	0.3206	2.954	16.14	131.7

Table 1. Evaluation of the trajectory-tracking performance with respect to the imprecise estimations of the homography.

The model predictive control algorithm can be tuned in terms of the prediction horizon. To evaluate the influence of the prediction horizon h on the control quality, we considered cases where the prediction horizon takes values in the range from one to ten. The trajectory-tracking performance is evaluated in Table 2, and also shown in Figure 4 for the length of the prediction horizon one and five. The measurements were repeated twenty times. In this case we assumed that the homography matrix is known precisely.

h	SSE px	SAE px	$SSdU_v$ m s^{-1}	$SAdU_v$ m s^{-1}	$SSdU_\omega$ s^{-1}	$SAdU_\omega$ s^{-1}
1	**270.1**	**2439**	0.094 05	1.344	4.641	37.36
2	129.5	1049	0.1724	2.042	**14.34**	**148.3**
3	179.4	1356	0.1054	1.493	5.644	57.62
4	147.2	1093	0.1245	1.66	7.746	83.08
5	132.4	988.7	0.1416	1.783	9.807	104.9
6	127.9	992.7	0.1566	1.902	11.88	122.8
7	127.8	1000	0.1644	1.975	12.74	132.9
8	129.3	1039	0.1685	2.028	13.85	143.5
9	130.2	1066	0.1721	2.047	14.01	146.3
10	130.4	1076	**0.1752**	**2.058**	14.07	147.5

Table 2. Evaluation of trajectory-tracking performance with respect to different lengths of the prediction horizon h.

In all the experiments considered the output of the system was corrupted by white Gaussian noise of variance $\sigma_n^2 = 2\,\text{px}^2$. Furthermore, we also evaluated the performance due to different levels of noise, and the results are gathered in Table 3. We again assumed that the homography matrix is precisely known. Figure 5 shows reference-tracking in a case where the level of the noise is $\sigma_n^2 = 3\,\text{px}^2$, and this figure can be compared to Figure 4(c), Figure 4(d) where the value of noise is $\sigma_n^2 = 2\,\text{px}^2$, but other simulation conditions are the same.

To test the performance of the tracking controller on the real system, we implemented a simple image-tracking algorithm. To simplify the image-based robot tracking, we equipped a robot with a colour marker that can be recognized easily by the image recognition system [7]. The position of the robot is supposed to be at the centre of the largest detected patch. The recognized robot in the image is shown in Figure 6(a). In Figure 6(b) the results of the trajectory-tracking on the real system are presented. To improve readability, only every fifth sample in Figure 6(b) is marked with a symbol, and all the samples are connected with a line.

σ^2 px^2	SSE px	SAE px	$SSdU_v$ m s^{-1}	$SAdU_v$ m s^{-1}	$SSdU_\omega$ s^{-1}	$SAdU_\omega$ s^{-1}
0.0	123.8	769.8	0.1339	1.562	6.109	37.81
0.5	124.1	778.5	0.1353	1.583	6.629	50.78
1.0	124.7	824.3	0.1398	1.649	8.056	74.76
1.5	125.9	896	0.1472	1.759	9.807	98.07
2.0	127.7	962.6	0.1527	1.839	11.15	114.7
2.5	133	1137	0.1663	2.071	14.48	150.7
3.0	**137.6**	**1268**	**0.1771**	**2.223**	**16.08**	**174.6**

Table 3. Evaluation of trajectory-tracking performance with respect to different levels of noise.

6. Conclusion

We presented the design of a visual controller for trajectory-tracking of a mobile robot that is observed by an overhead camera that can be placed at arbitrary inclination with respect to the ground plane. An image-based visual servoing principle was used in the design of the control algorithm. The presented algorithm includes a Kalman filter for state estimation and a model predictive control for reference-tracking. Since the proposed control algorithm requires a special trajectory that is at least twice differentiable, we gave an extensive description for the use of Bernstein-Bézier curves to tackle the trajectory design problem. The presented control algorithm was designed in the discrete-time space.

The control algorithm was derived assuming the homography $\boldsymbol{H}$ is precisely known, but actually it can only be estimated with finite precision in the process of calibration. Nevertheless, the experimental results in Figure 3 and Table 1 show that the control algorithm is robust to imprecise estimation of the homography.

Another important parameter of the control algorithm is the length of the model predictive control horizon h. The evaluation of reference tracking performance with respect to the length of the prediction horizon (Figure 4 and Table 2) shows that good tracking performance is not achieved when the control horizon is only of unit length. When we increase the length of the control horizon the tracking performance improves, but too long a control horizon may smooth the trajectory sharp turns more than is desirable. The longer control horizon also increases the control high-frequency gain (Figure 4(d)). In our case, the control horizon of length five seems to give good trade off between tracking performance and the control gain (Table 2).

The measured position of the robot in the image, obtained by the image-tracking algorithm, is inherently corrupted with noise. The experimental results in Figure 5 and Table 3 show the influence of the noisy measurements on the trajectory-tracking performance, and confirm that the controller is robust to noise.

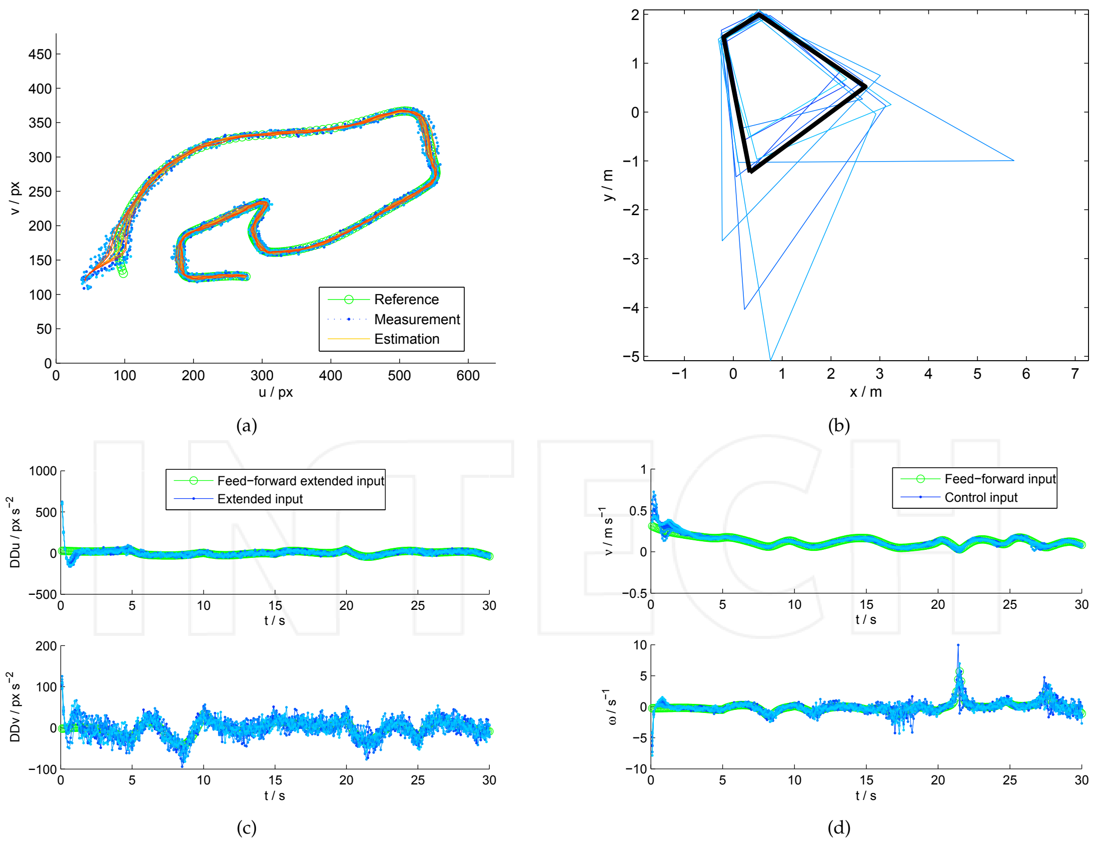


Figure 3. Trajectory-tracking (a) with respect to the imprecise estimation of the homography; (b) boundaries of the plane that are visible in the image for different estimations of the homography (quadrilateral with strong edges corresponds to the true homography); (c) extended inputs and (d) control inputs. In all the figures all ten signals are overlaid.

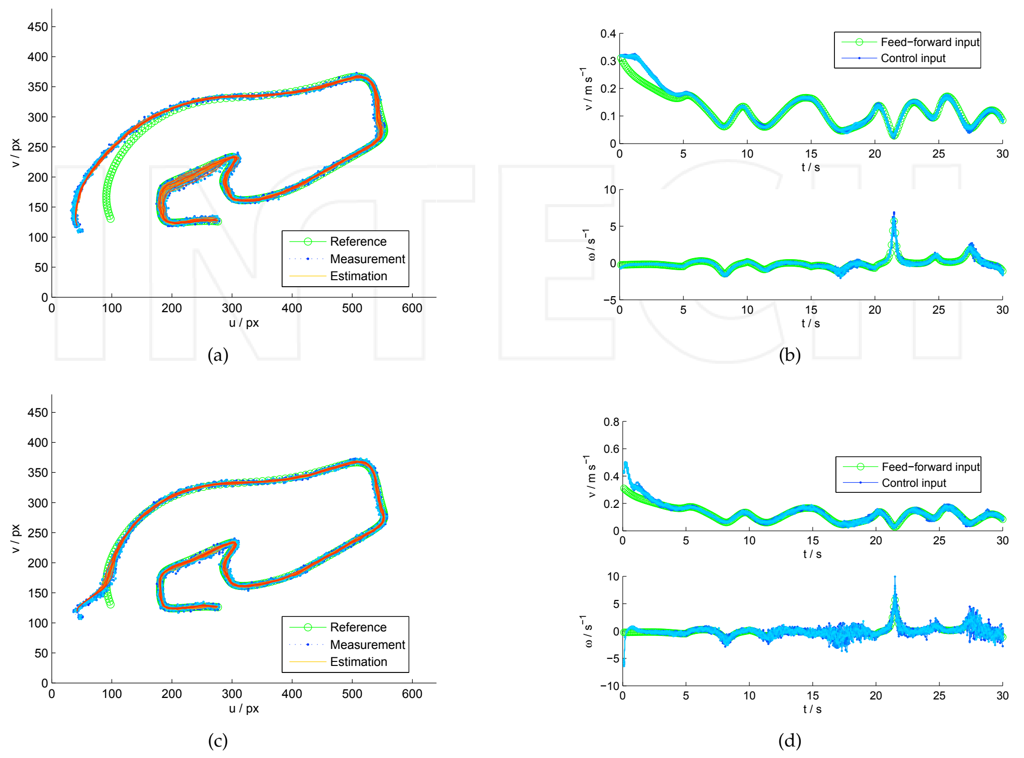
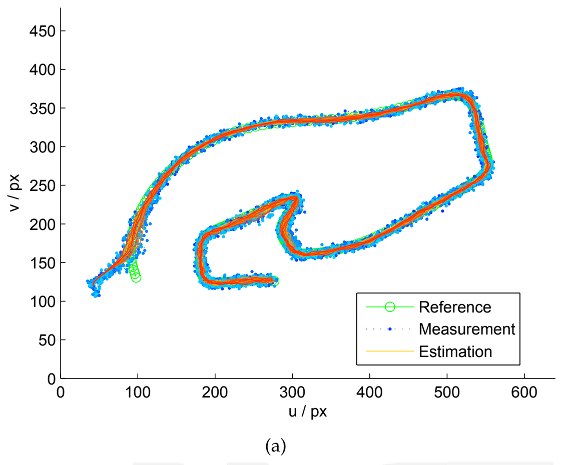


Figure 4. Trajectory-tracking (a), (c) for the length of the prediction horizon $h = 1$ (top row) and $h = 5$ (bottom row); (b), (d) control inputs. In all the figures all the signals from 20 repeats of the experiment are overlaid.



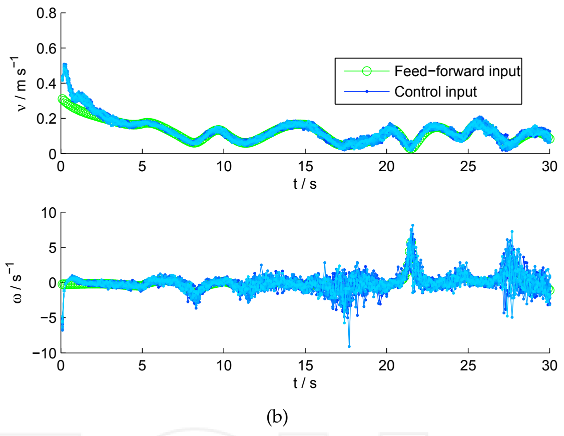
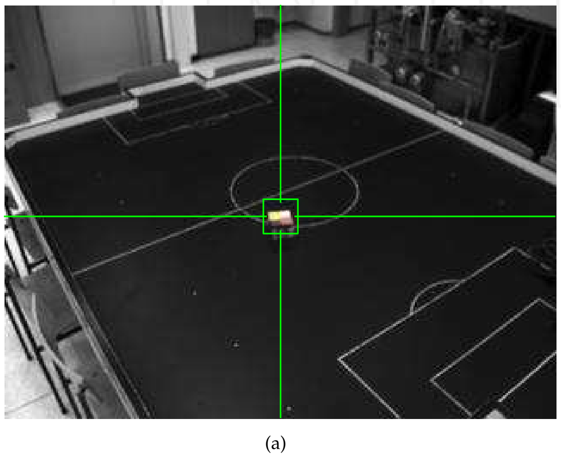


Figure 5. Trajectory-tracking (a) with respect to different levels of noise; (b) control inputs. In all the figures all the signals from 20 repeats of the experiment are overlaid.



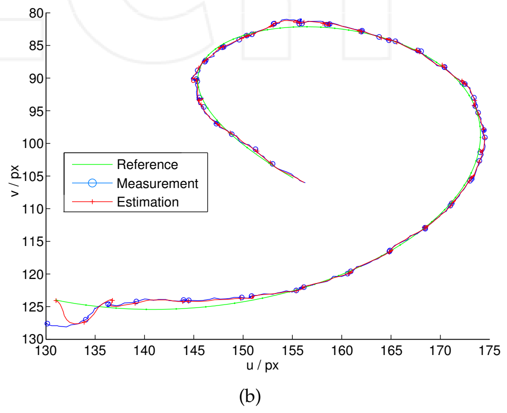


Figure 6. Trajectory-tracking on a real robot: (a) view from the camera with the detected robot and (b) trajectory-tracking results in the image frame.

The simulation experiments as well as experiments made on the real system (Figure 6) confirm that the presented control approach is suitable for solving the trajectory-tracking task. The results prove the applicability of the presented control design, even in the case of non-ideal conditions.

7. References

[1] Peter Corke. Visual Control of Robot Manipulators — A Review. In *Visual Servoing*, pages 1–31. 1993.

[2] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011.

[3] Francois Chaumette and Seth Hutchinson. Visual Servo Control, part I: Basic Approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, 2006.

[4] Francois Chaumette and Seth Hutchinson. Visual Servo Control, part II: Advanced Approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118, 2007.

[5] Danica Kragic and Henrik I. Christensen. Survey on Visual Servoing for Manipulation. Technical report, Centre for Autonomous Systems, Numerical Analysis and Computer Science, Stockholm, Sweden, 2002.

[6] Ambrose Chan. *Constraint-Aware Visual Servoing for Teaching Practical Robot Motion*. PhD thesis, The University Of British Columbia, 2009.

[7] Gregor Klančar, Matej Kristan, Stanislav Kovačič, and Omar Orqueda. Robust and efficient vision system for group of cooperating mobile robots with application to soccer robots. *ISA transactions*, 43(3):329–342, July 2004.

[8] Alessandro De Luca, Giuseppe Oriolo, and Paolo Robuffo Giordano. Feature Depth Observation for Image-based Visual Servoing: Theory and Experiments. *The International Journal of Robotics Research*, 27(10):1093–1116, October 2008.

[9] Felix von Hundelshausen. *Computer Vision for Autonomous Mobile Robots*. PhD thesis, Freien Universität Berlin, 2004.

[10] Young-Ho Kim, Sang-Wook Lee, Hyun S. Yang, and Dylan A. Shell. Toward autonomous robotic containment booms: visual servoing for robust inter-vehicle docking of surface vehicles. *Intelligent Service Robotics*, October 2011.

[11] Marci Meingast, Christopher Geyer, and Shankar Sastry. Vision Based Terrain Recovery for Landing Unmanned Aerial Vehicles. In *43rd IEEE Conference on Decision and Control*, pages 1670–1675, 2004.

[12] Daniel Eberli, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Vision Based Position Control for MAVs Using One Single Circular Landmark. *Journal of Intelligent and Robotic Systems*, 61(1-4):495–512, November 2010.

[13] Matevž Bošnak, Drago Matko, and Sašo Blažič. Quadrocopter Hovering Using Position-estimation Information from Inertial Sensors and a High-delay Video System. *Journal of Intelligent and Robotic Systems*, December 2011.

[14] Matevž Bošnak, Drago Matko, and Sašo Blažič. Quadrocopter control using an on-board video system with off-board processing. *Robotics and Autonomous Systems*, 60(4):1–40, April 2012.

[15] Omid Shakernia, Yi Ma, T. John Koo, Joao Hespanha, and S. Shankar Sastry. Vision Guided Landing of an Unmanned Air Vehicle. In *Proceedings of the 38th Conference on Decision and Control*, pages 1–6, 1999.

[16] Odile Bourquardez and Francois Chaumette. Visual Servoing of an Airplane for Alignment with respect to a Runway. In *IEEE International Conference on Robotics and Automation*, volume 7, pages 1330–1335, 2007.

[17] Satja Lumbar, Gregor Dolanc, Darko Vrečko, Stanko Strmčnik, and Drago Matko. Automatic Guidance of an aircraft using predictive control in a visual servoing scheme. In *18th IFAC Symposium on Automatic Control in Aerospace ACA*, page 6, Nara, Japan, 2010.

[18] Gregor Klančar, Sašo Blažič, Drago Matko, and Gašper Mušič. Image-Based Attitude Control of a Remote Sensing Satellite. *Journal of Intelligent and Robotic Systems*, 66(3):343–357, August 2011.

[19] Guan-Lu Zhang. *Image-based Visual Servoing with Hybrid Camera Configuration for Robust Robotic Grasping*. PhD thesis, University of British Columbia, 2009.

[20] Dimitrios Kosmopoulos. Robust Jacobian matrix estimation for image-based visual servoing. *Robotics and Computer-Integrated Manufacturing*, 27(1):82–87, February 2011.

[21] Rafael Kelly, Ricardo Carelli, Oscar Nasisi, Benjamín Kuchen, and Fernando Reyes. Stable Visual Servoing of Camera-in-Hand Robotic Systems. *IEEE/ASME Transactions on Mechatronics*, 5(1):39–48, 2000.

[22] Ezio Malis, G. Chesi, and R. Cipolla. 2 1/2 D Visual Servoing with Respect to Planar Contours having Complex and Unknown Shapes. *The International Journal of Robotics Research*, 22(10):841–853, 2003.

[23] Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. Technical report, INRIA Sophia Antipolis, Nice, France, 2007.

[24] Claudiu Pozna and Radu-Emil Precup. Aspects Concerning the Observation Process Modelling in the Framework of Cognition Processes. *Acta Polytechnica Hungarica*, 9(1):203–223, 2012.

[25] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[26] Rahul Rao, Vijay R. Kumar, and Camillo J. Taylor. Visual servoing of a UGV from a UAV using differential flatness. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 743–748. IEEE, 2003.

[27] Rahul S. Rao, Vijay Kumar, and Camillo J. Taylor. Planning and Control of Mobile Robots in Image Space from Overhead Cameras. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, number April, pages 2185–2190. IEEE, 2005.

[28] Gregor Klančar and Igor Škrjanc. Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55(6):460–469, June 2007.

[29] Sašo Blažič. A novel trajectory-tracking control law for wheeled mobile robots. *Robotics and Autonomous Systems*, 59(11):1001–1007, November 2011.

[30] Claudiu Pozna, Fritz Troester, Radu-Emil Precup, József K. Tar, and Stefan Preitl. On the design of an obstacle avoiding trajectory: Method and simulation. *Mathematics and Computers in Simulation*, 79(7):2211–2226, March 2009.

[31] Igor Škrjanc and Gregor Klančar. Optimal cooperative collision avoidance between multiple robots based on Bernstein-Bézier curves. *Robotics and Autonomous Systems*, 58(1):1–9, January 2010.

[32] Steven M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[33] David Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall, 2003.

[34] Giuseppe Oriolo, Alessandro De Luca, and Marilena Vendittelli. WMR Control Via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, November 2002.

[35] Gene F. Franklin, J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems, Second Edition*. Addison-Wesley, 1990.

[36] Ajit Gopalakrishnan, Niket S. Kaisare, and Shankar Narasimhan. Incorporating delayed and infrequent measurements in Extended Kalman Filter based nonlinear state estimation. *Journal of Process Control*, 21(1):119–129, January 2011.

[37] Luka Teslić, Igor Škrjanc, and Gregor Klančar. Using a LRF sensor in the Kalman-filtering-based localization of a mobile robot. *ISA transactions*, 49(1):145–53, January 2010.