Charles University in Prague
Faculty of Mathematics and Physics

DIPLOMA THESIS

Pavel Janda

Control of the Industrial Robot with Cameras

Department of Software Engineering

Supervisor: Prof. Ing. Václav Hlaváč, CSc.
Study programme: Computer Science

Acknowledgment

Above all, I would like to thank my supervisor Prof. Ing. Václav Hlaváč, CSc. for leading the diploma thesis, for all his expert advice and suggestions. Furthermore, an invaluable assistance was rendered by Ing. Oleksandr Shekhovtsov, Ing. Tomáš Werner Ph.D. and Ing. Vojtěch Franc, Ph.D., especially during the works on the software related tasks. I would also like to thank Ing. Pavel Krsek Ph.D. and Ing. Vladimír Smutný for their support of the technical part of this project.

I wish to thank my mother for the proofreading this diploma thesis and Jitka for her support and great patience with me.

I hereby declare that this diploma thesis is the result of my own work and that I only used the cited sources. I agree with lending and distribution of the thesis.

Prague, April 18, 2007 Pavel Janda

Abstract

Title: Control of the Industrial Robot with Cameras
Author: Pavel Janda
Department: Department of Software Engineering
Supervisor: Prof. Ing. Václav Hlaváč, CSc.
Supervisor's e-mail address: hlavac@fel.cvut.cz

Abstract: This thesis belongs to the implementation category of diploma theses. The work contributes to the EC project COSPAL, IST-2003-004176. The main goal of the project was the creation of the system of control and processing of the camera information for the assembly task. The demonstrator is used within the scope of the COSPAL project to solve a child's game – shape sorter puzzle. For solving the puzzle, the industrial robot CRS Robotics A465 was used. The robot has been available at the supervisor's place.
Keywords: COSPAL, robot control, electromagnet, homography

Název práce: Řízení průmyslového robotu s kamerami
Autor: Pavel Janda
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: Prof. Ing. Václav Hlaváč, CSc.
E-mail vedoucího práce: hlavac@fel.cvut.cz

Abstrakt: Tato práce patří do kategorie implementačních diplomových prací. Práce přispívá do projektu EU COSPAL, IST-2003-004176. Hlavním úkolem této práce bylo vytvoření systému řízení a zpracování informace z kamer pro úlohy montážního typu. Demonstrátor je používán v rámci projektu COSPAL na řešení dětské hry – skládací krabička. Pro řešení úlohy byl využit průmyslový robot CRS Robotics A465, který byl k dispozici na pracovišti vedoucího práce.
Klíčová slova: COSPAL, řízení robotu, elektromagnet, homografie

Contents

Chapter 1

Motivation

1.1 Introduction

Artificial cognitive systems are one of the key technologies which will probably have a significant impact in many areas of human activity in the future. Lately, a great emphasis has been given to the field of cognitive systems by agencies funding research worldwide. This diploma project contributes to one of European Commission supported projects with the acronym COSPAL, IST-2003-004176, running in years 2004 and 2007. COSPAL is the acronym for "**CO**gnitive **S**ystems using **P**erception-**A**ction **L**earning". The advisor of this diploma project is responsible for the team from the Czech Technical University in Prague participating in the project. Results of this work directly support experimental activities of the project. Cognitive abilities developed in the project have to be presented in the demonstrator exploiting an industrial robot.

The COSPAL project tries to develop a new system architecture and new learning strategies for the artificial cognitive system. The project COSPAL studies how the cognitive system can learn its capabilities from scratch based on its own exploration of the environment. The project also likes to answer the question which capabilities have to be innate and which can be learned. These tasks are rather complicated. The strategy chosen in the COSPAL demonstrator was to use a simple children toy for experimentation, a shape sorter puzzle. A kid, usually around two years of its age, learns to insert puzzle pieces (prisms of different colours and shapes of their bases) into the box through the holes in its top side. The base of the prism must match the shape of the hole.

The demonstrator is a system consisting of a six degrees of freedom (DOF) electrically driven industrial robot equipped with camera(s), planar working area, a gripper (oriented downwards) and the shape sorter puzzle. The actual setup is relatively easy. However, the tasks solved by it could be interesting from the cognitive system research point of view. The goal of this diploma project has been to adapt the existing industrial robot, its control system for the demonstrator, construct the gripper and write the basic system software. This software should allow the experimenters to explore the demonstrator in their research and experiments.

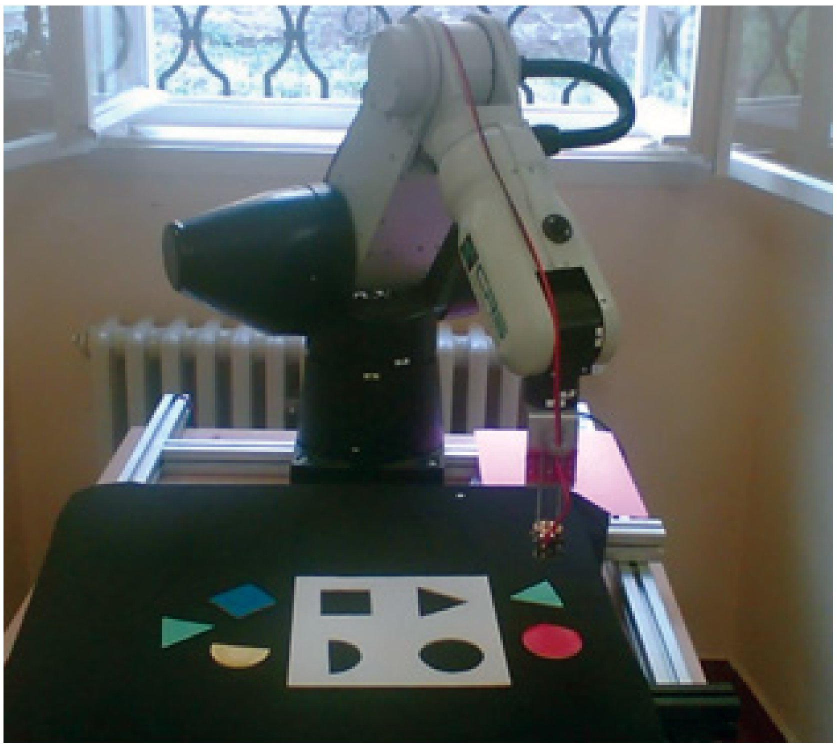


Figure 1.1: COSPAL project demonstrator solving shape sorter puzzle.

The biological cognitive system, e.g., a monkey or a child, is capable of reaching the state of a fully learned system. This is an ideal limit state which the COSPAL system will never achieve. The system has no information about the task and the environment. In the COSPAL scenario, the system has to find out the tasks according to the relations between objects in the environment. The more information the system has a priori, the simpler the solution can be. There are many ways how to advise the system, e.g., to show the basics of the game – provide a learning sequence (find an object, move a robot gripper to the object, grip the object, move the gripper to a box, release the object into an appropriate hole), give feedback for single actions and sequences of actions.

The results of this diploma project were used for the first time in experiments prepared for the review meeting of the COSPAL project held in the laboratory of the advisor's institution in September 2006. The higher level of the system exploring reinforcement learning was prepared for the review meeting by the fellow student Mr. Radim Krupička. His diploma project had been running in parallel with the reported one.

Some more information about the COSPAL project is provided in the following section.

1.2 Project COSPAL

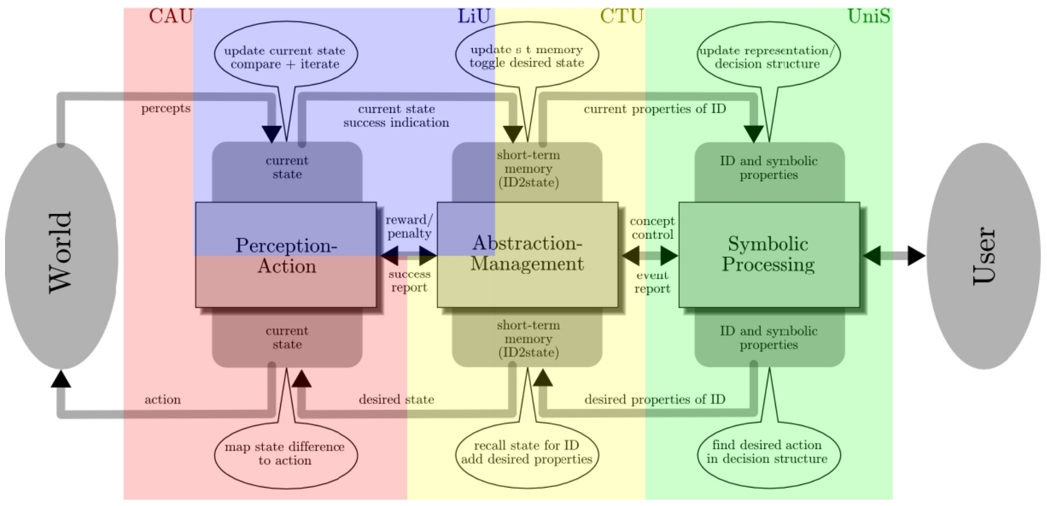
The purpose of this project is to investigate design principles and architectures for technical systems which are capable of learning basic cognitive skills in a similar way as humans do. It might appear surprising that, for instance, learning basic motor skills as done by children during the first three years of their life are hard to simulate in a technical system [6].

To understand why the "simpler" capabilities of humans are more difficult to simulate than "higher level" capabilities like, for instance, playing chess, one has to consider the different approaches in human learning and in implementing technical systems, which appear to be intelligent.

Nowadays, technical systems typically consist of a predefined set of rules, which control the system's actions depending on the inputs. Unforeseen inputs and constellations cannot be processed by such a system. Playing chess is a very well suited problem for this kind of system design because the chess game follows exact rules and has a finite (although large) number of distinct constellations. Simulating certain cognitive capabilities of a one-year-old child, however, cannot be implemented by a rule based system for two reasons: First, there are no exact rules in basic behavioural schemes, they remain too fuzzy and qualitative to be used in a technical system. Second, the number of possible inputs and the number of possible constellations are infinite in a way that for any number of rules there exist cases that are not covered by the rules.

Due to these limitations of rule-based systems, several attempts were made in the past to design systems showing fuzzy behaviour as well and where rules were replaced with different methods of machine learning. In machine learning, the technical system is supposed to adapt itself to a large set of examples, the training data, and will generalize from the number of used examples to all potential constellations.

The new aspects to be considered in context of the COSPAL project are a suitable learning strategy and system architecture for simple perceptual schemes where the associations of input and behaviour are triggered by the system's own actions. Furthermore, a suitable interface to rule-based systems is required in order to make the basic capabilities available to a system in a larger context and to solve tasks. In order to evaluate the progress of the COSPAL architecture, a technical system will do the same what small children do: learn to manipulate simple objects like building blocks and to puzzle objects into a shape sorter.



CAU
LiU
CTU
UniS
update current state compare + iterate
update s t memory toggle desired state
update representation/ decision structure
percepts
current state success indication
current properties of ID
current state
short-term memory (ID2state)
ID and symbolic properties
World
Perception-Action
reward/ penalty
success report
Abstraction-Management
concept control
event report
Symbolic Processing
User
current state
short-term memory (ID2state)
ID and symbolic properties
action
desired state
desired properties of ID
map state difference to action
recall state for ID add desired properties
find desired action in decision structure

Figure 1.2: Overview of the COSPAL project architecture.

Chapter 2

Problem formulation

The aim of the diploma thesis is to design and implement software support enabling to use the industrial robot CRS A-456 and its control unit in the demonstrator of the COSPAL project. The robot has been available and the demonstrator has been developed at the Center for Machine Perception, Faculty of Electrical Engineering, Czech Technical University in Prague. The software support should allow the higher layers of the demonstrator to access the robot.

The CRS A-456 and its control unit is equipped with the dedicated language for robot programming named TROL. In the COSPAL demonstrator both MATLAB and C++ modules are used. The MATLAB serves as the user interface. One of the tasks of this diploma thesis is to extend the existing robot control software. The extension should support the feedback loop having vision components in it. One or more stationary cameras should be supported. In the original diploma thesis assignment, the camera on the robot arm was foreseen. However, the COSPAL demonstrator did not need this option and thus methods for moving the camera held by the robot gripper were not developed.

The above mentioned general assignment can be decomposed into several subtask:

Extension of the robot control system. This will extend and upgrade the control system of the robot. Calls from higher level languages as, e.g., MATLAB and C++ should be supported.

Image acquisition module. The aim is to design and implement software for acquiring images by a camera so that they could be used for the assembly task.

Interface to shape sorter puzzle. The demonstrator uses a simplified twodimensional (2D) shape sorter puzzle. The robot is used for manipulating the puzzle. The aim of this module is to create a simple dedicated hardware of the puzzle which allows exploiting a simple magnetic gripper. The other task is to extend the software by manipulation abilities dedicated to the shape sorter puzzle. Means for adjusting coordinate systems of the robot and puzzle have to be provided.

Software methodology, documentation, testing. The diploma thesis should follow common software design methodology. The design and implementation phases should be documented.

Chapter 3

Available robot and its software

In this chapter the following topics are discussed:

1. CRS Robotics A465 Industrial Robot.
2. CRS Robotic C500 Robot Controller.
3. Computer.
4. Dragonfly®Colour Camera.
5. TROL Robot Control Library.
6. MATLAB Interface for CRS A465 Robot.
7. MATLAB Toolboxes.
8. 2D COSPAL Puzzle Simulator.

These topics are divided into two sections. Section 3.1 discusses all the hardware devices that were used for the project's purposes. The following section, Section 3.2, concerns the most important software equipment. Information contained in the text below originate from the following literature – [8], [3] and [7]. The cited sources are introduced in the text.

3.1 Hardware

The following devices were used at the workspace. The A465 industrial robot (see Section 3.1.1) with the C500 robot controller (see Section 3.1.2) were at the project's disposal. The controller was connected to the computer (see Section 3.1.3) which was equipped with the corresponding software (the software will be discussed later in Section 3.2). A colour camera was attached to the computer. The camera (see Section 3.1.3) was supposed to take the images of the scene. The images would be processed subsequently with the software.

3.1.1 CRS Robotics A465 Industrial Robot

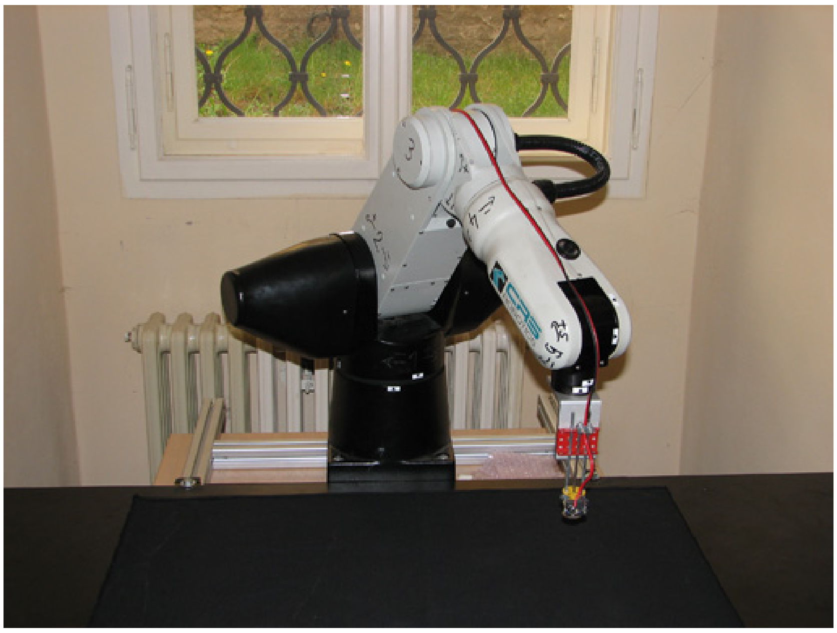
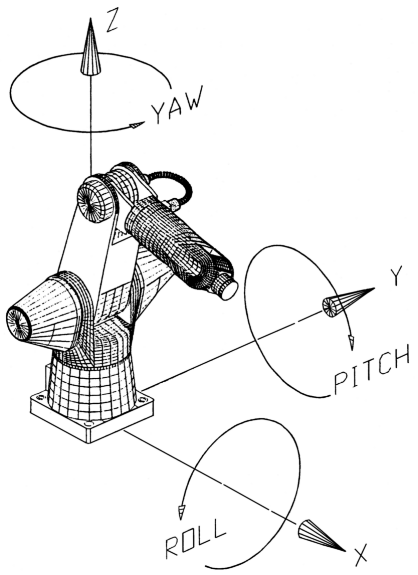


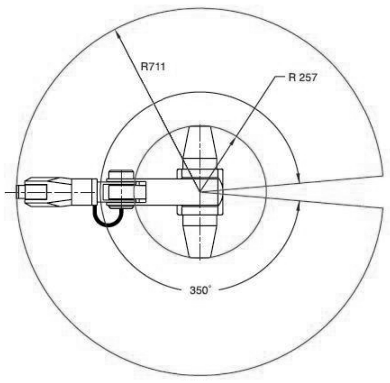
Figure 3.1: CRS Robotics A465 Industrial Robot.

The CRS Robotics A465 robot (see Figure 3.1) is designed with the same range of motion and payloads as the human arm. The robot has six DOF (degrees of freedom), realized by a sequence of seven links (rigid component connecting the joints) and six revolute joints. Each joint is actuated by a DC motor via a harmonic drive. The robot supports a maximum payload of 2 kg. Typical uses for the A465 robot include a wide range of laboratory automation and industrial processes such as machine loading, dispensing, polishing, deburring, cutting, drilling, trimming, and parts transfer. The last segment of the robot arm is equipped with a tool flange that allows easy mounting of a variety of tools.

The robotic arm is situated in the six-dimensional Cartesian coordinate system. The position and orientation of the arm is described by coordinates <X, Y, Z, YAW, PITCH, ROLL>(see Figure 3.2(a)). The working area of the robot is in Figure 3.2(b).



(a) Robot coordinate system.

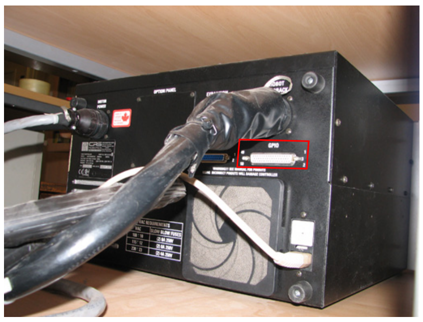


(b) Robot working area.

Figure 3.2: Robot world coordinates and working area.

3.1.2 CRS Robotics C500 Robot Controller

(a) Front view.



(b) Back view; the location of the GPIO connector is marked.

Figure 3.3: CRS Robotics C500 Robot Controller.

All requests to the A465 robot are processed by the C500 robot controller (see Figure 3.3) which is equipped with an Intel 286 CPU. The C500 provides 2 serial communication channels, one of which is connected to a special input device, the so called teach pendant. The other serial channel is connected to a DOS or Windows 3.1/95/NT/2000 PC; normally, this connection is used for character based RAPL command transfer. The RAPL is the acronym for "**R**obotic **A**utomation **P**rogramming **L**anguage", a programming language developed by CRS Robotics. The controller does, however, also support a proprietary master-slave protocol called ACI (**A**dvanced **C**ommunication **I**nterface) which can be used for binary data transfer (e.g., download of RAPL programs into the controller memory).

Furthermore, the controller provides several digital input and output lines. This GPIO (**G**eneral **P**urpose **I**/**O**) connector is located at the controller rear panel (see Figure 3.3(b)).

3.1.3 Other Devices

PC

The machine that was used during the whole work is described in this paragraph. The computer was equipped with AMD Athlon$^{\text{TM}}$XP 2400+ CPU, 512 MB RAM, 80GB HDD and Nvidia GeForce4 MX 440 GPU. The C500 robot controller was connected via serial port, the camera used FireWire interface. The computer ran under the Microsoft Windows 2000, 5.00.2195, Service Pack 4. The software equipment (important for the work) consisted of the MATLAB 7.1.0.246 (R14) Service Pack 3, CRS Robcomm Robot Communication and Programming Software for Windows v4.32i and Borland C++ 3.1 compiler.

Camera

The image acquisition device used in the diploma project was a Point Grey Research colour camera Dragonfly®(see Figure 3.4). The Dragonfly®was a FireWire camera equipped with a Computar 12,5 mm f1:1,3 lens.

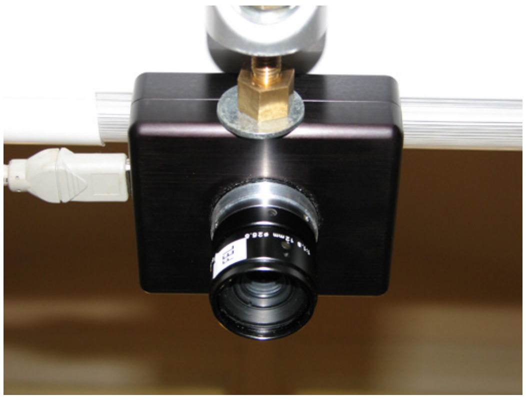


Figure 3.4: Point Grey Research Dragonfly®camera.

3.2 Software

The following text covers the software that was in use. The C500 robot controller is equipped with the TROL Robot Control Library (see Section 3.2.1). This will be discussed first. The following topic is the MATLAB Interface for CRS A465 Robot (see Section 3.2.2). A short introduction of the MATLAB system is included in this part. Next section concerns the most important MATLAB Toolboxes (see Section 3.2.3) used in this project. Section 3.2.4 introduces 2D COSPAL puzzle simulator.

3.2.1 TROL Robot Control Library

Programming of the A465 industrial robot has normally to be done in the RAPL robot control language. While being well suited for typical industrial applications, RAPL lacks the richness of data types and control structures of modern high level languages, features which will be needed when implementing more complex (e.g., 3D vision or real time control) tasks. For this reason, an A465 control library was designed and implemented, which enabled the programmer to control the robot from within his application written in "C(++)" [8].

Robot Programming

The A465 is normally programmed in RAPL, a BASIC-like interpreter language. It is possible either to issue single commands interactively – whereby the PC acts as a terminal – or to load a complete RAPL program into the controller memory, where it is executed. See [2] for further details.

What makes the A465 attractive is the possibility of downloading 8086 code (which was generated with a standard assembler or C-compiler) into the controller's program buffer and execute it there; in CRS terminology, these programs are called PCPs (**P**rocess **C**ontrol **P**rogram). Most of the low level system routines (the so called RAPL BIOS) that initiate and control the actions of the robot arm – similar to the PC BIOS – can be called by a PCP via a software interrupt; thus, almost any desired action of the arm can be initiated by a PCP. However, more complex applications, e.g., from the field of computer vision, cannot be implemented as a PCP for several reasons:

- The size of a PCP (program code + data) may not exceed 64K.
- It is not possible to use hardware devices such as frame grabbers, LAN adapters, etc.

- PCPs are executed on an Intel 286 CPU, and the code must not be register optimized. Thus, the execution speed of a PCP is not sufficient for many applications [8].

TROL

TROL is a layered communication protocol designed for the data exchange between a DOS/Windows PC and the C500 robot controller. It provides the PC-programmer with a comfortable, high level API (Application Program Interface) that gives him access to almost all RAPL BIOS functions. The TROL API is a subset of the RAPL command set, but does in some cases provide more functionality (e.g., the *TROLDepart()* command accepts full 6 DOF depart vectors, whereas the RAPL *DEPART* command does only allow for translations). The parameters of any API function are transferred via a serial communication channel to the controller, where the TROL server (which is implemented as a PCP) decodes them and calls the appropriate subroutine of the RAPL BIOS. The application programmer, however, does not have to concern himself with PCP programming.

A TROL API function call will be handled as follows [8]:

1. First, range and plausibility checks of the function arguments are applied. Then, the function code and the parameters are transferred via serial connection to the controller.

2. At the controller, the PCP server receives the request and decodes the parameters. If appropriate, some additional plausibility and safety checks of the arguments will be applied. Finally, the RAPL BIOS is called and its return value transferred back to the calling API function.

3. The API function receives the BIOS code and returns it to the caller.

It is important to understand that TROL API functions behave strictly synchronously [8]: they will not return before the corresponding RAPL BIOS subfunction has been called (or an error has occurred). However, this does not imply that the requested action (e.g., robot arm movement) has finished on return of the API function!

Any errors will be reported by API return codes. Furthermore, TROL provides several call back addresses which are called in the event of communication errors, thus allowing the user to write his own error handlers [8].

The TROL Layer Model

This paragraph describes the communication protocol used by TROL to establish a communication link between the application and the PCP server and will give an overview over the TROL's layered communication model. The architecture of the TROL's communication model follows roughly the suggestions of the OSI standard. Below, the four TROL communication layers and their OSI [1] counterparts are given [8]:

TROL	**OSI**
Physical Layer (1)	Physical Layer (1)
Data Link Layer (2)	Data Link Layer (2)
Transaction Layer (3)	Session Layer (3)
Application Layer (4)	Presentation Layer(6) & Application Layer(7)

3.2.2 MATLAB Interface for CRS A465 Robot

TROL Robot Control Library effectively raised the abilities of the A465 robot in the field of 3D-vision or real time control tasks. Because of the need for controlling the robot in a mathematical environment as MATLAB, the MATLAB Interface for CRS A465 Robot has been designed [3].

The MATLAB System

The MATLAB System consists of five main parts. These are the MATLAB language, the working environment, Handle Graphics, the mathematical function library and the **A**pplication **P**rogram **I**nterface (API). The MATLAB language is a high-performance language for technical computing. It can be used for both small simple programs and large complex applications. The second part is the MATLAB working environment. This is a set of tools and facilities that the user and programmer work with. The examples are data-managing and debugging. The third main part of the MATLAB System is the handling of graphics. This graphical system makes it possible to visualize 2D and 3D data, process images or build a complete Graphical User Interface on MATLAB applications. The MATLAB mathematical function library contains a vast collection of computational algorithms from elementary functions like sums to more sophisticated functions like matrix

[1]OSI: **O**pen **S**ystems **I**nterconnection, a (logic) model for communication networks pro posed by ISO.

inverse. The final part of the MATLAB System is the Application Program Interface. This library allows programmers to write for example C programs that can interact with MATLAB through a dynamically linked calling routine (DLL) [3].

The MATLAB API

The MATLAB API provides a way how to let MATLAB interact with external data and programs. Functions supported by the API include the possibility to call C or Fortran programs from MATLAB, to import and export data to and from the MATLAB environment and the ability to create client/server relationships between other applications and MATLAB. The most important feature of the MATLAB API for this project is that in MATLAB it is possible to call C subroutines as if they were built-in functions. These MATLAB callable C functions are referred to as MEX-files. MEX-files are dynamically linked subroutines that MATLAB can automatically load and execute.

The source code for an MEX-file consists of two distinct parts. The computational routine contains the code that should be implemented in the MEX-file. The other part is the gateway routine. This routine connects the computational routine with MATLAB by the entry point mexFunction comparable to main in C-file [3].

MATLAB Interface for CRS A465 Robot

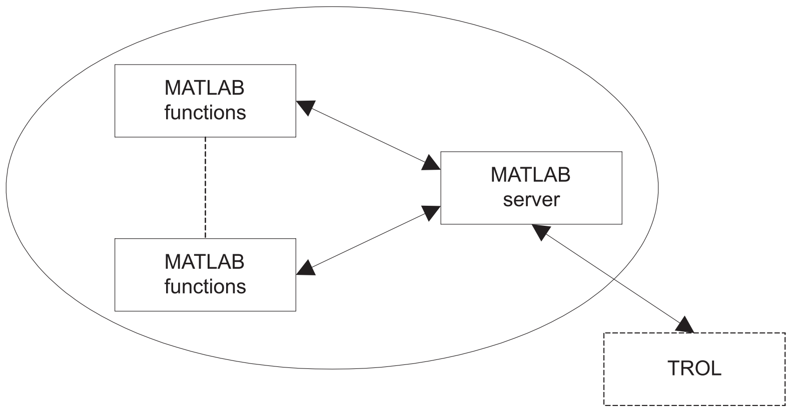


Figure 3.5: Architecture of MATLAB Interface for CRS A465 Robot.

All of the functions of the interface are separate DLL's. These DLL's will communicate through a DLL server with the TROL library. When calling the *TINIT* command, the server is initiated. This server ensures that the robot will remain initialized until it is cleaned up with *TCLEANUP*. When first started, the server function is locked. This capability was introduced to make sure that communication with the robot controller is maintained when all functions are cleared in MATLAB. If no lock were applied, the server would also be cleared and it would be impossible to reinitialize the robot. The architecture of the MATLAB Interface for CRS A465 Robot is shown in Figure 3.5 [3].

3.2.3 Used MATLAB Toolboxes

There is a very important feature of the MATLAB environment which is a family of add-on application-specific solutions called toolboxes. The toolboxes allow applying specialized technology. The toolboxes are, in fact, comprehensive collections of MATLAB functions that extend the MATLAB environment to solve particular classes of problems, such as signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

Image Acquisition Toolbox

The Image Acquisition Toolbox implements an object-oriented approach to image acquisition. The connection between MATLAB and specific image acquisition devices is represented by a special object. Various aspects of the acquisition process can be controlled. All image acquisition is initiated by a trigger. The toolbox supports several types of triggers. Acquired frames are stored in a memory buffer. The frames imported into the workspace behave as any other multidimensional numeric array. Finally, the image acquisition application can be enhanced by using event callbacks.

Image Processing Toolbox

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox supports a wide range of image processing operations, such as image processing, analysis, visualization, and algorithm development. Noisy or degraded images can be restored, shapes and textures analysed, and two images registered. Most toolbox functions are common MATLAB files and thus can be easily modified.

3.2.4 COSPAL 2D Puzzle Simulator

The COSPAL 2D Puzzle Simulator is an artificial environment for solving cognitive related tasks. It was created by Erik Jonsson from the cooperative Linköping University and is entirely implemented in MATLAB system. Vojtěch Franc modified the simulator, so that it corresponded with needs of the Prague COSPAL team. The main purpose of the simulator was to provide a test bed for different architectural experiments. The simulator design took also into account the compatibility with the real demonstrator in the future. The artificial environment can be seen in Figure 3.6. More information can be found in [4].

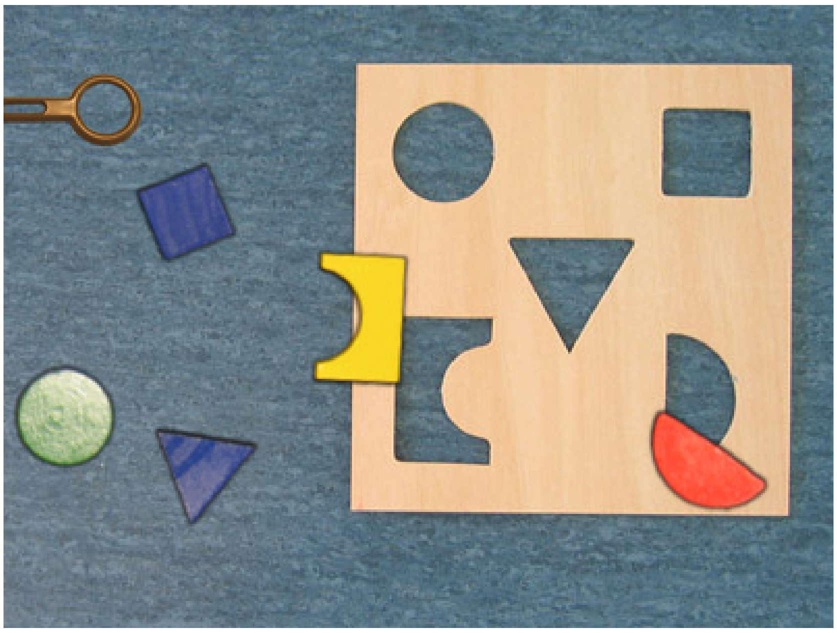


Figure 3.6: Artificial environment of the COSPAL 2D puzzle simulator.

Chapter 4

Implementation

Chapter 2 discussed the diploma thesis tasks. The entire work is described by those four tasks. The way the tasks divide the work is not too suitable for writing the thesis. It is difficult to recognize hardware from software related task. Another logical structuring of the text is used in the following chapters. Furthermore, the sectioning honours the chronology of all actions. This stands both for chapters and their contents. The work will be split into the following two chapters:

1. COSPAL Project Demonstrator.
2. MATLAB Module for the COSPAL Project.

The construction of the COSPAL project demonstrator is covered in Chapter 5. The demonstrator consists of the robot (the robot arm is equipped with an electromagnet) and its controller, the puzzle, the camera and the computer which is used for operating the whole demonstrator. The tasks "Extension of the robot control system" and "Interface to shape sorter puzzle" are solved in that chapter.

Chapter 6 describes the MATLAB module that is used in the COSPAL project. This module is responsible for many things, such as camera control, robot control or integration to the COSPAL project. This chapter solves the tasks "Image processing and analysis module" and "Interface to shape sorter puzzle".

Chapter 5

COSPAL Project Demonstrator

Introduction

The construction of the COSPAL project demonstrator will be described in this chapter. The status of the project at that time will be depicted in the following lines. The A465 robot, the C500 robot control system, the camera and the control computer were at the project's disposal. There was no puzzle, no suitable grabbing device. The camera had to be located somewhere. The whole working environment had to be created.

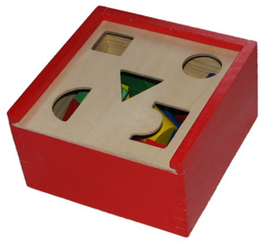
The chapter will be divided into four sections. Each section will discuss some part of the demonstrator construction. Section 5.1 will describe the construction of the 2D puzzle. Everything related to the electromagnet construction will be solved in Section 5.2. The electromagnet would not work without some software modifications. This problem is handled in Section 5.3. Section 5.4 will deal with the construction of a stand. This stand will be designed to support the camera.

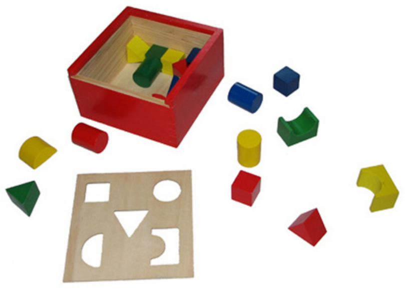
5.1 2D Puzzle Construction

In this work, a simplified 2D version of the classical 3D shape sorter puzzle is used. One instance of a common 3D puzzle is shown in Figure 5.1. The 3D puzzle is a box, whose top face has holes of several shapes in it. The top face of the box is removable which allows the kid to remove the pieces and repeat the game.

Each hole represents a different geometrical shape (e.g., a triangle, a square, a circle). The different puzzle pieces fit into different holes. The puzzle pieces are usually coloured. Each piece is painted only with one colour.

After some discussions in the COSPAL project consortium, it was decided


(a) Puzzle 1.


(b) Puzzle 2.

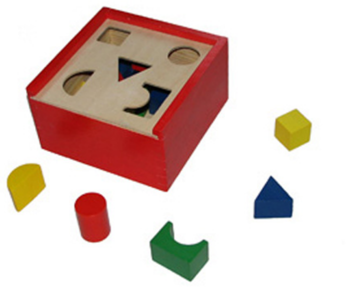

(c) Puzzle 3.

Figure 5.1: Shape sorter puzzle.

that a simple 2D version of a shape sorter puzzle will be used. This was a very important decision. It enabled to get over some technical difficulties easily, which could emerge with the 3D version of the puzzle. The main advantage of that conclusion was having only a little impact on COSPAL project itself. The aim of the project is in cognitive learning, not in solving low-level problems. For example, this change effected both the image processing (3D scene is more complex than 2D one) and the objects manipulation in the first place.

As it was mentioned earlier, the task was simplified by going from 3D to 2D. This meant that a new shape sorter puzzle was needed. No suitable puzzle was available, so a special one had to be made. Here is the list of all the requirements the simplified 2D puzzle was expected to have:

- The base board replaces the top face of the classical 3D shape sorter puzzle. It contains holes where the puzzle pieces will be placed. This allows the pieces to be removed easily. This operation can be performed by the robot.
- Each hole has a different shape. The puzzle pieces have to be recognized

according to their shape. The shapes should have reasonable variability. This would prevent pointless complications.

- Each piece fits only one hole. There are no pieces with the same shape.
- The puzzle pieces can be distinguished by colour. Each piece has different colour. The picked colours should be contrast enough.
- The manipulation method should be simple but effective. It is important that the puzzle pieces can be placed one over another.

First of all, it was necessary to choose suitable materials for the puzzle construction. The puzzle pieces were expected to keep their shape perfectly, in addition to that, be lightweight enough. Otherwise it would be quite difficult for the electromagnet to operate with them. The material that fitted best the needs was plywood. Plywood is widely used by hobby modellers because of its typical features, such as resistance to cracking, shrinkage, twisting, etc. Afterwards the base board with holes and appropriate pieces were cut off. Each piece fitted exactly into one corresponding hole.

Another important property of the puzzle, the coloured pieces, had to meet the given requirements. It was necessary for the robot to be able to distinguish the pieces according to the their colour. There were two options – either to paint all the pieces with a different colour or use glued coloured papers. After some discussions the painting method was eliminated. The advantage of the papers consisted in easy implementation and having less problems in image segmentation (coloured paper has the same tone almost in all places). Opaque papers were founded the most suitable. They cause only a little or hardly any reflection and thus improve conditions for successful image segmentation. However, before pasting papers on the pieces, one more thing had to be done.

The last required feature of the puzzle was the ability of the pieces to be gripped with the electromagnet (in other words – pieces are “grippable”). The idea of the manipulation with the pieces was quite simple. The robot places the electromagnet right on the piece. The electromagnet is turned on, the piece is gripped. The robot moves the electromagnet to a different position. The electromagnet is turned off. The piece is released and the robot moves away. To make this work, certain modifications of the puzzle pieces were required. A thin steel plate was glued on the top of each piece. This enabled proper manipulation with the pieces.

At that moment the puzzle was almost finished. The placing of the papers was the last thing which remained. At first, two layers of the papers were used. This intention was not correct and caused many difficulties. The two layers showed up to be too thick. The electromagnet had problems with gripping the pieces. The pieces also tended to fall of the electromagnet during the electromagnet movement. The solution was obvious – only one layer of paper had to be fasten with glue.

Before proceeding to the next task, the electromagnet construction, one more thing has to be mentioned. The puzzle base board and the puzzle pieces were placed on the board that was in the reach of the robot arm. In attempt to improve the image processing of the scene, some minor adjustments to the board were tried. Covering the board with a black cloth was the most significant one. The tests proved this to be a good idea. The base board and pieces were contrastive enough with the background.

5.2 Electromagnet Construction

In the previous section, the puzzle construction is discussed. According to the decision, which was made at the beginning of the work, 2D version of the shape sorter puzzle was chosen. The problem of how to grip the puzzle pieces had to be solved. Consequently something capable of gripping a flat plywood piece was needed. Two possible options were considered: to use either a mechanical gripper or an electromagnet.

The first option, the mechanical gripper, was soon rejected. This approach had only one real advantage. The A465 robot was already equipped with a working gripper. It was part of the original robot accessories. However, the original gripper did not fit the purpose – to grip the puzzle pieces. It was necessary to find a way how to modify this gripper. It was expected to be capable of operating with the flat puzzle pieces. No simple working solution regarding the gripper came up. For example, one proposal assumed the usage of the original gripper (without modifications) and some sticks added to the puzzle pieces. The stick would be approximately 5 cm long, placed in the centre of the piece, perpendicular to the piece. However, it would be impossible to lay a piece over another one.

As it was mentioned earlier, there were two options. The gripper, the first option, was not suitable for the task. The second option, the electromagnet, was chosen. The biggest disadvantage of this approach was that, in fact, no working electromagnet was actually available. All possible alternatives were considered. The outcome of the discussion was the following decision: an electromagnet designed specially for the project's purposes should be constructed and used. This was not only about the electromagnet construction but also about other certain modifications. These things will be dealt with for the rest of this section and in Section 5.3. At this moment, it would be a good idea to make a list of all the necessary tasks leading to the working electromagnet:

- The electromagnet must be constructed (the device itself). It should be able to manipulate with the puzzle pieces and should work reliably.
- The electromagnet has to be mounted on the robot arm. The construction should be simple but it should protect the electromagnet from the damage caused by a careless manipulation.
- The computer has to control the electromagnet. It is needed to solve and implement the way the computer communicates with a device the electromagnet is connected to.

According to the preceding list, the first task was the electromagnet construction. That was quite a simple task. The main part of an electromagnet is an electromagnetic coil. The easiest way how to obtain one is to use an old relay. A relay is an electrical component which is, in fact, built around a coil. The only "problem" was to select the right one. The electromagnet had to be strong enough to hold the puzzle pieces. The relay was downgraded – the expendable parts were removed. Adding some necessary electrical components as some semiconductor diodes, a resistor and, of course, some wires an operational electromagnet was created (see electrical scheme in Figure 5.2). The components were mounted on the relay coil. Finally, the wire was attached to the relay. At that moment, the functional electromagnet was available.

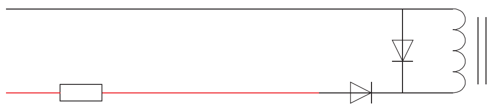


Figure 5.2: Electrical scheme of the electromagnet.

The first task was successfully completed. The second task, mounting the electromagnet on the robot arm, was a more complex one. As it was said before, the A465 robot is equipped with the mechanical gripper. The gripper is attached to the robot at the end of the arm. However, the arm's tool flange allows removing the gripper and replacing it with another device. The robot had been used in different projects in the past and various devices had been mounted on the arm. Some special holders had been developed for those purposes. The most appropriate, the L-shaped holder, was mounted in the A465 tool flange.

The biggest issue, talking about the second task, had not been solved yet. The electromagnet had to be connected in a way that would prevent the electromagnet damage during the manipulation. The most critical operation was gripping and placing the puzzle pieces. The attachment must guarantee a secure approach of the electromagnet, for example, to the puzzle pieces, the base board and the background desk.

There were many suggestions on how to solve this problem. The winning proposition was based on a movable attachment. The main idea was to use a thin short bar (approximately 20 cm long). The bar was fixed on one side to the electromagnet. On the other side, it went through a hole in a component that was attached to the L-shaped holder. See Figure 5.3. This was only a brief description. The structure of the connection mechanism was more complex. The most suitable tool for the construction was a child's construction set called "Merkur". This is a Czech product that had educated two or three generations of "future designers". It seems to be very simple but enables rather complicated constructions to do. The second task hereby could be declared as done.

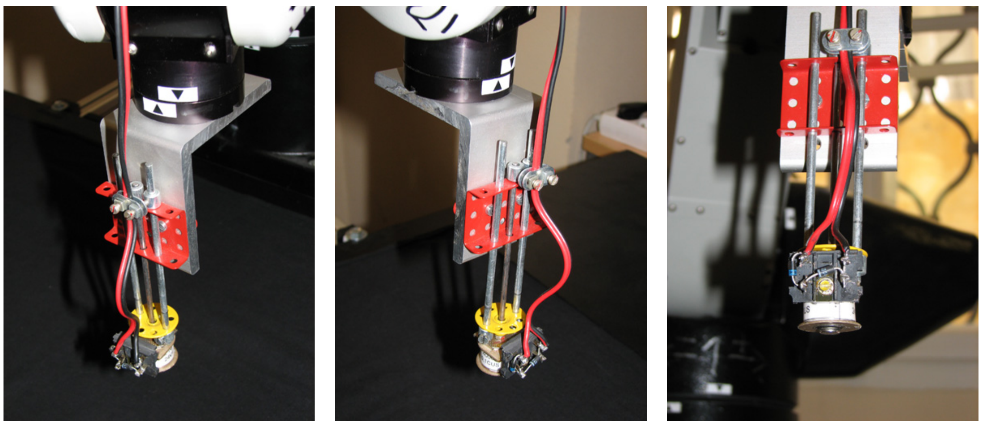

(a) Electromagnet 1.
(b) Electromagnet 2.
(c) Electromagnet 3.

Figure 5.3: Electromagnet.

The last remaining task from the electromagnet task list was about solving and implementing the communication with the computer. This task could be divided into these three following subtasks:

- The electromagnet needs a power supply for proper work. The power supply should provide enough power to hold the puzzle pieces.

- The power supply has to be controlled via computer. It must be possible to turn the electromagnet on and off programmatically.

- According to the previous tasks either a new software will be created or the current one upgraded to allow the MATLAB system to control the electromagnet.

Two possible ways were taken into the consideration: to use either an external device or the robot accessories.

The first idea concerned the external device capable of powering the electromagnet. It was also equipped with a serial port. A serial port RS-232 is a serial communication physical interface which enables connecting a device to a computer. The computer would be able to operate the electromagnet. Finally, the MATLAB environment would be updated. A MATLAB library allowing the electromagnet control would be created.

The second idea concerned using a special communication interface the robot is equipped with. The interface would provide the computer with the ability to power and control the electromagnet. Also in this case, the MATLAB environment would have to be updated. Besides, the robot software would be modified.

Both approaches had their advantages and disadvantages. For better understanding, here is a short overview:

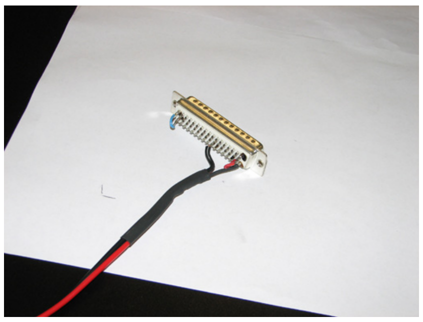
	The external device	**The robot interface**
Advantage	standalone MATLAB library	not all-in-one system
Disadvantage	all-in-one system	somebody else's sources

In other words, a standalone MATLAB library would be programmed for the external device. It would manage the communication through the serial port. The bad thing is that an external device would be needed. The robot would be dependent on that device. It would not be the all-in-one system. It would be, in fact, a one-time solution with no capability of the further enhancement in the future. On the other hand, the robot interface would benefit from the all-in-one solution. The simulator would not be dependent on an external device. That solution would enable to utilize upgraded robot software in further projects. However, modifying and maintaining somebody else's source codes is always very problematic, especially when the documentation is poor.

The second approach was picked after some discussions with some researches of both ideas. The all-in-one system and the opportunity to prepare the background for further projects decided that the second idea won. Some difficulties were expected to be encountered during the robot software upgrade. However, the advantages were assumed to outweigh the drawbacks.

The C500 robot controller [1] has a total of 16 inputs and outputs that can be used for any general purpose. These facilities are isolated from the main controller power and logic circuits either mechanically or optically. This configuration makes it very easy to interface to the C500. The general purpose, or GPIO are those general purpose inputs and outputs that can be used to interface other machines to the C500, either to be controlled by the C500 or just to be monitored by the C500. The GPIO connector allows the C500 to control and monitor external events. The C500 has 16 isolated inputs, and 16 isolated outputs, 12 outputs are low current optically isolated relay drivers with 60 mA capacity. 4 outputs are 3 A relay contact outputs, with NC and NO contacts. All relays are connected to a common point, which is fused on the front panel. The GPIO connector is a female DD-50S connector.

50 pin male connector had to be constructed to use the GPIO connector. The GPIO connector provided both power for the electromagnet and the ability to control the electromagnet. The wire leading from the electromagnet was attached to the connector. The functional description of the GPIO connector from the robot manual was used to connect appropriate pins on the male connector. This was important for proper functionality of the electromagnet. See Figure 5.4.


(a) 50-pin connector.

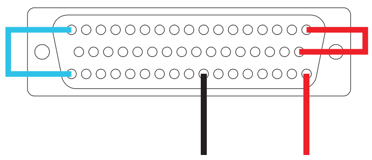

(b) Electrical scheme of the 50-pin connector.

Figure 5.4: 50-pin connector and its electrical scheme.

This section solved the construction of electromagnet. At this moment, the functional electromagnet capable of connecting to the robot was available. However, it was not able to work with GPIO port. The third task of the electromagnet task list had not been finished yet. The third subtask concerning the software modifications still had to be accomplished. This will be finished in the next section.

5.3 Software Upgrade

The third task of the electromagnet construction list will be accomplished in this section. The last thing that prevented the computer from communication with the robot was the functionless software part. Two different interfaces were available with the robot. The older one is “TROL robot control library” (hereinafter TROL library). The second one is extending the TROL library and implements an interface to the MATLAB environment. It is called “MATLAB Interface for CRS A465 Robot” (hereinafter MATLAB-A465 interface). The documentation and source codes of both TROL library and MATLAB-A465 interface were examined. As a result, the modification of both interfaces was necessary. Some information about this topic can be found in Section 3.2. For more information, the reference to the documentation can be found in the bibliography (see page 47).

Before getting started with the modifications description, a brief summary will introduce the main ideas of how the things work. A MATLAB server runs on the computer. When a function from the MATLAB-A465 interface is called, it is processed with the server. The server communicates with the TROL library. The function is recognized and TROL sends the function identification and the data to the C500 robot controller. An instance of the TROL PCP server is running in the C500 robot controller. According to the identification data, the function is recognized. The PCP server handles this function and calls RAPL BIOS directly. Furthermore, the error checks are run during the function calls. The results are returned back for further error handling.

Finally, it is possible to examine the modifications part by part. The detailed description is in the code documentation located on the enclosed CD (for CD contents see Appendix A). The MATLAB-A465 interface will be the first and will be followed by the TROL library. The modifications of the MATLAB-A465 interface:

1. Two functions were added to the MATLAB-A465 interface.
2. MATLAB server was modified to be able to proceed new function calls.

The modification of the MATLAB-A465 interface was relatively straightforward. Two functions were added to the interface – a simple MATLAB function without any DLL library – *tdigout* – and an ordinary DLL function of the MATLAB-A465 interface – *trapl*. The first one represents the top-level function of the electromagnet control. It is responsible for turning the electromagnet on and off. It calls the second function. The *trapl* function is more complicated and more important. It processes the calls from the *tdigout* function. The main purpose is to call the MATLAB server and pass it the data. The main idea of the electromagnet control is in using RAPL commands. The MATLAB-A465 interface functionality was extended so that it is possible to use RAPL commands directly within the MATLAB system. The RAPL command is passed straight to the *trapl* function.

The second modification of the MATLAB-A465 interface are changes in the MATLAB server itself. The main reason for doing so is the need to provide the functionality for the *trapl* function. When the *trapl* function calls the MATLAB server, the server must handle this function. Accordingly, a section which performs that activity was added. In this section, a TROL library function *TROLRAPL* is called. *TROLRAPL* function will be examined below. The description of the MATLAB-A465 modification hereby can be considered as finished.

The modifications of the TROL library:

1. TROL library was modified to be able to handle function calls from MATLAB server.

2. Handling of the new function was added to TROL PCP server.

The work required updating the TROL library showed up to be less complicated than it had been expected earlier. At first, a function had to be added to TROL. This function – *TROLRAPL* – is called from MATLAB-A465 interface. The purpose of the function is to work as an intermediate between MATLAB server and TROL PCP server. When the *TROLRAPL* function is called, it sends the passed data from the MATLAB server through the serial port to the C500 robot controller.

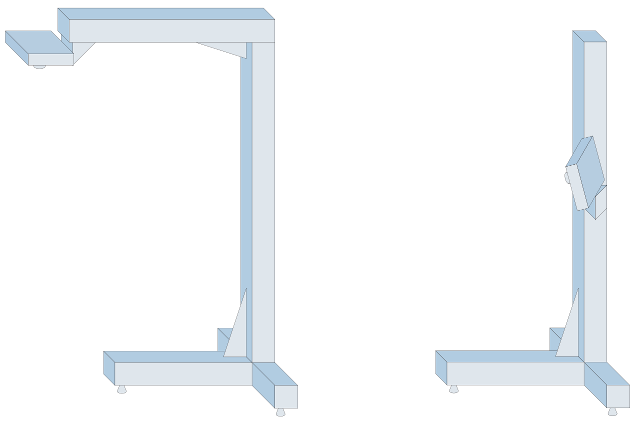
The last modification involved the TROL PCP server which runs in the C500 robot controller. Two changes were needed to finish the software upgrade. One function was slightly updated (it was just a formal update and thus it will not be discussed further) and a new function was added. First, the PCP server processes the data sent through the serial port, then recognizes the required service function (this is the update) – the *HandleRAPL* function and finally, it calls this function. The functionality of the *HandleRAPL* function lies in direct communication with the RAPL BIOS.

The modification of the TROL PCP server was the last remaining thing that prevented the electromagnet from working. From this moment on, fully operational electromagnet was at the project's disposal. It was possible to control the electromagnet from the MATLAB system. The third task of the electromagnet construction list was successfully accomplished.

5.4 Camera Installation

The COSPAL project demonstrator will be finished in this section. At one moment, the robot was equipped with the electromagnet, the electromagnet was working and everything could be controlled via the MATLAB-A465 interface which ran on the computer. The last thing to be done was to install a camera.

There are two possible approaches – a camera can be either static or dynamic. A common definition of a static camera is that the camera does not move. Such a camera is usually located on a stand, has an overview of the whole working area and needs only one initial calibration. On the other hand, a dynamic camera changes its position often. It can be attached to the robot – usual location is the robot arm near the gripper so that it looks the same direction as the gripper, has only a limited view and needs regular recalibrations. A two-camera system was expected at the first stages of the work. One camera would be static and the other would be attached to the robot arm. Some of the early experiments indicated that a well positioned static camera could do the job. The camera mounted on the robot arm showed up to be redundant.


(a) The refused camera stand construction.

(b) The final camera stand construction.

Figure 5.5: Camera stands.

The first idea where to place the camera was the top view position – the camera would be directly above the working area. A new stand was built for this purpose. It can be seen in Figure 5.5(a). However, the camera suffered from vibrations. The structure of the stand did not prevent such a behaviour. The problem was with the gallows-like part. It is quite hard to build a stand where camera does not suffer from various vibrations. This was too bad. Thus, a new solution came up.

The majority of the problems originated from the fact that the top view position was required. It was difficult to build an appropriate stand only with limited resources. This seemed to be a significant complication. However, the solution was simple. A fact that all the obtained images can be transformed by means of a homography was used (see Section 5.4.1).

Because of the homography, the top view camera position was no longer required. The camera could be located in any general position. The gallows-like part of the stand could be removed and the camera was attached to the stand, which can be seen in Figure 5.5(b). The tests proved this approach to be correct. The camera installation task was finished.

5.4.1 Homography

A homography is one of often-used operations in the computer vision. That is the reason why it deserves a more close explanation. At first, some examples of the homography will be shown. Afterwards, the term itself will be introduced. The following information was gathered from [9], more details can be found also there.

Two simple cases can be pointed out. The first example is a 2D homography of a planar scene and its representation in a pinhole camera. This can be used to rectify images of planar scenes to frontoparallel view. An example (from this project) of an image mapped by a 2D homography is in Figure 6.2. The second example is two pinhole cameras that share a single centre of projection of a 3D scene (planar or non-planar). Panoramic images can be stitched from the sequences of the photographs [9].

Finally, the term homography[1] can be explained. Homography is any mapping $\mathcal{P}^d \rightarrow \mathcal{P}^d$ that is linear in the embedding space $\mathcal{R}^{d+1}$. That is, a homography is given up to unknown scale and written as

$$\mathbf{u}' \simeq H\mathbf{u}, \tag{5.1}$$

where H is a $(d+1) \times (d+1)$ matrix. The transformation maps any triplet of collinear points to a triplet of collinear points. Non-singularity of the H matrix causes the distinct points to be mapped to the distinct points. The (5.1) uses homogeneous vectors. To become familiar with homogeneous notation, it is instructive to show in detail how the non-homogeneous 2D point $[u, v]^T$ (e.g., a point in an image) is actually mapped to the non-homogeneous image

[1] In literature, homography can be also referred to as "collineation" or "projective transformation".

point $[u', v']^T$ by H using (5.1). With the components and the scale written explicitly, the equation reads

$$\alpha \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{5.2}$$

The third coordinate of u' is set to 1. This tacitly assumes that u' is not a point at infinity. That means $\alpha \neq 0$. Elimination of the scale α is desired to compute $[u', v']^T$. This yields the expression

$$u' = \frac{h_{11}u + h_{12}v + h_{13}}{h_{31}u + h_{32}v + h_{33}}, v' = \frac{h_{21}u + h_{22}v + h_{23}}{h_{31}u + h_{32}v + h_{33}},$$

familiar to people who do not use homogeneous coordinates. Note that compared to this expression (5.1) is simpler, linear, and can handle the case when u' is a point at infinity. These are the practical advantages of homogeneous coordinates [9].

The homography has some important features, the invariants. It is collinearity, closely related tangency and cross-ratio on a line. There are some subgroups of the projective transformation group – affine, similarity, metric (Euclidean, isometric) and identity subgroup. Each subgroup enhances the invariants of the previous one, thus it is possible to write

$$projective \supseteq affine \supseteq similarity \supseteq metric \supseteq identity.$$

The identity is the most strict one. Everything is invariant. There are other subgroups, but these are often met in computer vision [9].

Any homography can be uniquely decomposed as $H = H_P H_A H_S$ where

$$H_P = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{a}^T & b \end{bmatrix}, H_A = \begin{bmatrix} K & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, H_S = \begin{bmatrix} R & -R\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{5.3}$$

and the matrix K is upper triangular. Matrices of the form of H_S represent Euclidean transformations. Matrices $H_A H_S$ represent affine transformations and matrices $H_P H_A H_S$ represent the whole group of projective transformations [9].

A frequent task in 3D computer vision is to compute the homography from (point) correspondences. A set $\{\mathbf{u}_i, \mathbf{u}'_i\}_{i=1}^m$ of ordered pairs of points presents a correspondence. Each pair corresponds in the transformation. To compute H, the homogeneous system of linear equations need to be solved

$$\alpha_i \mathbf{u}'_i = H\mathbf{u}_i, i = 1, \ldots, m \tag{5.4}$$

for H and the scales α_i. This system has $m(d+1)$ equations and $m+(d+1)^2-1$ unknowns; there are m of the α_i, $(d+1)_2$ components of H, while -1 suffices to determine H only up to an overall scale factor. $m = d+2$ correspondences are needed to determine H uniquely (up to scale) [9].

Summary

This chapter was aimed at the construction of the COSPAL project demonstrator. Section 5.1 described the construction of the 2D shape sorter puzzle. The two following sections discussed all tasks necessary to putting the electromagnet into service. The first of the two sections, Section 5.2, contained the hardware related tasks, such as the electromagnet construction. The software related tasks were described in Section 5.3. The last section of this chapter, Section 5.4, was concerned with the works required for the camera system construction. The demonstrator consisted of the robot arm (equipped with the electromagnet) and its controller, the shape sorter puzzle, the camera system and the computer. The demonstrator could be controlled from the computer. The robot control system was extended and one part of the "Interface to shape sorter puzzle" task was also accomplished.

Besides the accomplishment of the requested tasks a positive side-effect was achieved. The utilization of the GPIO port for project's purposes is based on controlling only one pin (out of 50). The pin has two states – on and off. This was the way how the electromagnet was turned on and off. The idea of the side-effect is that, in fact, the TROL library and the MATLAB-A465 interface can be upgraded to enable the full control of the GPIO port.

The next chapter will deal with the creation of the module for MATLAB system. This module will be an interface that could be used in COSPAL project.

Chapter 6

MATLAB Module for the COSPAL Project

Introduction

This chapter is dedicated to the MATLAB module for the COSPAL project. The COSPAL project demonstrator was already available at the time the works on this module started. The main task was to create this module so that it would be possible to use the demonstrator for COSPAL project's purposes.

One of the significant advantages of the COSPAL project is the modular concept. Each module has a specific interface. It is possible to switch the corresponding modules. For example, different segmentation methods can be implemented, different object detection approaches can be tested, the software simulator can be switched for the real demonstrator. The COSPAL project uses a modified COSPAL 2D puzzle simulator. The simulator can perform all the tasks the real demonstrator is capable of. The idea was that the replaced module would work without any restrictions. The images generated by the simulator would be replaced by the camera output, the real robot arm would move instead of the simulator one. The puzzle pieces could be recognised, gripped, moved and released again. A module with the same interface as the software simulator had to be created. This would enable the demonstrator to be used in further experiments.

As mentioned earlier, the COSPAL project demanded an operational demonstrator. The demonstrator was already built (see Chapter 5) and it was possible to control it. The robot control had been implemented by means of the MATLAB-A465 interface (MATLAB Interface for CRS A465 Robot) functions calls so far. However, this was very impractical indeed.

6.1 Object-Oriented Design of the MATLAB Module

Only a basic overview will be examined here. The details are situated in the code documentation on the enclosed CD. The object-oriented design of the MATLAB module can be seen in Figure 6.1. There are three objects – the camera (see Section 6.2), the robot (see Section 6.3) and the control object (see Section 6.4).

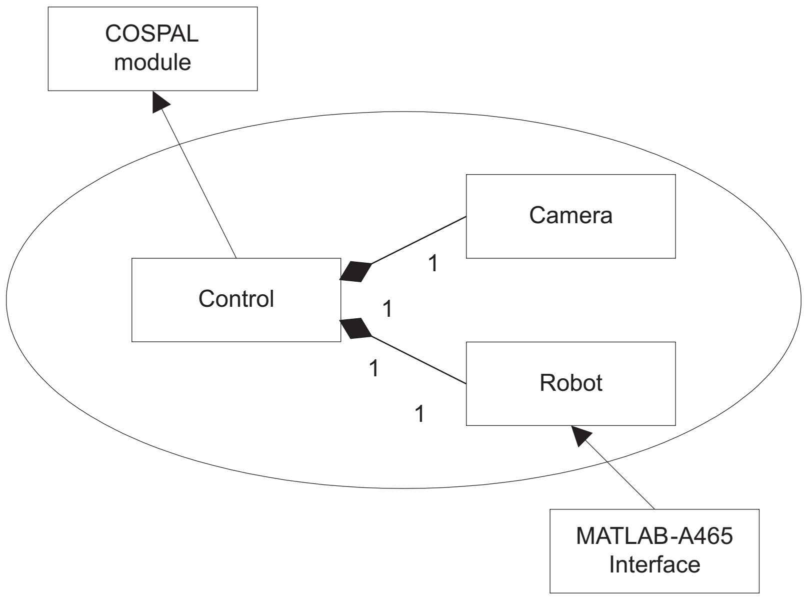


Figure 6.1: Object-oriented design of the MATLAB Module.

The camera object is responsible for all the things relevant to the camera functionality. The next item is the robot object. It communicates right with the MATLAB-A465 interface. In fact, it encapsulates the MATLAB-A465 interface functions. These functions are no longer used directly. The manipulation with the robot is realized by the methods belonging to the robot object. The methods are not only exact copies of the MATLAB-A465 interface functions, but they also use the functions to create complex actions. The topmost item of the hierarchy is the control object. It has some important features. The control object defines the communication interface of the MATLAB module. The methods are called in order to exploit all the module abilities. The control object is superior to both other objects. More complex actions can be created. They are based on the methods of the other two objects. The camera object methods take care of the image acquisition and the robot object methods take care of the manipulation with the objects in the scene.

The following text is divided, according to the objects, into three parts. The first section is interested in the camera object, the second in the robot object and the third in the control object.

6.2 Camera Object

The first object in the object-oriented design is the camera object. This is a very simple object. It is responsible for the camera initialization in the MATLAB environment and acquiring the images of the scene. The initialization is a standard procedure but the image acquisition is a more interesting one.

There are two options on how the acquired image looks like. As mentioned above, the camera is located in a general position but the top view is also required. Therefore, the first option represents the image in the way it is obtained directly from the camera. While the second option returns a transformed image (homography). In this case, the top view transformation is taken into account. Any transformation can be used though, see Figure 6.2.

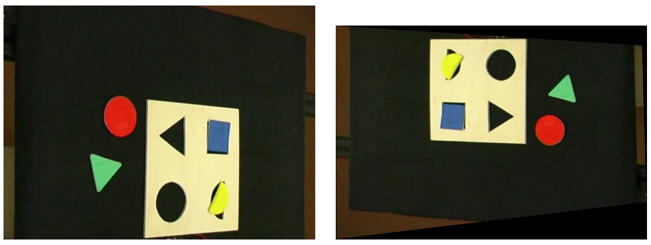

(a) Normal image. (b) Transformed image.

Figure 6.2: Image acquired from the camera before and after transformation.

6.3 Robot Object

This is the second object in the object-oriented design of the MATLAB module. This object is far more complicated. It communicates with the MATLAB-A465 interface. It directly uses the interface function to control the robot. The robot object is responsible for the robot initialization and robot operability. The MATLAB-A465 interface functions are used in two possible ways, either their counterparts were implemented or the functions created a new compound method (the sequence of the MATLAB-A465 interface functions represents a more complex action). Here are two examples. First, the implementation of the "grip" ability of the robot:

1. Approach the working surface.
2. Turn on the electromagnet.
3. Draw away from the working surface.

Second, the implementation of the "restart" ability of the robot:

1. Turn off the electromagnet.
2. Set higher speed of the motion.
3. Move the robot arm the initial "ready" position.
4. Turn the semifinal joint so that the electromagnet points to the ground.
5. Set the robot to the standby position.
6. Set lower speed of the motion.

There are a few methods in the robot object that are created in such a way, e.g., grip, release, robot homing, move to position, set speed. The complete list will be presented in documentation on the enclosed CD.

6.4 Control Object

The last object in the object-oriented design is the control object. This topmost object, see Figure 6.1, creates the main communication interface of the MATLAB module. This interface enables the compatibility with the COSPAL 2D simulator module. The methods of the control object are complex procedures that often combine both the robot and the camera methods, e.g., the calibration method.

The start-up method of this object causes both the camera and the robot initialization. All of the control object methods are based on the various ones of the robot and the camera object. These are the most interesting methods of this object: image acquisition (several variants are available), camera calibration, the robot arm movement, performing a special action, object removal. Here goes the description of each method:

Image Acquisition. The image acquisition methods are described in Section 6.4.1.

Camera Calibration. The calibration of the camera is described in Section 6.4.2.

The Robot Arm Movement. The movement of the robot arm around the working area. It changes, in fact, the position of the electromagnet in a 2D working area.

Performing a Special Action. This method is described in Section 6.4.3.

Object Removal. This method differs from the others. It is used when an inserted object has to be removed from the environment. The object is just erased from the simulator environment. In the real environment, a supervisor must remove the object.

There are three methods that deserve a little more detailed description. The first one explains the image acquisition process. The second one implements the calibration of the camera. The third one serves the other COSPAL project modules as a communication interface.

6.4.1 Image Acquisition

According to the two different criterions (the acquired image is transformed to the top view, the robot arm is present at the scene), four different variants can be found. The overview of all possible variants is presented in the chart below:

	with robot arm	**without robot arm**
top view	image	scene image
normal view	real image	real scene image

Each variant has its specific purpose to be used for.

Image method. This variant either serves for presentation purposes or is used by other COSPAL project modules. For example, the detection of successful grip of a puzzle piece.

Real image method. The only reason for having this variant is a quick preview function. It can be used for various setup purposes.

Scene image method. This is the most important variant because it is used by other COSPAL project modules. For example, the image is processed by the image segmentation and the object detection method.

Real scene image method. This variant is used by the calibration method because the transformation is not available during the calibration process. Furthermore, the robot arm does not cover the working area.

6.4.2 Camera Calibration

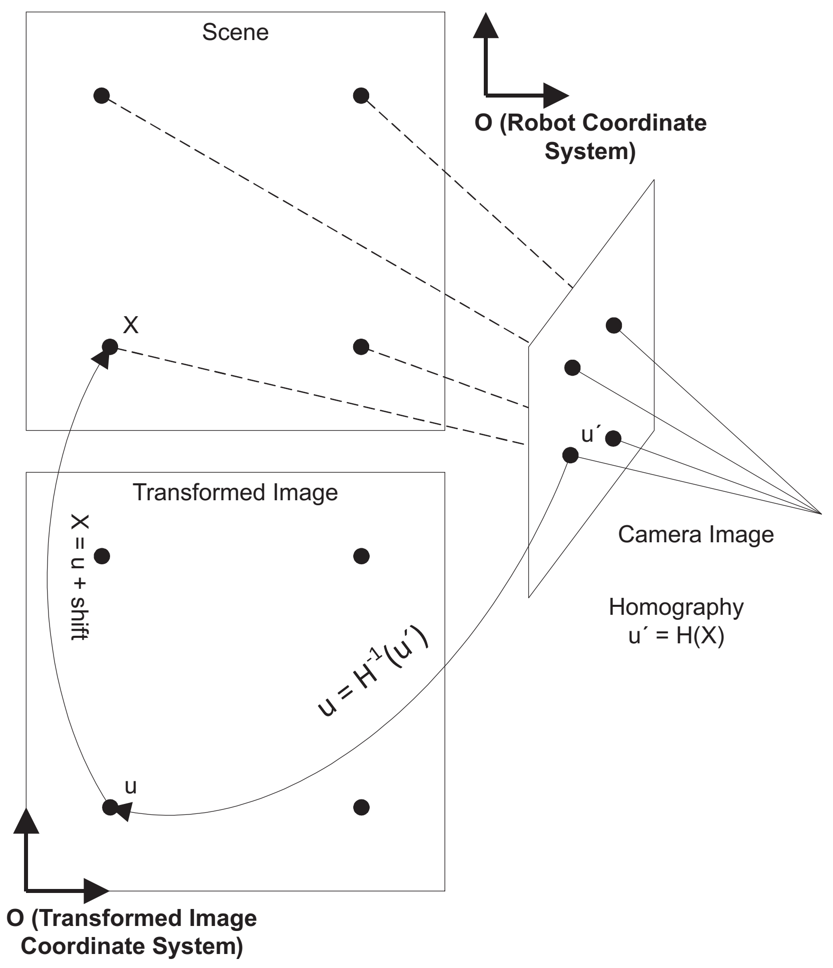
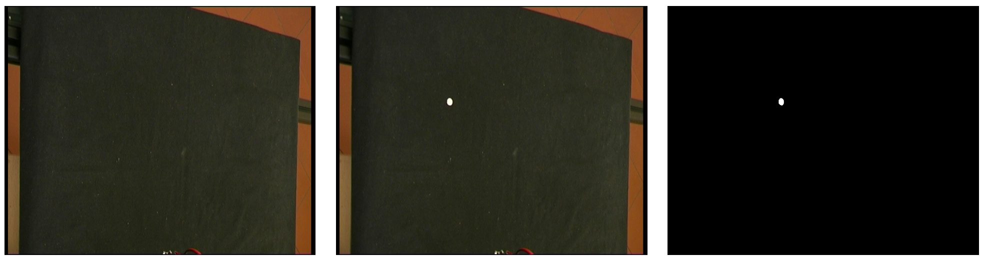


Figure 6.3: Homography and transformations between coordinate systems.

The camera calibration is a very important operation. A camera placed in a general location can be used. The result of this operation is a certain transformation matrix. Such a matrix enables various transformations, e.g., a top view image can be created.

The camera calibration process is based on the homography (see Section 5.4.1). Two things had to be worked out. First, a top view was required. Second, a transformation from the image to the robot coordinate system was needed. The determination of the homography would solve both cases. The solution can be seen in Figure 6.3. There is a scene and an image acquired by a camera. Providing that the homography is known, the top view can be obtained. Furthermore, the coordinate system of the transformed image is determined. The robot coordinate system differs only in the position of the origin. Thus, a point in the transformed image coordinate system can be easily transformed to the robot coordinate system.

As mentioned earlier, the homography is needed. According to the (5.4) in Section 5.4.1, this can be computed from the point correspondences. The homography is mapping $\mathcal{P}^2 \rightarrow \mathcal{P}^2$, and thus $d = 2$ and $m = 4$. Four correspondences are required to determine the homography uniquely.



(a) The empty working area. (b) The fourth calibration point. (c) The difference between previous images.

Figure 6.4: Camera calibration example.

The calibration method used by the MATLAB module is quite simple. Before the operation can be realized, an empty working area is required. Then the first image is acquired (the "real scene image" variant is used during the calibration process – see Section 6.4.1). A little white object (a little metal plate covered with white paper) is gripped by the electromagnet. The electromagnet is moved step by step into the four different positions in the working area (the corners of a rectangle). At each corner, the object is released, a new image is acquired and the object is gripped again. Then the image is compared to the first one (the empty working area) and the difference defines a point. The coordinates of the point in the image coordinate system are found. At this moment, four points in the robot coordinate system and four points in the image coordinate system are available. Therefore, the transformation matrix can be calculated. The camera is calibrated now, see Figure 6.4.

6.4.3 Perform a Special Action

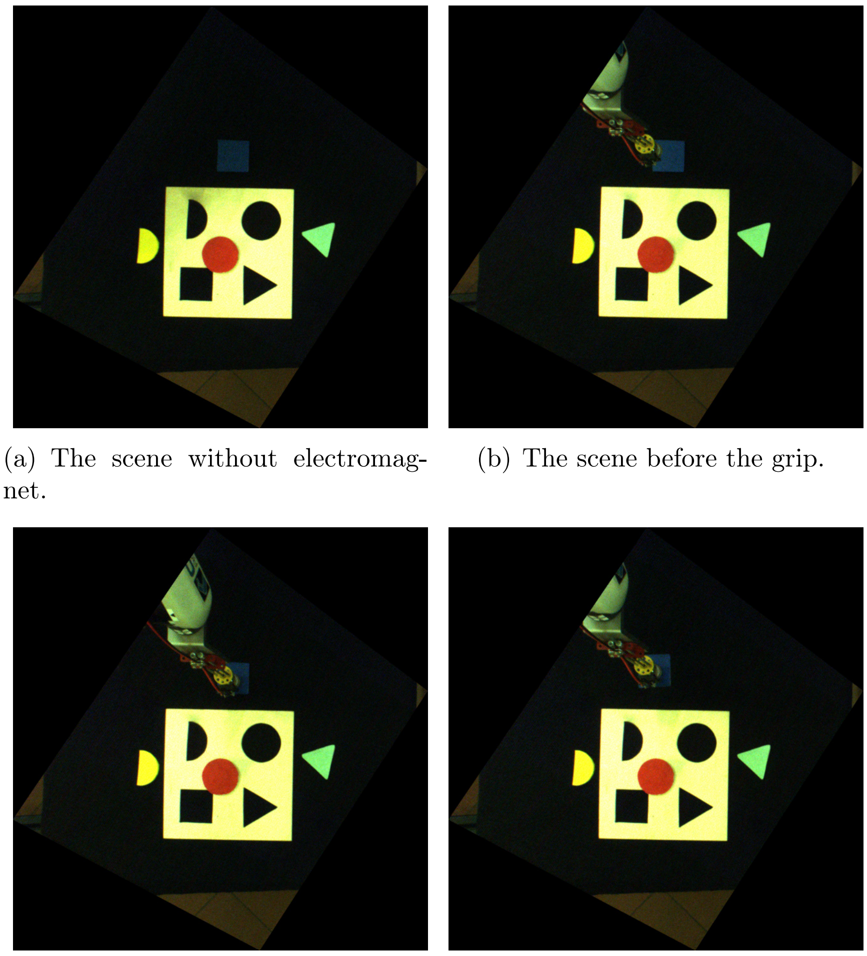
This is the most important method of the MATLAB module interface. It is used by other modules. It has to carry out an action in the given environment. As mentioned earlier, it does not matter whether it is an artificial environment in the simulator or it is a real in the demonstrator. The same interface is implemented by both the simulator and the demonstrator. These modules can be freely interchanged. The information concerning the action success is returned when it is performed. Three main and one auxiliary actions can be distinguished. The reason for implementing the auxiliary action is in demonstrator properties. The simulator does not need to perform this action. Here is the list of all action:

Absolute move. The electromagnet is positioned into a certain location. The coordinates are in the image coordinate system and are later converted to the robot coordinate system.

Grip. This action is responsible for a puzzle piece gripping. Moreover, the presence of an image feedback is worth mentioning. Two images are acquired during the action – the first before the grip and the second after the grip. The images are used in the late processing phase to detect a successful object grip. See Figure 6.5 for more details.

Release. The electromagnet releases a puzzle piece.

Alignment. This is the auxiliary action (it is not present in the 2D COSPAL simulator). All the simulator pieces are the same (just circles). The demonstrator puzzle contains the pieces of various shapes. A piece needs to be aligned so that it can be inserted into a hole.


(a) The scene without electromagnet.

(b) The scene before the grip.

(c) The scene during the grip.

(d) The scene after the grip.

Figure 6.5: Images acquired for the grip test purposes.

Summary

The MATLAB module for the COSPAL project was described in this chapter. This module is based on the original MATLAB-A465 interface which was extended so that the module could be incorporated into the other COSPAL project modules. The object-oriented design is discussed in Section 6.1. The following three sections dealt with each particular object of the object-oriented design. The camera object was described in Section 6.2, the robot object in Section 6.3 and the control object in Section 6.4.

As soon as the works on this module finished, the COSPAL project related tasks could be solved. The main part of the diploma work was done by accomplishing this module. Finally, next two tasks can be marked as done.

This concerns the “Image acquisition module” and the “Interface to shape sorter puzzle” tasks.

Chapter 7

Conclusion

This chapter summarizes the results of the diploma project. First, the diploma project's tasks are checked whether they have been accomplished. Then, the utilisation of the project's outcomes is discussed.

7.1 Tasks Accomplishment

In Chapter 2, the diploma project was decomposed into four subtasks. Each task covers different part of the work. All four tasks are investigated here.

Extension of the robot control system. The robot control system was modified. The upgrades affected both the TROL library and the MATLAB-A465 interface. All the modifications resulted in the ability of the MATLAB module for the COSPAL project to operate the electromagnet through the GPIO interface. The upgrades also had a positive side-effect. The usability of the robot in other projects has been increased. By simple modification of the current software, the full control of the GPIO port can be achieved.

Image acquisition module. This module is included in the MATLAB module for the COSPAL project. In the demonstrator environment, the images are acquired with a camera. Besides the unaltered images, some transformed images are required. This module takes care of this. The advantage of the applied approach is in the camera position independent acquisition of the images.

Interface to shape sorter puzzle. The demonstrator of the COSPAL project capable of playing shape sorter puzzle game was created. The demonstrator consists of simplified 2D version of the puzzle, the robotic arm equipped with the electromagnet and the camera that takes the images of the working area. The demonstrator control was implemented in the MATLAB module for the COSPAL project. Besides all this, the module is responsible for the system calibration.

Software methodology, documentation, testing. Chapter 5 and Chapter 6 are concerned with the design and the implementation of the demonstrator and the MATLAB module. The software and its documentation is available on an enclosed disc.

According to the above-mentioned, the author hereby declares all the goals of his diploma thesis to be accomplished.

7.2 Demonstrator Utilisation

The demonstrator started to be used for solving the tasks related to the COSPAL project. This was within the scope of the diploma project of Mr. Radim Krupička. His work was concerned with a certain module capable of object alignment, object tracing, object recognition and grip detection. Both the demonstrator and the 2D simulator can be used by that module. For more details, see the diploma thesis of Mr. Krupička [5].

The next important thing worse mentioning was the COSPAL project review meeting. It took place on the ground of the Center for Machine Perception, Faculty of Electrical Engineering, **C**zech **T**echnical **U**niversity in Prague (CTU) in September 2006. Within the frame of the meeting, the capability of the demonstrator was showed off. Some methods related to the COSPAL project were presented, such as manipulation abilities of the robot and learning abilities of the COSPAL system. This was all demonstrated on the shape sorter puzzle. The COSPAL project was successfully defended and can further continue.

In the end, the further COSPAL project development will be presented. The students at the CTU solve COSPAL project related tasks. They use resources developed during the works on this diploma project. They also modify and extend the work. One of the modifications can be seen in Figure 7.1.

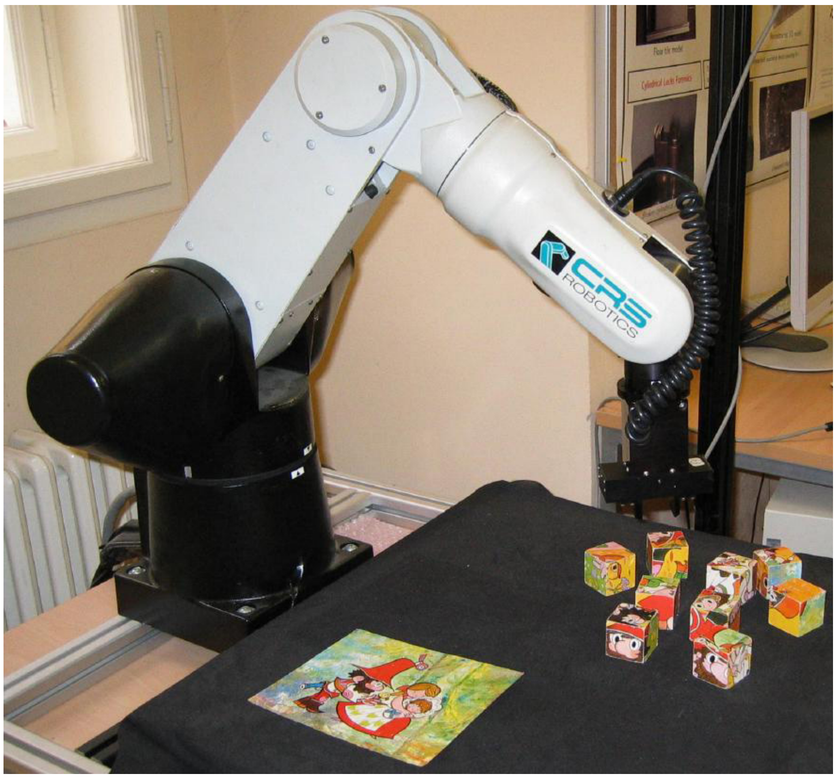


Figure 7.1: COSPAL project related task - cognitive robot puts the cubes together.

Bibliography

[1] CRS Plus Inc., Burlington, Ontario, Canada. *CRS Plus, A465, Robot manual*, 1993.

[2] CRS Plus Inc., Burlington, Ontario, Canada. *RAPL-II Programming Manual*, 1993.

[3] H.J.A. de Wolf and E. Fok. MATLAB Interface for CRS A465 Robot, Technical Report. Technical report, TU Delft, 1998.

[4] Erik Jonsson. The COSPAL 2D puzzle simulator v1.1, October 2004.

[5] Radim Krupička. Unsupervised Learning Based on Images, 2007.

[6] LiU. COSPAL Deliverable D1: Project Presentation. Technical Report IST-2003-004176, LiU, 2004.

[7] http://www.mathworks.com MATHWORKS. The Mathworks, Inc. 2007. MATLAB.

[8] Thomas Melzer. TROL TU Vienna Robot Control Library V2.2, User Manual. Technical report, TU Vienna, 1998.

[9] Milan Šonka, Václav Hlaváč, and Roger Boyle. *Image Processing, Analysis and Machine Vision.* Thomson Toronto Canada, third edition, 2007.

List of Figures

Appendix A

Contents of the Enclosed CD

A.1 Overview of the Contents of the Enclosed CD

The enclosed CD contains:

- System guide and code documentation.
- PDF file containing this diploma thesis.
- Project source codes and compiled binaries.

The contents of the individual directories:

gm_module – The main control module of the application is here.

graph_segmentation – This directory contains the segmentation algorithm written in "C" language.

hw_ctu – The contents of this folder present both the robot control and the image acquisition modules.

rl – The source codes of the reinforcement learning modules are stored here.

robot – This directory contains the source codes, documentation and compiled DLL libraries of both the TROL Robot Control Library and MATLAB Interface for CRS A465 Robot.

SEDUMI, stptool and yalmip – These folders incorporate external libraries required for the work.

A.2 List of the Created and Modified Files

The integral component of this diploma project are the files containing the source codes. The works on this project required the creation of the new files as well as the modification of the current ones. The enclosed CD contains some functional modules of the COSPAL project. A group of people has been engaged in the COSPAL project development, and thus the CD contains the works of many authors. All files referring to this diploma thesis contain author's mark *PJ*. The following list contains both newly created and modified items. Only the most important files are listed:

- */hw_ctu/@hw_ctu_camera* All files in this directory.
- */hw_ctu/@hw_ctu_control* All files in this directory.
- */hw_ctu/@hw_ctu_robot* All files in this directory.
- */hw_ctu/control_init.m*
- */robot/Matlab/Source/TRAPL.C*
- */robot/Matlab/Source/TSERVER.C*
- */robot/troldev/SOURCE/SEND.C*
- */robot/troldev/SOURCE/PCP/PCPTROL.C*