Elektrotehniški vestnik 77(1): 69–74, 2010
Electrotechnical Review, Ljubljana, Slovenija

Robot Grasping Using an ArToolKit Library

Bojan Nemec

Institut Jožef Stefan, Department for Automatics, biocybernetics and robotics, Jamova cesta 39, 1000 Ljubljana, Slovenija
E-mail: bojan.nemec@ijs.si

Abstract. The paper describes an efficient approach to robot grasping of randomly positioned and oriented objects by using a single camera. For object location and orientation estimation, an open source ArTool Kit library was used. A miniature camera was mounted at the robot gripper and an appropriate gripping pose was achieved by using visual servoing, thus eliminating the tedious calibration procedures. The overall control scheme was implemented in Matlab/Simulink using a newly developed robot programming language MatRol (Matlab Robot Language). An illustrative example is shown consisting of sorting randomly scattered cubes on the table by using a vision-controlled robot.

Key words: robotics, robot vision, visual tracking, robot programming

Robotsko prijemanje objektov z uporabo ArToolKit knjižnice

Povzetek. V prispevku opisujemo učinkovit način prijemanja naključno pozicioniranih in orientiranih objektov v prostoru z uporabo ene same kamere. To je mogoče samo v primeru, če natančno poznamo obliko in velikost predmeta ter na podlagi slike sklepamo o legi in orientaciji. To možnost nam omogoča javno dostopna knjižnica ArToolKit. Da bi se izognili zamudnim postopkom kalibracije, smo kamero namestili v robotsko prijemalo in problem reševali s kamero v povratni zanki (visual servoing). Celotno regulacijsko shemo smo implemetirali v programskem okolju Matlab/Simulink. Razvili smo tudi nov robotski programski jezik MatRoL (Matlab Robot Language), ki je zelo učinkovit pri raziskovalnem delu na področju robotike. Opisali smo ilustartiven primer sestavljanja napisa s pomočjo raztresenih kock na mizi, kjer pokažemo, kako lahko enostavno rešimo sicer za robota zahtevno nalogo.

Ključne besede: robotika, umetni vid, programiranje robotov

1 Introduction

In robotics, we are very often faced with the problem of determination of the pose (position and orientation) of an object in the environment to be used for robot manipulation. To determine the pose of an object, we use various senors, either CCD cameras, laser scanners, RF tags or special markers. The most universal sensors are CCD cameras but they require more sophisticated algorithms for object determination and recognition. In determination of the location of an object in a 3D-space, the most common solution are stereo cameras, since it is well known that it is not possible to estimate the position of a point in a 3D-space using a single camera. On the other hand, we know that humans can grasp and manipulate with objects using only one eye. This is possible because humans know the objects and gather from their size and appearance to the pose. Also when handling with unknown objects, we can successfully manipulate with them if we can compare their appearance with known objects in the neighborhood, e.g. unknown, previously unseen object placed on a table. This ability can be imitated also with computer vision. If we know the appearance of the object and have seen it from different angles, we can estimate its pose using a single camera. If we have a complete geometric presentation of the object, the pose can be also analytically calculated using a single image. The most simple case is a square. If we know the position of the corners in the image, we can exactly calculate the pose of the square in the space assuming that the camera is calibrated. This functionality is offered also by the ArToolKit library [3, 1]. In the paper, we show how a single camera can be used in robotic applications for manipulation with objects. We used a Mitsubishi Pa10 robot. The camera was fitted inside its gripper. In order to avoid calibration and to enhance the performance, accuracy and reliability of the system, we used visual servoing. The overall control scheme was realized in the Matlab/Simulink environment. The developed framework enables to control both the real and simulated robot without any modification of the program code [8, 9]. To describe the complex robot tasks, we developed a new robot programming language MatRoL in the Matlab/Simulink environment. Namely, it turned out that the standard Simulink blocks can generate an arbitrary trajectory, but can not provide all the flexi-

Received 29 September 2009
Accepted 11 January 2010

bility needed for the complex robot tasks, especially for experimental work.

2 ArToolKit library

ARToolKit is a software library for building Augmented Reality (AR) applications [2]. These applications allow overlaying virtual imagery on the real world. The poses of virtual objects in the captured camera images are determined by the marker poses. ArToolKit recognizes multiple markers in the scene, usually composed of a pattern in a black square. For example, Fig. 1 shows a dinosaur's skeleton standing on a real card. When the user moves the card, the virtual character moves with it and appears attached to the real object.

Some of the features of ARToolKit include:

- Single-camera position/orientation tracking.
- Tracking code that uses simple black squares.
- Ability to use any square marker patterns.
- Easy camera calibration code.
- Simple graphic library (based on GLUT).
- Fast rendering based on OpenGL.
- 3D VRML support.
- Simple and modular API (in C).
- Other language support (JAVA, Matlab).
- SGI IRIX, Linux, MacOS and Windows OS distributions.
- Complete source-code distribution.

The block scheme of the ArToolKit library is presented in Fig 1. The main benefit from using the ArToolKit Library in robotic applications is the ability to trace the marker pose using a single camera. The ability to overlay an object over a recognized marker helps to visualize the estimated orientation of the object to be manipulated with robot.

3 Visual servoing using kinematically redundant robots

Let us consider n degrees of freedom (DOF) of kinematically redundant serial manipulators, having more DOF than needed to accomplish the task, i.e. the dimension of joint space n exceeds the dimension of task space m, $n > m$ and $r = n - m$ denotes the degree of redundancy. Joint positions of the manipulator are described with n dimensional vector $\mathbf{q}$ and the end effector position (and orientation) is described with m-dimensional vector

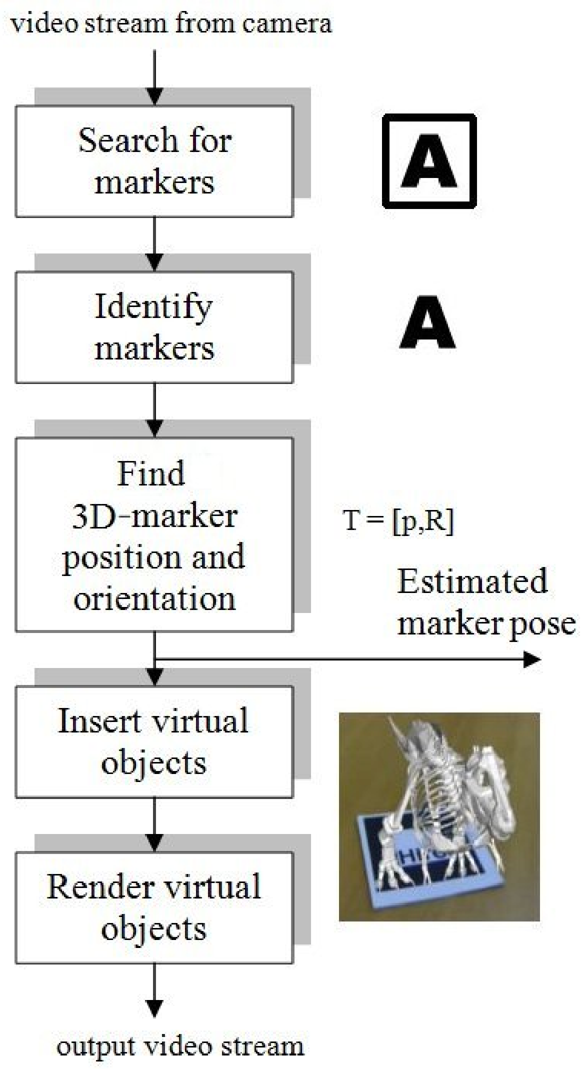

Figure 1. ArToolKit block scheme

$\mathbf{x}$. The relation between the joints and the task velocities is given by the following well known expression

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}} \tag{1}$$

where $\mathbf{J}$ is the $m \times n$ manipulator Jacobian matrix. The solution of the above equation for $\dot{\mathbf{q}}$ can be given as a sum of a particular and homogeneous solution

$$\dot{\mathbf{q}} = \bar{\mathbf{J}}\dot{\mathbf{x}} + \mathbf{N}\xi \tag{2}$$

where

$$\bar{\mathbf{J}} = \mathbf{W}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T)^{-1}. \tag{3}$$

Here, $\bar{\mathbf{J}}$ is the weighted generalized-inverse of $\mathbf{J}$, $\mathbf{W}$ is the weighting matrix, $\mathbf{N} = (\mathbf{I} - \bar{\mathbf{J}}\mathbf{J})$ is an $n \times n$ matrix representing the projection into the null space of $\mathbf{J}$, and ξ is an arbitrary n dimensional vector. We will denote this solution as the generalized-inverse based redundancy resolution at the velocity level [4]. The homogenous part of the solution belongs to the null-space of the Jacobian matrix.

The control law for the velocity-controlled robots can be derived from Eq. 2 by assigning

$$\dot{\mathbf{x}} = \mathbf{K}_p(\mathbf{x}_d - \mathbf{x}) + \dot{\mathbf{x}}_d + \mathbf{K}_f(\mathbf{f} - \mathbf{f_d}) \qquad (4)$$

and

$$\xi = \mathbf{K}_n \mathbf{q}. \qquad (5)$$

Eq. 4 represents control law P in the task space with the velocity compensation and force control term, and Eq. 5 describes the null space control [5]. Vectors $\mathbf{x}_d$ and $\dot{\mathbf{x}}_d$ are the desired task position and velocities of the robot, $\mathbf{f_d}$ is the desired task force and $\mathbf{f}$ is the force measured from the force sensor mounted in the robot wrist. $\mathbf{K}_p, \mathbf{K}_f$ and $\mathbf{K}_n$ are appropriate positive definite matrices representing the position, force and null-space control gains. In our case, we chose the most simple null-space control law, which simply preserves the joints of the robot as close as possible to 0. Null space control is required in order to assure conservative motion of the redundant robot during manipulation and depends on the current configuration of the environment. Obviously, we can not restrict all joints to be zero. In our case it is preferable to keep only the first joint close to zero and therefore matrix $\mathbf{K}_n$ has the form

$$\mathbf{K}_n = \begin{bmatrix} k_{n1} & 0 & .. & 0 \\ 0 & 0 & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & 0 \end{bmatrix}.$$

Note that the control law (4) feeds the desired velocities to the control unit of the Pa 10 robot which already incorporates velocity control. Therefore, Eq. 4 does not incorporate velocity control.

Visual tracking control is implemented as a pose control with appropriate $\mathbf{x}_d$ and $\dot{\mathbf{x}}_d$. We will describe positions and rotations using homogenous matrices,

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix},$$

where $\mathbf{R}$ is a 3×3 dimensional rotation matrix and $\mathbf{p}$ is a 3-dimensional position vector. Obviously, in order to grasp the desired object, robot-gripper homogenous transformation matrix $\mathbf{T_r}$ has to be aligned with desired transformation matrix $\mathbf{T_d}$ which defines the object gripping pose (Fig. 2). Assuming that the camera was in the gripper, the transformation required to align current robot pose $\mathbf{T_r}$ with desired pose $\mathbf{T_d}$ is the transformation among the camera transformation $\mathbf{T_c}$ and the desired transformation $\mathbf{T_d}$. Hence, visual control law is

$$\mathbf{T_x} = \mathbf{T_r}\mathbf{K_g}\mathbf{T_c}\mathbf{T_d^{-1}}, \qquad (6)$$

where $\mathbf{K}_g$ is a positive definite diagonal matrix of the video feedback gain and $\mathbf{T_x}$ is a new reference transformation. For the control purpose, orientations are described with quaternions in order to avoid singularity of the roll-pitch-yaw angles. In this case, the control law (4) has the form [6, 7]

$$\dot{\mathbf{x}} = \mathbf{K}_p \begin{bmatrix} \mathbf{p_x} - \mathbf{p} \\ Q_{\mathbf{x}} Q^{\mathbf{-1}} \end{bmatrix} + \dot{\mathbf{x}}_d + \mathbf{K}_f(\mathbf{f} - \mathbf{f_d}), \qquad (7)$$

where Q and $Q_{\mathbf{x}}$ denote the corresponding quaternion representation of rotation $\mathbf{R}$ and $\mathbf{R_x}$, respectively. Note that only the vector part of quaternion product $Q_x Q^{-1}$ is used in the control law (7) while the scalar part of the product is omitted.

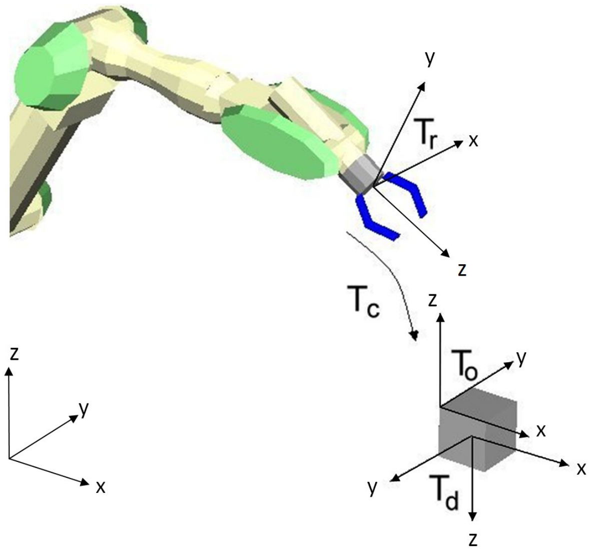

Figure 2. Coordinate systems in visual tracing

4 Experimental setup

The experimental setup consists of a 7-DOF robot arm Mitsubishi PA10, robot hand Barret Hand, inexpensive Logitech USB camera and PC computer [9]. The PC computer communicates with the robot power electronics using ArcNet with the frequency of 700Hz. ArToolKit was modified in order to communicate with Matlab/Simulink via the UDP protocol. The basic communication protocol between ArToolKit and Matlab/Simulink consists of commands for selecting the observed object and data with the position and rotation matrix of the detected object. The communication speed between ArTollKit and the Simulink is determined with the maximal camera frame rate, which in our case was 30Hz. Note that the UDP protocol permits to implement vision system on a separate computer as well as on the computer used in robot control. In our case, a Laptop computer with a Pentium dual-core 1.833 Mhz processor was fast enough to handle both tasks on a single computer. Figure 3 shows a

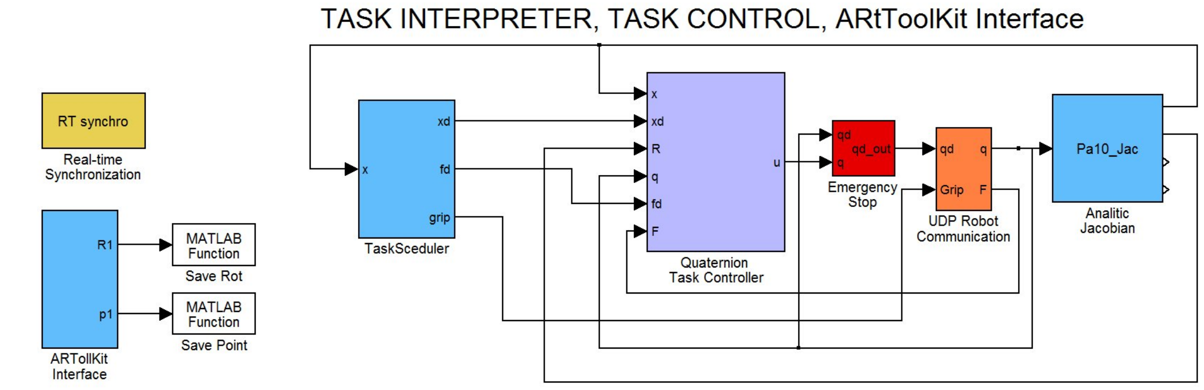

Figure 4. Simulink control scheme

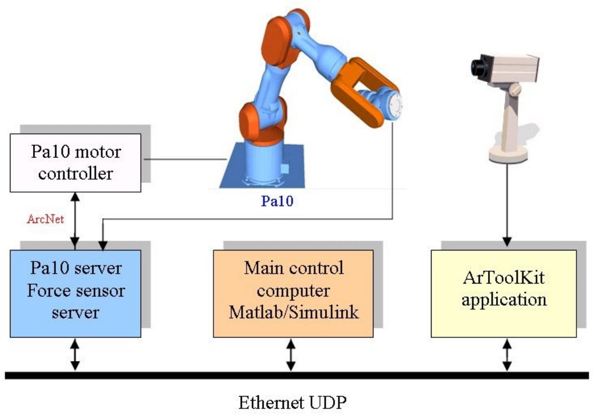
Pa10 motor controller
Pa10
ArcNet
Pa10 server Force sensor server
Main control computer Matlab/Simulink
ArToolKit application
Ethernet UDP

Figure 3. Block scheme of the experimental setup

block scheme of the whole setup. The control scheme implemented in Simulink is shown in Fig 4. Note that the basic position and force control were implemented in Simulink (Quaternion Task Controller block), while the visual tracking algorithm was implemented in MatRoL due to the low camera frame rate (Task Scheduler block). Signals from the ArToolKit (ArToolKit block) block were fed to the MatRoL using global variables (SaveRot and SavePoint Matlab function blocks).

5 Robot programming language MatRoL

When designing and testing complex robot tasks, it turned out that standard Simulink blocks, which can generate an arbitrary trajectory, can not provide all the flexibility needed for complex robot tasks, especially for experimental work in service and humanoid robotics where the desired motion depends on the system/environment states. A commonly used solution for the definition of robot tasks are robot languages. Therefore, we developed a MATLAB/Simulink block which can interpret the robot language. Included in the simulation it serves as a robot motion generator and supervisor. The developed interpreter module for the Matlab Robot Language (MatRoL) is BASIC like programming language extended with special commands for the robot control and supports all the Matlab interpreter commands. In this way we took advantage of the simple robot task definition and benefits of Matlab computation capabilities. The usage of the robot language is also advantageous for the educational purposes. Students can learn and accomplish their laboratory exercises much faster by using a robot language and the integrated environment allows safe testing of their algorithms on models and final tests on the real robots [8].

MatRoL is entirely written in Matlab. It has common instructions for the program flow (`IF THEN ELSE`, `FOR NEXT`, `REPEAT UNTIL`, `GOTO <label>`, `GOSUB <procedure_name> RETURN`) and special commands for the robot control (`FRAME`, `MOVE`, `APPROACH`, `DEPART`, `SPEED`, `ACCELERATION`, `FORCE`, `NULLMOVE`, `TRAJECTORY`). Also, all the Matlab commands can be executed within a MatRoL program as an instruction. In this way we can use the powerful Matlab matrix computation capability for controlling the robot pose and for complex computation generally needed when vision and force sensors are applied. MatRoL supports various interpolation modes in the Cartesian and task space, and supports also redundant robot systems, e.g. a special command `NULLMOVE` is used to define a self movement when kinematically-redundant mechanisms are used. It supports also multi-robot systems. Each robot in the simulation environment has its own MatRoL block, i.e. a special program. Program synchronization is done by assigning global variables. These can be signals, vectors, frames, or others. MatRoL supports the frame orientation definition in roll-pitch-yaw angles, Euler angles and quaternions, while the interpolation is accomplished using the quaternions.

6 Experimental results

In order to verify the efficiency of the proposed approach, the robot was assigned a task of sorting randomly scattered cubes marked with letters. The robot was ordered to compose the desired string. It started from the pose where it was able to see all cubes and check which cubes were visible. Then it grasped the randomly positioned and oriented cubes and composed the desired inscription string. During grasping the robot controlled also the wrist forces when an undesired collisions with the environment occurred. Figure 6 shows the robot performing the task. To allow for simplicity, only the part of the program in which the robot grasped and sorted one cube is presented here (Fig. 5). It is shown how easy complex tasks can be programmed. Figure 7 shows the time plot of the position and orientation error during visual servoing. The error decay is determined primarily by the camera frame rate. It can be seen that the robot reaches the desired position accuracy of 5 mm and the desired orientation accuracy of 6 degrees within the second, which is quite fast. Further improvements in the orientation accuracy can not be obtained since ArToolKit orientation estimation chatters for values around the unity rotation matrix. Note that in this experiment we have applied ordinary cheap web camera with a 30 Hz frame rate. Although this experiment serves only as a demonstration, it can be easily adopted for may industrial application, for example collecting randomly scattered object from the conveyer belt. Note also that this experiment is very robust regarding lighting conditions of the environment.

```
% VISUAL SERVO EXAMPLE - MatRol 2.0
%
GLOBAL ex ey atk_pos atk_rot %define variables
% scene observation frame and release frame
FRAME 1 =  [0.0,0.0,1.1,0.707,0,0.707,0]';
FRAME 2 =  [-0.393,0.0,0.85,0.707,0,0.707,0]';
! atks('A'); %track pattern with letter A
SPEED 0.5;   %define robot speed
ACC 0.5;     %define robot acceleration
GRIP 1;      %open the gripper
move 1;      %move to the observation frame
GOSUB vservo %execute visual servoing
GOSUB grasp  %grasp the object
GOSUB put    %move it
STOP
%
% ----------------------------------------
LABEL vservo
% ----------------------------------------
% visual servoing script vsm_Rot (Matlab)
REPEAT
     ! [xc,ex,ey] = vsm_Rot(atk_pos,atk_rot);
UNTIL (ex < 0.005) && (ey < 0.06);
RETURN
%
% ----------------------------------------
LABEL grasp
% ----------------------------------------
SPEED 0.1;              %set grasping speed
FORCE 1,[0 0 0 0 0 0];%turn force control on
TDEPART [0 0 0.15];    %approach the object
DELAY 0.5;
GRIP 0;                %grasp it
DELAY 0.5;
TDEPART [0 0 -0.1]     %depart Z in tool c.s.
FORCE 0;               %turn force control off
RETURN
%
% ----------------------------------------
LABEL put
% ----------------------------------------
SPEED 1;
APPRO 2 + [0.05 0 0 0 0 0 0]
SPEED 0.1;
move 2;
DELAY 0.5;
GRIP 1;              %release the object
TDEPART [0 0 -0.1]
RETURN
END
```

Figure 5. MatRoL program for visual servoing

7 Conclusion

In the paper we propose an ArToolKit library interface for single camera object tracking in robotics. We show how to implement an efficient visual-servoing algorithm. We developed a new powerful framework for robot programming and control, which is especially convenient for research and educational work in robotics. We describe an illustrative example demonstrating the capabilities of the proposed setup. The proposed visual servoing algorithm can be implemented in the majority of the industrial robots enabling the real time trajectory control in the task space. Hence, the algorithm is very convenient also for the low-cost industry automation.

8 References

[1] E. Woods, P. Mason, M. Billinghurst. MagicMouse: an Inexpensive 6-Degree-of-Freedom Mouse. Proceedings of Graphite 2003, Feb 11th-13th, 2003, Melbourne.

[2] Billinghurst, M., Kato, H., Poupyrev, I. (2001) The MagicBook: A Transitional AR Interface. Computers and Graphics, November 2001, pp. 745-753.

[3] H. Kato, M. Billinghurst. Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System. In Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99). October (1999), San Francisco, USA.

[4] D. N. Nenchev. Redundancy resolution through local optimization: A review. *J. of Robotic Systems*, 6(6):769 – 798, (1989).

[5] B. Nemec, L. Zlajpah, and D. Omrcen. Comparison of null-space and minimal null-space control algorithms. In *Robotica, 2007, 25(5):511–520*,(2007).

[6] L. Sciavicco, B. Siciliano. Modeling and Control of Robot Manipulator McGraw-Hill, New York 1996

[7] S. Chiaverini, B. Siciliano. The Unit Quaternion: A Usefull Tool for Inverse Kinematics of Robot Manipulator System Analysis, Modelling and Simulation, 35(35):45–60, 1999

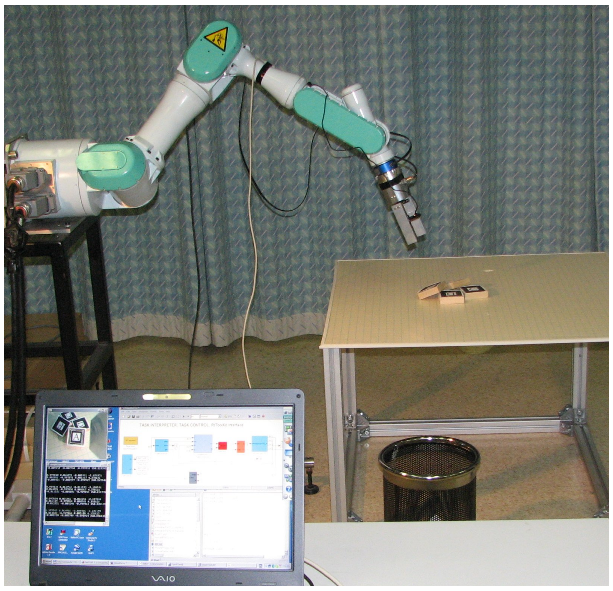

Figure 6. Robot during visual servoing

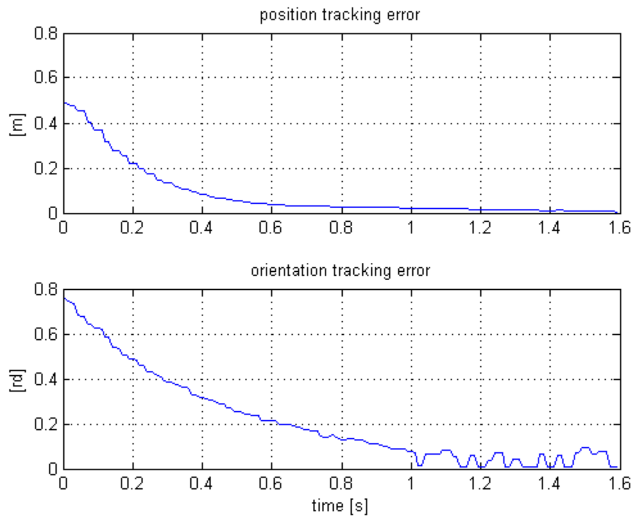

Figure 7. Tracking errors during visual servoing

[8] L. Žlajpah. Simulation in robotics. Mathematics an computers in simulation, 79(4):879–897, 2008

[9] L. Žlajpah, B. Nemec, D. Omrcen. MATLAB based robot control design environment for research and education. In MATHMOD Vienna 09 proceedings,304–315, 2009.

Bojan Nemec is a Senior Research Associate at the Deptartement of Automatics, Biocybernetics and Robotics at the Jožef Stefan Institute. He received B.Sc, M.Sc and Ph.D degrees from the University of Ljubljana in 1979, 1982 and 1988, respectively. In 1993 he spent his sabbatical leave at the Institute for Real-Time Computer Systems and Robotics, University of Karlsruhe. His research interests include robot control, robot simulation, sensor-guided control, service robots and biomechanical measurements in sport. Between 2002 and 2005 he was a task leader of the largest NAS European project EUROShoE. He has published over 100 conference and journal papers and is author of a patent, and co-author of a book.