

VYŠŠÍ ODBORNÁ ŠKOLA, STŘEDNÍ ŠKOLA,
CENTRUM ODBORNÉ PŘÍPRAVY



ABSOLVENTSKÁ PRÁCE

Postprocesor robota pro metodu Laser
Shock Peening

Sezimovo Ústí, 2022

Autor: Marek Böhm



ZADÁNÍ ABSOLVENTSKÉ PRÁCE

Student: **Marek Böhm**
Obor studia: 26-41-N/01 Elektrotechnika – mechatronické systémy
Název práce: **Postprocessor robota pro metodu Laser Shock Peening**
Anglický název práce: **Robot post processor for Laser Shock Peening technique**

Zásady pro vypracování:

1. Seznamte se s metodami programování průmyslových robotických ramen.
2. Seznamte se s metodou Laser Shock Peening a identifikujte vstupy, výstupy a parametry této metody.
3. Navrhněte funkce rozšiřující postprocesor Vámi zvoleného CAM programu.
4. Naprogramujte rozšiřující funkce postprocesoru ve Vámi zvoleném vyšším programovacím jazyku a proveďte simulaci programu robotického ramene v CAM programu.
5. Ověřte a zdokumentujte funkci postprocesoru.
6. Absolventskou práci napište v anglickém jazyce.
7. Absolventskou práci vypracujte problémově ve struktuře odpovídající vědecké práci.


Doporučená literatura:

- [1] ŠVEJDA, M. Kinematika robotických architektur, (Diplomová práce), ZČU v Plzni, FAV, Plzeň, 2008.
[2] CORKE, P. *Robotics, Vision & Control*. New York: Springer Publishing, 2017, ISBN 978-3-319-54413-7.


Vedoucí práce: Mgr. Bc. Miroslav V. Hospodářský, VOŠ, SŠ, COP
Sezimovo Ústí
Odborný konzultant práce: Ing. Jakub Horáček, HiLASE centrum, Fyzikální ústav
AV ČR, v. v. i.
Oponent práce: Ing. Jan Kaufman, HiLASE centrum, Fyzikální ústav
AV ČR, v. v. i.

Datum zadání absolventské práce: **3. 9. 2021**

Datum odevzdání absolventské práce: **13. 5. 2022**


Mgr. Bc. Miroslav V. Hospodářský
(vedoucí práce)




doc. PhDr. Lenka Hrušková, Ph.D.
(ředitel školy)

V Sezimově Ústí dne 3. 9. 2021

Declaration

I declare that I have prepared my graduation thesis independently, and I have used only the materials (literature, projects, software, etc.) listed in the attached list.

In Sezimovo Ústí on 29.4.2022

Böhm

Signature

Acknowledgments

I want to thank my supervisor Mgr. Bc. Miroslav V. Hospodářský for the invaluable advice and help in creating this graduation thesis.

Anotace

Laser shock peening je proces povrchové úpravy používaný ke zlepšení fyzikálně-chemických vlastností (únavová životnost, odolnost proti korozi) kovových součástí. Laser shock peening vnáší pod povrch kovových materiálů zbytková napětí. Aplikace procesu laser shock peening v posledních letech rostou, a to především díky stále rostoucím energiím a klesajícím cenám laserových systémů s parametry vhodnými pro tento proces. Tato práce řeší problém, jak lze software RoboDK a jeho Python application user interface efektivně využít k vytváření programů robotických ramen pro proces laser shock peening. Problém je vyřešen úpravou postprocesoru RoboDK. Řešení umožňuje vytvářet programy pro robotická ramena speciálně uzpůsobené pro proces laser shock peening. Hlavní přínos práce je zjednodušení vytváření programů robotických ramen pro díly se složitou geometrií.

Klíčová slova: RoboDK; průmysloví roboti; průmyslová robotická ramena; zabránění kolizím; simulace robotických ramen; postprocesory robotických ramen.

Annotation

Laser shock peening is a surface treatment process used to improve the physico-chemical characteristics (fatigue life, corrosion resistance) of metallic components. Laser shock peening induces residual stress beneath the treated surface of metallic materials. Laser shock peening applications have been on the rise in recent years, mainly due to the ever-increasing energy and decreasing price of laser systems with parameters suitable for this treatment process. This graduation thesis seeks to solve the following problem: how can RoboDK with its Python application user interface be effectively used to generate robotic arm programs for laser shock peening applications. The problem is solved by modifying an existing RoboDK post processor. The created solution provides the possibility of generating robotic arm programs specially tailored for laser shock peening applications. The main contribution of this graduation thesis is to simplify the generation of robotic arm programs for parts with complex geometries.

Key words: RoboDK; industrial robots; robotic arms; collision detection; robotic arm simulation; robotic arm post processors.

Contents

List of Figures	xi
List of Tables	xiii
1 Introduction	1
2 Basics of industrial robotics	3
2.1 Classification of handling devices	4
2.2 Industrial robotic arms	5
2.2.1 Classification of industrial robotic arms and their structures	6
2.3 Industrial robotic arms programming	7
2.3.1 Offline programming	8
3 Laser shock peening	11
3.1 Laser shock peening overview	12
3.1.1 Laser shock peening process	12
3.1.2 Plasma, shock wave and residual stress generation	13
3.1.3 Transparent overlay and absorbent coating	15
3.1.4 Laser shock peening overlapping strategy	16
3.1.5 Laser system parameters affecting the LSP process	17
3.1.6 Laser shock peening applications	19
3.2 Laser shock peening station and laser sources	20
3.2.1 LSP station layout	20
3.2.2 Bivoj laser system	21
3.2.3 Litron LPY ST 7875-10 2HG laser	23
3.2.4 FANUC M-20iA/20M robotic arm	23
3.2.5 Electrical connection between controller and laser	24
3.2.6 LSP process example	26

4	RoboDK software	29
4.1	RoboDK overview	29
4.1.1	RoboDK features	30
4.2	RoboDK robotic arm library	31
4.3	RoboDK interface	31
4.3.1	RoboDK station	31
4.4	RoboDK API	32
4.5	RoboDK Add-ins	33
4.5.1	RoboDK Add-in for SolidWorks	33
4.6	RoboDK collision detection	35
4.7	RoboDK post processors	35
5	Post processor implementation	37
5.1	FANUC robotic arms programming specifics	37
5.1.1	FANUC Roboguide	37
5.1.2	FANUC robotic arm controller programming languages	38
5.1.3	Teach pendant program structure	39
5.1.4	Compiling a FANUC teach pendant program	40
5.2	Robot machining projects in RoboDK	41
5.2.1	Setting up a curve follow project in RoboDK	42
5.3	RoboDK API	43
5.3.1	Python API for RoboDK	44
5.4	RoboDK post processors	44
5.4.1	RoboDK post processor workflow	45
5.4.2	RoboDK preprocessed file	45
5.4.3	Selecting a post processor	48
5.4.4	Post processor example and post processor example output	48
5.5	FANUC R-30iA RoboDK post processor	51
5.5.1	Modifying the FANUC R-30iA post processor methods	51
5.5.2	Example output of the modified FANUC R-30iA post processor	54
6	Testing of robotic arm program	57
6.1	Simulation	57
6.2	Testing on the physical robotic arm	59
6.3	Results of simulation and testing	59

6.3.1 Results of simulation	59
6.3.2 Results of testing on a physical robotic arm	60
7 Conclusions and future work	61
7.1 Conclusions of the thesis and future work on thesis	61
Bibliography	63
Appendices	I
A Content of the enclosed CD/DVD	III
B Used software and software libraries	V
C Graduation thesis timetable	VII
D English-Czech glossary	IX

List of Figures

2.1 Classification of handling devices	4
3.1 Schematic configuration of laser shock peening process	13
3.2 Residual stresses formation with LSP	14
3.3 Laser shock peening strategy – one layer consisting of four sequences	16
3.4 Laser system Bivoj: Diode-pumped solid-state (DPSSL) laser	21
3.5 Schematic layout of LSP station and photography of LSP station	21
3.6 Spatial profile of laser beam at output of Bivoj 10 J amplifier	22
3.7 Temporal profile of laser beam at output of Bivoj 10 J	22
3.8 Litron LPY ST 7875-10 2HG laser at HiLASE centre	24
3.9 Block diagram of the electrical connection	25
3.10 Timing diagram for triggering of flashlamp and the Q-switch	25
3.11 FANUC M-20iA/20M industrial robotic arm and FANUC R-30iB controller	26
3.12 Network diagram of the robotic arm workcell	27
4.1 RoboDK version 5.2 running on Windows 10	32
4.2 SolidWorks RoboDK Add-in toolbar	33
4.3 SolidWorks RoboDK Add-in settings window	34
4.4 Example of a collision map	35
5.1 FANUC Roboguide workcell – user interface example	38
5.2 Example of a TP program with some typical program elements	40
5.3 Example of motion instruction	41
5.4 Curve follow project settings in RoboDK	43
5.5 RoboDK post processor workflow	45
6.1 CAD drawing of forging die	58
6.2 RoboDK workcell with forging die	58
6.3 LSP process performed on the forging die	60

List of Tables

3.1	Litron LPY ST 7875-10 2HG parameters	23
3.2	Experimental parameters of laser shock peening	26
6.1	Experimental parameters of LSP process on forging die	59

Chapter 1

Introduction

Laser shock peening (LSP) is a surface treatment process used to prolong the fatigue life of metallic components. LSP is successfully used in industrial applications, and there is a tremendous commercial interest in applying LSP. The layout of most modern LSP stations comprises a laser source and an industrial robotic arm holding the sample to be processed. One such LSP station is located at the HiLASE centre, Czech Republic. LSP is suitable for treating parts of complex geometry, such as forging dies and cutters. However, this advantage comes with a new challenge: to program the robotic arm in such a way as to ensure an optimal LSP process on parts with complex geometry. These programs can be generated by using Computer-Aided Manufacturing (CAM) software. A critical feature of every CAM software is the post processor of the CAM software. A post processor defines the vendor-specific rules a robotic arm program must follow. In other words, post processors serve as tools to convert a simulation to vendor-specific robotic arm programs [1].

The potential of high-energy pulsed laser systems to create plastic deformation was first discovered by White in 1963 in the USA [2]. The confined ablation mode in air was first recognised by Neuman in 1964 [3]. During the 1970s, research into the possible applications of LSP continued. During the next decade, in the 1980s, a team centred around Remy Fabbro at the Ecole Polytechnique made significant theoretical and experimental contributions to LSP [4]. At the same time, researchers at the Battelle Memorial Institute in Columbus, Ohio, developed a prototype pulsed laser to demonstrate the industrial viability of LSP. The 1990s marked the decade when LSP saw its first commercial applications. Notably, General Electric Aviation used LSP to solve Foreign Object Damage (FOD) on fan blades of turbofan jet engines [5]. The 2000s and 2010s saw the rise of companies performing LSP commercially [6].

The goals of this graduation thesis can be summarised into the following points:

1. create a simulation of a LSP process in CAM software,
2. develop a post processor for the LSP process,
3. generate a program for the robotic arm controller from simulation,
4. test simulation of LSP process on physical robotic arm.

This graduation thesis is divided into the following chapters: [Chapter 2](#) contains the basics of industrial robotics. [Chapter 3](#) acquaints the reader with the LSP process. It describes the research topic, related work, and the experimental setup. [Chapter 4](#) deals with the CAM program used in this graduation thesis, RoboDK. The actual implementation of the post processor in the programming language Python is the focus of [Chapter 5](#). In [Chapter 6](#), the post processor for the LSP process is tested on an actual industrial robotic arm setup. Finally, the closing [Chapter 7](#) contains the conclusions and future work.

The HiLASE research centre is an infrastructure focused on laser research and development, located in Dolní Břežany, Czech Republic. HiLASE's Industrial laser applications program (ILA) focuses on LSP, laser induced damage threshold, and laser micromachining technologies. ILA is equipped with several experimental stations, including an LSP station. The author of this graduation thesis is employed as a control system engineer in the laser shock peening group of the ILA program. In addition, the author is responsible for carrying out LSP experiments for industrial partners and universities. The author's other tasks include programming the LSP station's robotic arm, integrating different laser sources with the LSP station's robotic arm and constructing and putting into operation subsystems of the LSP station.

Chapter 2

Basics of industrial robotics

Excerpt from the play Rossum's Universal Robots (R.U.R.).

In the introductory scene, Helena Glory is visiting Harry Domin, the director general of Rossum's Universal Robots, and his robotic secretary Sulla.

Domin: Sulla, let Miss Glory have a look at you.

Helena (stands and offers her hand): Pleased to meet you. It must be very hard for you out here, cut off from the rest of the world [the factory is on an island].

Sulla: I do not know the rest of the world Miss Glory. Please sit down.

Helena (sits): Where are you from?

Sulla: From here, the factory.

Helena: Oh, you were born here.

Sulla: Yes I was made here.

Helena (startled): What?

Domin (laughing): Sulla isn't a person, Miss Glory, she's a robot.

Helena: Oh, please forgive me ...

Karel Čapek

Rossum's Universal Robots (R.U.R.)

This chapter gives a short introduction to the domain of industrial robotics. This chapter deals with the classification of industrial robotic arms, industrial robotic arms programming, and offline programming of robotic arms. A basic understanding of industrial robotics is necessary to understand the implementation of the post processor.

2.1 Classification of handling devices

Figure 2.1 gives a basic idea of the different types of handling devices.

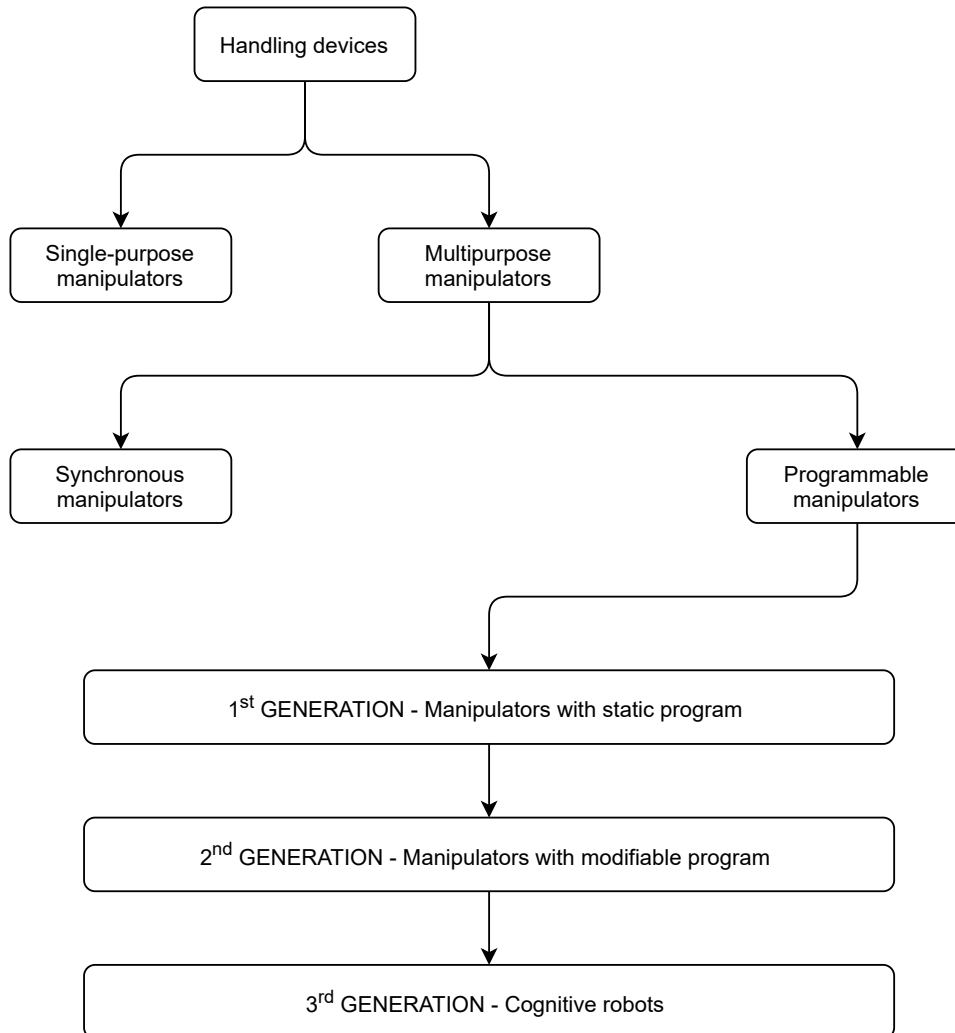


Figure 2.1: Classification of handling devices 7

- single-purpose manipulators – They are characterised by limited movement possibilities, level of control suitable for the application, design, and drives corresponding to the equipment being operated and the technology being used,
- multipurpose manipulators – These manipulators are multipurpose with the possibility of adapting to different technologies. The choice between single-purpose and multipurpose industrial robots and manipulators is based on the overall assessment of the technology, workplace, etc., and must respect both the technical and economic aspects,

- synchronous manipulators – The control is performed continuously by a human. These handling mechanisms are amplification devices for amplifying the force and motion variables on based on stimuli triggered by a human. They are independent of the machine being operated. The manipulator and the human form a closed control loop. These devices transmit human commands remotely. This possibility of remote control was used to a great extent for scientific, medical and military purposes in the past. Even today, some operations are carried out indirectly using miniature manipulators,
- programmable manipulators – They are controlled by a programming unit,
- programmable manipulators with fixed program – The program does not change during the operation of the handling mechanism, it is fixed, the programming unit is of simple design,
- programmable manipulators with modifiable programs – They have the possibility of switching or selecting the program, usually according to the scene in which the manipulators are currently in. These are usually devices with adaptive control. They are currently at the cutting edge of design and are called industrial robotic arms,
- cognitive robots – These robots equipped with the ability to perceive and think rationally [7].

2.2 Industrial robotic arms

A robot is an automatically or computer-controlled integrated system capable of autonomous goal-oriented interaction with the natural environment according to instructions from a human. This interaction consists of:

- in the perception and recognition of that environment,
- manipulating objects or moving around in this environment.

Industrial robotic arms are of more complex design than other handling devices and differ from other handling mechanisms primarily in the level of control. Industrial robotic arms are characterised by the following features:

- manipulation capability, i.e., grasping and moving objects,
- behavioural autonomy, i.e., a complex sequence of actions performed automatically according to a specific program. Of particular importance is the case where this program is not fixed (given by design, as in the case of classical automatic controllers) but modifiable,
- controlled either by a human or automatically by its own controller. This makes it different from, e.g., synchronous manipulators, which amplify and transmit motion commands remotely directly from the human, who is an integral part of the system,
- versatility in the sense of 'multipurpose', not 'omnipotence'. The devices do not serve a single purpose but for several, sometimes quite diverse, purposes. This is related to the possibility of changing the program,
- the existence of a link with the environment (perception). In addition to simple mechanical (touch) electromagnetic sensors, more complex systems can also include visual (using a television camera) and acoustic feedback,
- the spatial concentricity of the individual components (integration) is preferably (but not necessarily, if one of the components is a computer) into a single object. The consequence is, among other things, easy transportability. In some cases, the system may be required to be mobile [7].

2.2.1 Classification of industrial robotic arms and their structures

Industrial robotic arms can be classified according to various criteria: the number of degrees of freedom, kinematic structure, drives used, workspace geometry, motion characteristics, control method, or programming method. According to the above-mentioned criteria, several types of robotic arms are distinguished:

Number of degrees of freedom

- universal robotic arms – six degrees of freedom,
- redundant robotic arms – more than six degrees of freedom,
- deficient robotic arms – less than six degrees of freedom.

Kinematic structure

- serial robotic arms – open-loop kinematic chain,
- parallel robotic arms – closed-loop kinematic chain,
- hybrid robotic arms – combining both types of kinematic chains.

Type of drives

- electric,
- hydraulic,
- pneumatic.

Currently, industrial robotic arms with electric drives predominate in numbers. If high loads are required, hydraulic drives are used and for high speeds pneumatic drives are preferred.

Workspace geometry

- Cartesian – robotic arm equipped with three perpendicular linear axes,
- cylindrical – robotic arms equipped with two linear axes and one rotary axis,
- spherical – robotic arms equipped with two rotary axes and one linear axis,
- SCARA – Selective Compliant Articulated Robotic Arm, also equipped with with two rotary axes and one linear axis but in a different configuration than a spherical robotic arm [7].

2.3 Industrial robotic arms programming

Before we dive into the programming of industrial robotic arms, let us revise a few concepts. A workcell represents a robotic arm and a collection of machines or peripherals. A single robotic arm controller is responsible for controlling the various appliances of a workcell. A robotic arm end effector is a peripheral placed at the end of the robotic arm. The end effector represents the last link of the robotic arm. According to the

application, end effectors can be grippers, welding devices, spray guns, or grinding and deburring machines [8].

The robotic arm controller is equipped with a so-called interface software. The interface software makes it easier for the user to program the robotic arm. Two basic information need to be programmed into the robotic arm:

- position data – i.e., where the robotic arm has to move to,
- procedure – i.e., what action the robotic arm will perform at the specific position.

A robotic arm position can be taught in several different ways:

- positional commands – the programmer inputs the position data using a graphical user interface (GUI) or text-based commands,
- online programming, which is further divided into three types:
 - pendant – the pendant is a handheld control and programming unit attached to the robotic arm controller,
 - lead-by-the-nose – the robotic arm is de-energized and moved along the required positions by hand while the robotic controller saves them into memory,
 - direct control by CAM program – the robotic arm is moved along the required positions from an external PC using a CAM program.
- offline programming – the robotic arm cell including the robotic arm and machines are represented in a specialised software. The robotic arm can be moved in the software and brand-specific applications for the robotic arm controller can be created [9].

2.3.1 Offline programming

Robotic arms from different manufacturers have incompatible interfaces. Robotic arm programming languages evolve slowly, and robotic arm manufacturers offer backwards compatibility. For example, FANUC uses not one, but two programming languages for their controllers:

- TP language – suitable for programming position data,

- KAREL language – high-level language based on Pascal, suitable for programming logical constructs [10].

Offline programming (OLP) refers to a programming method in which the robotic arm is programmed outside the production environment. OLP helps to eliminate production downtime and allows studying multiple scenarios of a robot cell before setting up the production cell. OLP also aids in predicting mistakes made in designing a workcell. The OLP workflow can be summarised into the following steps:

1. creating/importing a robotic arm cell that mimics the real life workcell,
2. importing a CAD file representing the machined part,
3. programming (graphical or textual) and automatic generation of robotic arm path,
4. simulation and validation – collision detection, joint violations etc.,
5. reporting – efficiency, productivity, cycle time etc.,
6. translating the robotic arm program for a specific controller using a post processor,
7. executing the robotic arm program in an actual workcell [11].

Chapter 3

Laser shock peening

Since the early Middle Ages, mankind has always struggled to increase the hardness and flexibility of metals, to improve the durability and reliability of metal tools and parts. From the Damascus steel smiths through the swordsmiths of medieval Japan, the method of cold-working metal by the application of precise mechanical pressure such as hammer blows has a glorious history.

In modern times, this concept has reached its apex with industrial surface hardening using shot peening, where thousands of small lead shots are fired at the metal surface, compressing and hardening the surface, improving the durability. With the advent of high power laser systems, an improved process has been developed, the process of laser shock peening.

*Samuel Zhrdne
HoloOr company*

Laser shock peening (LSP) is a surface treatment process used to improve the mechanical and physico-chemical characteristics (fatigue life, corrosion resistance, etc.) of metallic components. LSP induces residual stress beneath the treated surface of metallic materials. This residual stress is generated by high magnitude shock waves. A high magnitude of shock waves is achieved by the confined plasma generated at the metal surface by means of a high-intensity laser beam with a pulse duration in the tens of nanosecond range. Compared to traditional processes such as shot peening (SP), an LSP process can

induce compressive residual stresses up to 1 mm in depth, which is approximately four times deeper than SP. LSP can be applied to various metallic components, such as:

- aluminium alloys,
- titanium alloys,
- cast irons,
- nickel-based superalloys.

Firstly, this chapter deals with the physical and mechanical mechanisms of laser shock peening. Secondly, it focuses on the parameters of laser shock peening, namely:

- laser power density,
- pulse shape and duration,
- laser wavelength,
- laser spot size and geometry,
- thermal protective coating, and confining overlay,
- coverage ratio of impacts [12].

Lastly, this chapter describes the experimental equipment used for LSP at the HiLASE centre.

3.1 Laser shock peening overview

3.1.1 Laser shock peening process

The configuration of an LSP process with a metallic component is shown in Figure [3.1]. An intense pulsed laser shock beam is focused onto a metal surface for a brief period of time (10 ns to 100 ns). The heated zone is vaporised and transformed to plasma by ionisation (the temperature of plasma is over 10 000 °C). The plasma is under high pressure, which propagates through the material via shock waves. Two modes of LSP exist:

- direct ablation mode,
- confined ablation mode.

The direct ablation mode refers to the interaction of plasma with metal without coating and confinement [6]. Plasma pressure of tenths of a GPa is achieved using direct ablation mode. Higher pressure of 1 GPa to 5 GPa can be obtained using the confined mode. The confined ablation mode is known not only to increase the peak pressure of plasma by a factor up to ten, but also increases the duration of plasma by a factor of three in comparison with the direct ablation mode. In the confined mode, the metal surface is usually coated with an opaque material such as black paint or aluminium foil and confined by a material transparent to the laser radiation such as water or borosilicate glass. A stronger pressure pulse results in a higher magnitude of compressive residual stress at a deeper depth [13].

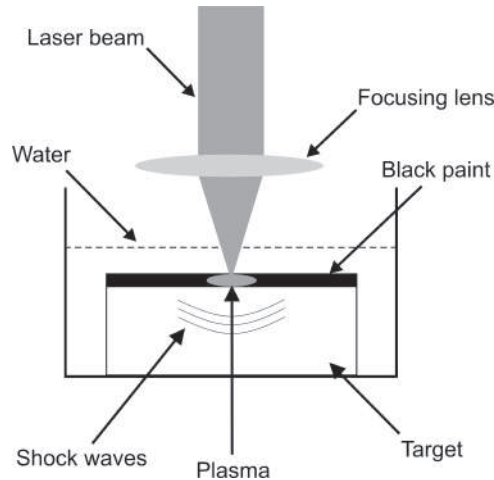


Figure 3.1: Schematic configuration of laser shock peening process [14]

3.1.2 Plasma, shock wave and residual stress generation

When the laser power density rises to several GW cm^{-2} , the surface of the material (or the absorbent coating – if used) is heated and vaporised and forms plasma with pressure of several GPa on the material's surface. The plasma absorbs the laser energy for the duration of the laser pulse. The plasma expands rapidly and generates a pressure pulse on the surface of the material. As a consequence, a high-amplitude shock wave propagates through the material. When using the confined ablation mode with a transparent overlay,

the plasma is held between the material and the transparent overlay, increasing the plasma's magnitude and amplitude. According to the above-mentioned description of the confined ablation mode, the LSP process can be regarded as a two-step method:

1. uniaxial compression of the irradiated area and dilation of the surface layer is caused by the rapid plasma expansion,
2. the surrounding material reacts to this deformation and generates a compressive stress field.

Before the reaction of the surrounding zones excluded from LSP, during the actual LSP process, uniaxial compression is induced in the direction of the shock wave propagation and a tensile state is induced in the plane parallel to the surface. After the reaction of the surrounding zones, a compressive stress field is generated in the affected zone, and a tensile stress field is created in the underlying layers of the material. The compressive stress field causes deformation in the material. The deformation is plastic as long as the peak dynamic stresses of the shock waves within the material are above the dynamic yield strength of the material. The process of compressive residual stress generation is shown in Figure 3.2 [15].

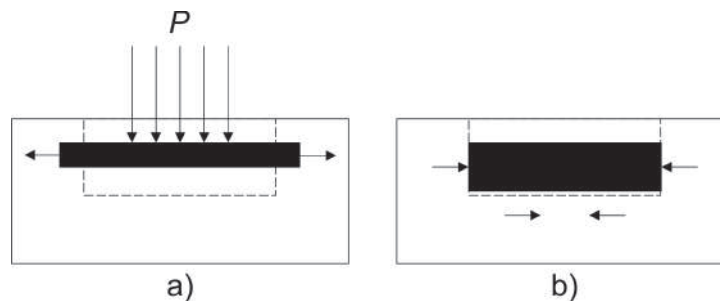


Figure 3.2: Residual stresses formation with LSP:(a) stretching of impact area during interaction, (b) recovery of material after laser pulse has elapsed [15]

The knowledge of the temporal and spatial profile of the confined plasma pressure provides insight into the optimisation of the LSP process. One of the simplest models for estimating plasma pressure was developed by Fabbro [15]. It is based on the physical and mechanical behaviour of the laser-induced plasma and describes the LSP process in three steps:

1. the laser pulse hits the material, and the confined plasma expansion causes shock waves to spread through the material,

2. after the laser pulse has elapsed, the plasma cools down adiabatically, but the pressure is still present over a period approximately twice as long as the laser pulse duration,
3. the adiabatic cooling continues, but the pressure of the plasma is already too low to drive shock waves deeper into the material.

Assuming the plasma to be a perfect gas, the peak plasma pressure for water-confined ablation mode then holds:

$$P = 0.01 \sqrt{\frac{\alpha}{2\alpha + 3}} Z I_0 \quad (3.1)$$

where:

- P – peak plasma pressure [GPa],
- I_0 – laser power density [GW cm^{-2}],
- α – efficiency of the interaction [-],
- Z – reduced shock impedance between the material and the confining medium [$\text{g cm}^{-2} \text{s}^{-2}$]

Based on Equation [3.1](#), we can assert that in the water-confined ablation mode, the peak pressure is directly proportional to the square root of the incident laser power density [15](#).

3.1.3 Transparent overlay and absorbent coating

In the confined mode, the material is covered with two layers. The material surface is first coated with a nontransparent (absorption) overlay. The second layer is a transparent (confinement) overlay placed on the absorption layer.

The transparent overlay aims to prevent the laser-induced plasma from expanding freely from the metal surface. The transparent overlay traps the expanding plasma on the metal surface. As a consequence, the plasma pressure rises higher than without the transparent overlay applied. Without the transparent overlay, the plasma would absorb the incident laser energy, but the laser energy would not be converted efficiently into a pressure pulse that induces compressive residual stresses in the material. The

transparent overlay is placed on top of the absorbent coating. Any transparent material such as water, glass, fused quartz, and acrylic can be used as a transparent overlay, but a thin layer of water flowing over the coated surface is usually used because it is the easiest to apply [16].

The metal surface is usually laminated with a nontransparent material such as black paint, black vinyl tape, or aluminium foil. The nontransparent coating serves two purposes:

- absorption coating – increasing the absorption of the incident laser energy and thus increasing the shock wave intensity,
- thermal protective coating – protecting the material surface from laser ablation and melting (the thermal effect is limited only to the protective coating) and thus preventing tensile strain from developing on the material's surface [17].

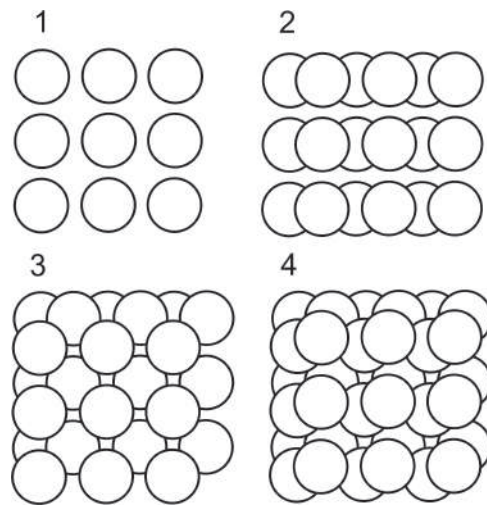


Figure 3.3: Laser shock peening strategy – one layer consisting of four sequences [14]

3.1.4 Laser shock peening overlapping strategy

Overlapping of successive LSP processes (i.e., individual laser impact areas) is used to cover large areas with LSP. The coverage ratio is defined as the ratio between the overlapping area and the impact spot size for two successive LSP processes. An increase in the coverage ratio increases the plastically affected depth. Overlapping successive LSP processes results in a comparatively uniform distribution of compressive residual stress across the overlapping regions.

During the peening process, a pattern is created made of individual laser pulses. Generally, multiple sequences of the pattern that are gradually shifted across the sample are used to create a layer, as seen in Figure 3.3. The pattern shifting ensures a more homogeneous residual stress distribution. The disadvantage of this strategy is that the protective coating needs to be replaced in between sequences 18.

3.1.5 Laser system parameters affecting the LSP process

When performing a LSP process, the most challenging task is the choice of optimal LSP parameters (laser power density, laser spot size, absorbent coating) for the given material. If these parameters are chosen incorrectly, high tensile residual stresses can be induced on the material's surface, which can worsen the mechanical performance of the component 19.

Laser system for laser shock peening overview

Choosing an appropriate laser system is the first step in fulfilling the LSP process requirements. In LSP laser systems, the focus is mainly on the following parameters:

- pulse energy: 1 J to 100 J/pulse,
- pulse duration: 10 ns to 100 ns,
- laser wavelength: three wavelengths are used for the LSP process most commonly:
 - 1064 ns (near-infrared),
 - 532 ns (green),
 - 355 ns (ultraviolet).
- sufficient repetition rate in the tens of hertz range.

The laser pulse energy influences the laser power density and therefore the magnitude of surface residual stress. The laser pulse duration and temporal shape affect the breakdown threshold of the laser power density. A shorter laser pulse with a short rise time usually results in a higher power density and higher peak pressure. The wavelength of the laser system influences the interaction between the laser pulse, absorption and confinement overlay and the material surface 15.

Laser spot size and geometry

One of the advantages of LSP is that the laser spot size and geometry can be adjusted to suit different applications. Laser spot sizes used for LSP vary between hundreds of micrometres and 6 mm and usually have a round or square profile. The usable laser spot sizes are restricted mainly by the incident laser power density. The main difference between using a laser spot with a small diameter compared to a laser spot with a larger diameter lies in the way the shock waves expand in the material:

- shock waves originating from a laser spot with a small diameter (r) expand like a sphere and attenuate at a rate of $1/r^2$,
- shock waves originating from a laser spot with a larger diameter (r) expand like a planar front and attenuate at a rate of $1/r$.

Consequently, the planar shock wave propagates further into the material, and the residual stress reach deeper than the spherical shock wave [20].

The magnitude of residual stress usually reaches its maximum at the surface and decreases with depth. This rule is valid for square laser spots. which minimize the instability of the residual stresses at the centre of the laser spot. On the other hand, when using a circular laser spot, the residual stresses at the centre of the laser spot are unstable due to the complicated interactions of shock waves in this region [21].

Laser power density and wavelength

The magnitude of the laser power density plays a pivotal role in determining the magnitude of the plasma pressure. Moreover, the magnitude of the surface residual stresses is directly proportional to the magnitude of the plasma pressure. It is tempting to assume that by just increasing the laser power density, we will obtain higher surface stresses and thus a better quality LSP process. In reality, we are limited by several effects accompanying the laser-matter interaction:

- dielectric breakdown of air can occur if the magnitude of the laser power density is too high. The dielectric breakdown is the generation of plasma not on the material surface, but in the area in front of it. This plasma absorbs the incoming laser pulse energy, thus limiting the energy to generate a shock wave in the material and affects the LSP process negatively,

- when the magnitude of laser power density is higher than a certain threshold, the residual stresses increase in depth but decrease at the surface because of surface release waves,
- another consequence of increasing the laser power density above a certain threshold is the saturation and scattering of the corresponding pressure. The saturation of pressure is caused by the confining overlay breakdown phenomenon. The breakdown of the confining overlay has two negative consequences:
 - the pressure duration is shortened,
 - the peak pressure is saturated.

As mentioned earlier, the three most commonly used wavelengths for LSP are near infrared, green, and ultraviolet. The near infrared wavelength has the highest absorption coefficient in the confining overlay and a high breakdown threshold. By contrast, the green wavelength has the lowest absorption in the confining overlay and a lower breakdown threshold. The pressures produced by the laser pulses have similar profiles regardless of the wavelength [22].

Laser pulse temporal profile and duration

Laser systems with parameters suitable for LSP usually deliver two types of temporal shapes:

- Gaussian,
- short rise time (SRT).

It was observed that SRT pulse shapes produce higher pressure on the metal surface and have a higher breakdown threshold than Gaussian pulse shapes. The pressure pulse duration is directly proportional to the laser pulse duration. This fact is advantageous when it is possible to change the laser pulse duration of a laser system [23].

3.1.6 Laser shock peening applications

Nowadays, LSP is heavily commercialised, and hundreds of patents regarding LSP have been issued. The industry leaders are General Electrics, LSP Technologies and Curtis Wright Surface Technologies. LSP is used in the following applications:

- increase of fatigue life and fatigue strength of structures [24],
- strengthening of thin sections [25],
- shaping or straightening of parts (laser peen forming).

LSP is expected to expand from current high-value, low volume parts to more mass-produced parts in the future, due to the steady advancements in laser systems suitable for LSP [26].

3.2 Laser shock peening station and laser sources

The HiLASE research centre is an infrastructure focused on laser research and development, located in Dolní Břežany, Czech Republic. HiLASE's Industrial laser applications program (ILA) focuses on LSP, laser induced damage threshold (LIDT), and laser micro-machining technologies. ILA is equipped with several experimental stations, including an LSP station.

The LSP station uses two laser sources:

- Bivoj laser system,
- Litron LPY ST 7875-10 2HG laser.

A laser beam distribution system (LBDS) distributes the beam from the laser laboratory located on the ground floor to the experimental laboratory located on the 1st floor. The Bivoj laser system is a Diode Pumped Solid State Laser (DPSSL) based on Yb-doped gain media with a laser wavelength of 1030 nm capable of delivering energy pulses up to 8 J at a 10 Hz repetition rate. The beam shape at the output of the Bivoj laser is a square flat-top pulse. An overview of the Bivoj laser system is shown in Figure [3.4]. The two main sections of the Bivoj laser are the front-end and the 10 J amplifier. The second laser source available at the LSP station is the Litron LPY ST 7875-10 2HG laser.

3.2.1 LSP station layout

The layout of the LSP station is shown in Figure [3.5]. The laser beam enters the LSP station through the LBDS output node and continues to the optical table, where it is

redirected and focused on the target. The target itself is mounted on the robotic arm. The laser beam position is fixed, so the robotic arm needs to be moved to direct the laser to the sample.

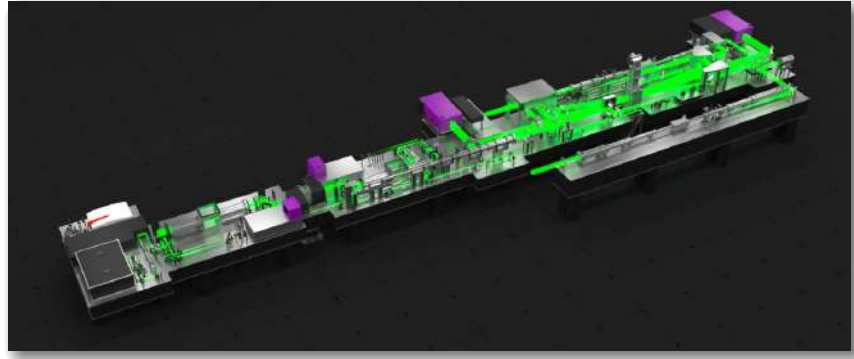
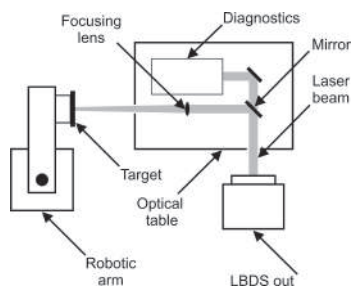


Figure 3.4: Laser system Bivoj: Diode-pumped solid-state (DPSSL) laser 27



(a) Schematic layout of LSP station



(b) Photography of LSP station

Figure 3.5: (Description of images from left to right) (a) schematic layout of LSP station, (b) photography of LSP station 14

3.2.2 Bivoj laser system

Bivoj front-end

The front-end starts with a fibre-based section, consisting of a fibre oscillator, a fibre amplifier, and a temporal pulse shaper with 125 ps resolution. The output of the fibre front-end is fed to the first booster amplifier, which is regenerative and increases the energy to 30 mJ and reduces the repetition rate to 10 Hz. The second booster amplifier works in a multi-pass regime and increases the pulse energy to approximately 100 mJ.

The repetition rate can be switched between 1 Hz and 10 Hz. The beam shape at the front-end output is $8 \times 8 \text{ mm}^2$ square flat-top.

Bivoj 10 J amplifier

The 10 J amplifier is the first-stage main amplifier based on cryogenically cooled multislabs technology. The principal component of the amplifier is the amplifier head, where the gain media are stored and cooled by gaseous helium to the temperature of about 150 K. The laser beam from the front-end is enlarged to $22 \times 22 \text{ mm}^2$ and sent to the 10 J amplifier, where it increases its pulse energy from approximately 100 mJ to approximately 8 J at 10 Hz. The spatial profile of the laser beam at the output of the Bivoj 10 J amplifier is shown in Figure 3.6 and the temporal profile is shown in Figure 3.7 [28].

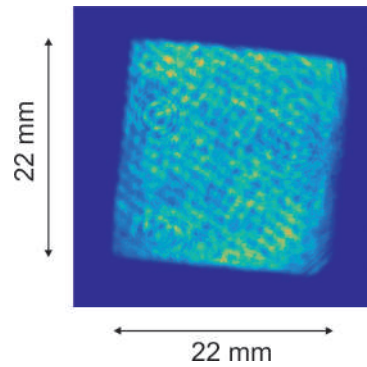


Figure 3.6: Spatial profile of laser beam at output of Bivoj 10 J amplifier and approximate dimensions [18]

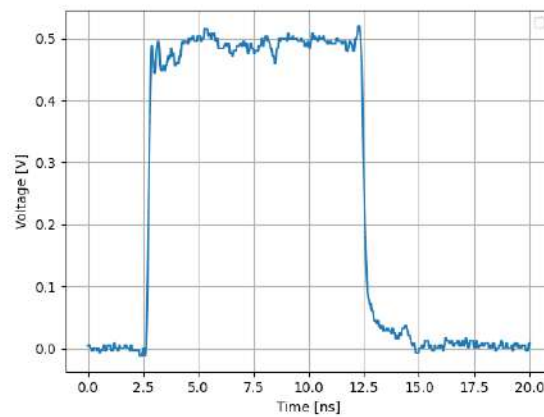


Figure 3.7: Temporal profile of laser beam at output of Bivoj 10 J amplifier measured with InGaAs photodetector [18]

3.2.3 Litron LPY ST 7875-10 2HG laser

The second laser source available at the LSP station is the Litron LPY ST 7875-10 2HG laser, a tabletop laser located directly in the experimental laboratory next to the LSP station. This laser is a pulsed Q-switched Nd:YAG laser suited for industrial or research applications. The Litron LPY ST 7875-10 2HG laser comes with a super-gaussian resonator. The most important parameters of this system are highlighted in Table 3.1 and the laser system is shown in Figure 3.8 [29].

Table 3.1: Litron LPY ST 7875-10 2HG parameters [30]

Parameter	Value
Repetition Rate [Hz]	10
Output Energy [mJ]	
1064 nm	3500 ^a
532 nm	1750
Beam Diameter [mm]	15
Beam Divergence [mrad]	<0.5 ^b
Pulse Width @1064 nm [ns]	10-12
Pointing Stability [μ rad]	25 ^c
Timing jitter [ns]	<0.5 ^d

^a Peak-to-peak Energy – 99 % of pulses.

^b Full angle for 90 % of the energy.

^c Half angle.

^d Jitter is measured concerning the external Q-switch trigger input.

3.2.4 FANUC M-20iA/20M robotic arm

The FANUC M-20iA/20M is a 6-axis universal industrial robotic arm with a maximum load capacity of 20 kg and a maximum reach of 1811 mm. The repeatability of the robotic arm is ± 0.08 mm. The FANUC M-20iA/20M is a multipurpose robotic arm and can be used for various applications such as assembling, packaging and machining [31].

The robotic arm is coupled with a FANUC R-30iB controller. The R-30iB controller is equipped with a FANUC AIF01A PLC interface module. The interface module is

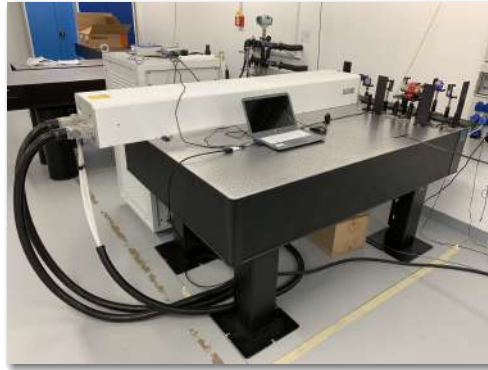


Figure 3.8: Litron LPY ST 7875-10 2HG laser at HiLASE centre

connected to the following expansion modules:

- AID16L – 16 digital inputs,
- AOD16D – 16 digital outputs,
- ADA02A – 2 analog outputs [32].

The FANUC AIF01A PLC interface module and its expansion modules are connected to the Bivoj and Litron lasers via digital inputs and outputs. In the case of the Bivoj laser system, the expansion modules are linked to a fast flipper located in the Bivoj laboratory. The fast flipper either redirects the laser beam into a cooled beam dump or lets it pass through the LBDS into the LSP station. The Litron laser is coupled to the robotic arm controller via an external trigger.

3.2.5 Electrical connection between FANUC R-30iB controller and Litron LPY ST 7875-10 2HG laser

A block diagram of the electrical connection between FANUC R-30iB controller and Litron LPY ST 7875-10 2HG laser is shown in Figure 3.9. The robotic arm PLC sends out a *ROBOT GATE* digital output signal which acts as a gating signal for the *EXTERNAL TRIGGER INPUT* signal. The *EXTERNAL TRIGGER INPUT* signal is generated by the microcontroller unit (MCU). The *LAMP SYNC OUTPUT* signal is fed back to the MCU and to the delay generator. The delay generator creates a delayed *Q-SWITCH TRIGGER* signal which is inputted into the *D/A* input of the laser controller. The *Q-SWITCH TRIGGER* signal is delayed with respect to the *EXTERNAL TRIGGER*

INPUT signal by t_{QSW} . The value of t_{QSW} is fine-tuned with the help of the delay generator. The timing diagram for internal triggering of the flashlamps and external triggering of the Q-switch is shown in Figure 3.10, where:

- t_{BT} – intra-cavity build up time (typically 50 ns to 100 ns),
- t_{QSW} – timed by external electronics with respect to the external lamp trigger input (typically 100 μ s to 500 μ s for maximum output)

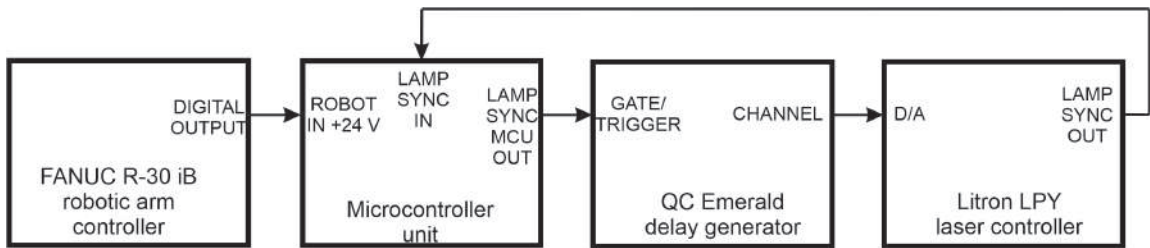


Figure 3.9: Block diagram of the electrical connection between FANUC R-30iB controller and Litron LPY ST 7875-10 2HG laser

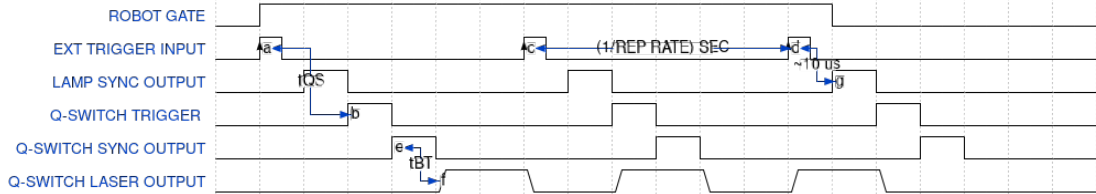


Figure 3.10: Timing diagram for external triggering of flashlamp and internal triggering of the Q-switch, where t_{BT} is the intra-cavity build up time (typically 50 ns to 100 ns) and t_{QSW} is the internally timed Q-switch delay (typically 100 μ s to 500 μ s for maximum output)

The FANUC M-20iA/20M robot and FANUC R-30iB controller are shown in Figure 3.11 [33]. The electrical connection also allows to change the repetition rate of the laser (1 Hz or 10 Hz).

Network connection of robotic arm controller

The robotic arm controller must be reachable from the computer running the RoboDK application described in Chapter 4. In the LSP station, this is ensured by having the controllers and computer on the same local network as depicted in Figure 3.12.

3.2.6 LSP process example

The following experimental parameters describe the improvement of cavitation erosion resistance by LSP carried out at the HiLASE centre. This process is developed by HiLASE center and SIGMA GROUP Plc. for the improvement of cavitation erosion resistance of pump blades. The parameters chosen for this experiment are shown in Table 3.2

Table 3.2: Experimental parameters of laser shock peening for improvement of cavitation erosion resistance

Parameter	Value
Laser source	Bivoj laser system
Wavelength [ns]	1030
Repetition rate [Hz]	10
Pulse energy at sample [mJ]	4500
Beam size at sample [mm ²]	9
Pulse width @1030 nm [ns]	10
Power density at sample [GW cm ⁻²]	5



(a) FANUC M-20iA/20M industrial robotic arm



(b) FANUC R-30iB controller with teach pendant

Figure 3.11: (Description of images from left to right) (a) FANUC M-20iA/20M industrial robotic arm, (b) FANUC R-30iB controller with teach pendant 33

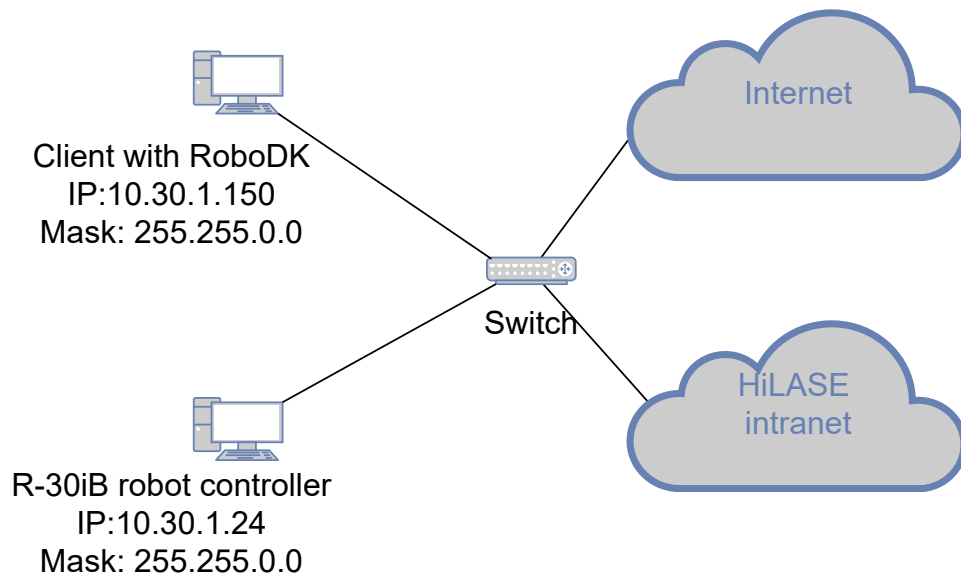


Figure 3.12: Network diagram of the robotic arm workcell

Chapter 4

RoboDK software

If you drive a car, it makes little difference what brand it is: all cars are driven in essentially the same way. The same applies to computers. If you have a Windows PC, the user interface won't be affected by your computer hardware. This is definitely not the case for industrial robots.

Albert Nubiola
CEO at RoboDK

This chapter gets the reader acquainted with the RoboDK software. Firstly, the history and features of the RoboDK software are described. Secondly, the RoboDK robotic arm library and RoboDK API are outlined. Finally, at the end of the chapter, a short introduction to RoboDK post processors is given. It is followed by [Chapter 5](#) where the knowledge gained in this chapter is used practically.

4.1 RoboDK overview

RoboDK (short for Robot Development Kit) is a development platform for industrial robotic arms offline and online programming and simulation. RoboDK (the company) was founded by Albert Nubiola and Lauren Ierullo in January 2015 as a spin-off company from the [CoRo laboratory](#) at ETS University in Montreal, Canada. RoboDK is a commercial version of [RoKiSim](#), a multiplatform educational software tool for 3D simulation of serial 6-axis robotic arms [\[34\]](#).

4.1.1 RoboDK features

The following list highlights some of the features RoboDK has to offer. These features include:

- graphical user interface,
- drag-and-drop functionality,
- support for 3D file formats – importing objects and creating new tools using 3D files such as STL, STEP and IGES,
- CAD-to-path features – generating robotic arm programs directly from curves placed on station objects,
- external axes – integrating external axes to extend the robotic arm’s reachability,
- generating programs – generating programs for various robotic arm manufacturers,
- running programs on the fly – executing programs directly from an external computer,
- real-time monitoring – viewing the robotic arm state on an external computer,
- computer aided manufacturing for robotic arms – converting 5-axis CNC toolpaths to robotic arm programs and using a robotic arm like a 5-axis CNC,
- automated path solving – avoiding robotic arm errors, including singularities, joint limits and collisions,
- fast collision detection – defining the object interactions,
- advanced use – creating robotic arm programs from an external computer using a higher programming language. The RoboDK API is available in Python, C#, Visual Basic, C++, and Matlab,
- simulating 2D vision cameras – testing image recognition algorithms in the simulation environment,
- multiple robotic arms simulation – synchronising and programming multiple robotic arms and moving them at the same time,
- customisable post processors – integrating specific sensors or actuators such as grippers, force control, image processing, etc. [35](#).

4.2 RoboDK robotic arm library

RoboDK supports offline programming and has an extensive robotic arm library supporting robotic arm controllers from various manufacturers, including, but not limited to:

- ABB,
- FANUC,
- KUKA ,
- Motoman,
- Universal Robots,
- Kawasaki.

Models of industrial robotic arms in RoboDK have the same properties as if used with a physical robotic arm controller. The RoboDK library can be accessed online either via a browser or in the RoboDK application itself (keyboard shortcut `Ctrl+Shift+0`) [36].

4.3 RoboDK interface

The interface of RoboDK consists of the main menu, the toolbar, the station tree, the status bar, the 3D view, and the robot panel. A picture of the RoboDK interface is shown in Figure 4.1 [37].

4.3.1 RoboDK station

A RoboDK project containing a robotic arm, robotic arm tools, additional CAD files, robotic arm frames, and a robotic arm program is called a station. A RoboDK station is saved as one file (`.rdk` extension). The FANUC M-20iA/35M model from the RoboDK robotic arm library is used in this thesis because it is readily available in the RoboDK library and differs from the FANUC M-20iA/20M robotic arm used in the LSP station at HiLASE centre only in payload capacity. Robotic arm files in RoboDK have a `.robot` extension [38].

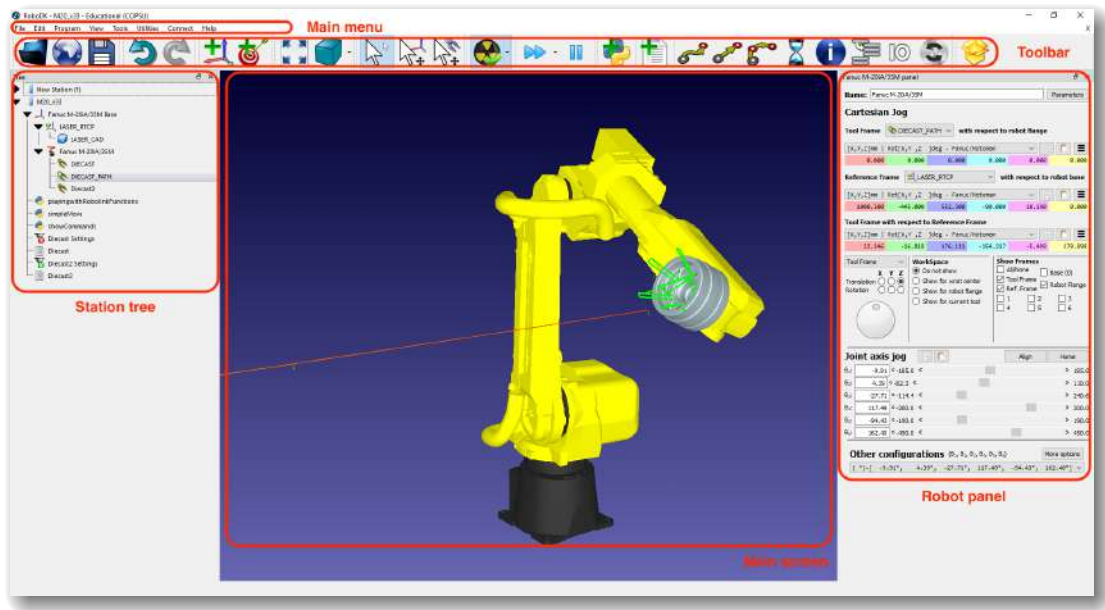


Figure 4.1: RoboDK version 5.2 running on Windows 10

4.4 RoboDK API

RoboDK provides a graphical user interface (GUI) to simulate and program industrial robotic arms. No programming experience is required to simulate and program robotic arms using the GUI. Unfortunately, the GUI has some limitations when used for simulation and offline programming. If such a situation arises, then the RoboDK application interface (API) can be used to extend the capabilities of RoboDK.

The API provides a set of routines and commands to RoboDK, enabling the user to program the robotic arm using high-level programming languages. The RoboDK API is implemented in Python, C#, C++, Visual Basic (.NET) and Matlab. Any of these programming languages can be used to simulate and program any robotic arm. The API can conveniently handle the following tasks:

- automating simulation,
- offline programming,
- online programming.

The RoboDK API for Python consists of the following modules:

- `roboDK` module,

- `item` module,
- `roboDK` module.

The RoboDK API is described in more detail in [Chapter 5](#).

4.5 RoboDK Add-ins

4.5.1 RoboDK Add-in for SolidWorks

Solidworks is a professional 3D CAD modelling application. The RoboDK Add-in for SolidWorks allows to combine SolidWorks' 3D CAD modelling features with robotic arm simulation and offline programming in RoboDK and considerably simplifies the programmer's workflow. The programmer can load the 3D models created in SolidWorks directly to RoboDK. Groups of curves or points can also be loaded to RoboDK, and robotic arm programs can be generated from the curves or points subsequently.

SolidWorks RoboDK Add-in toolbar

The RoboDK Add-in for SolidWorks is accessible directly from the SolidWorks toolbar. The RoboDK Add-in toolbar is shown in [Figure 4.2](#). The Add-in toolbar presents the programmer with several functionalities:

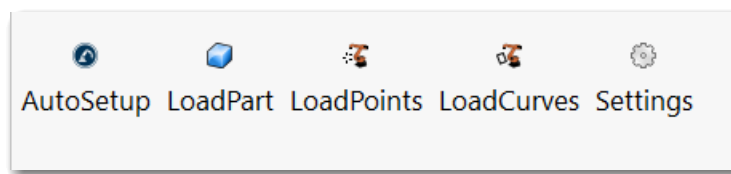


Figure 4.2: SolidWorks RoboDK Add-in toolbar

- `AutoSetup` – the user selects the geometry (curves and points) and the model, the curves are automatically transferred to RoboDK,
- `LoadPart` – only loads the 3D model from SolidWorks to RoboDK without importing the curves or points,
- `LoadPoints` – loads points to RoboDK as a new object. Selected surfaces are used to calculate the curve normals,

- **LoadCurves** – loads curves selected in RoboDK as a new object. Selected surfaces are used to calculate curve normals,
- **Settings** – opens the default settings window [39](#).

SolidWorks RoboDK Add-in settings window

Figure [4.3](#) shows the SolidWorks RoboDK Add-in settings window. The settings window contains two particular options the user should pay attention to. These options are called minimum and maximum step size and are highlighted in Figure [4.3](#) in red. By checking both checkboxes and entering the same value in both fields, it is ensured that the path points will be generated along the curve at equidistant distances. This is of particular importance if the user carries out the LSP process point-by-point (i.e., the robotic arm reaches the point, then stops, the laser fires one pulse, and the robotic arm continues to the next point).

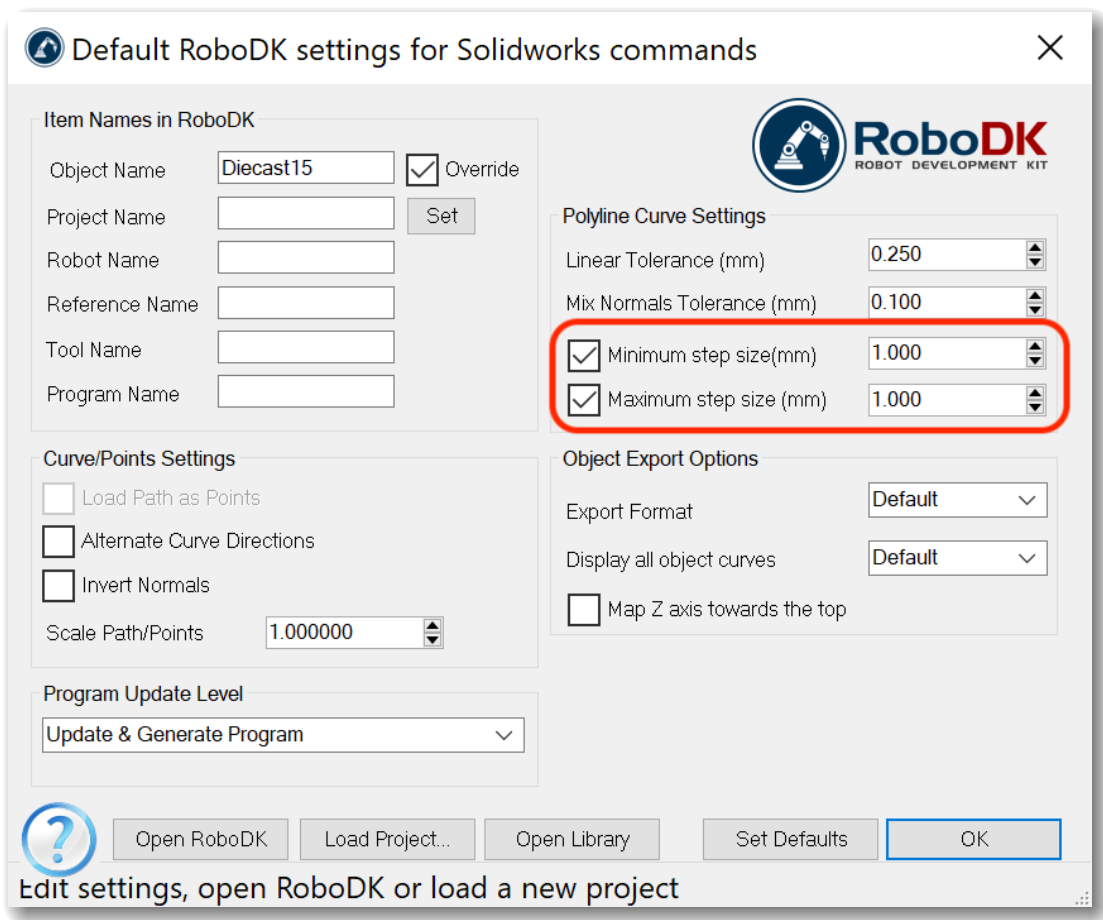


Figure 4.3: SolidWorks RoboDK Add-in settings window

4.6 RoboDK collision detection

RoboDK is equipped with a tool that checks for collisions in the simulated workcell. Unfortunately, RoboDK can not foresee collisions and can not stop a running simulation before a collision occurs. RoboDK can only detect collisions. A successful collision-free simulation is the first step in running a program. If the simulation is collision-free, the next step is to test the robotic arm program on the physical robotic arm. The simulated workcell should represent the physical workcell as accurate as required by the given application.

The collision detection is activated by selecting **Tools** → **Check collisions**. The setup of the collision detection is done via the collision map. An example of a collision map is shown in Figure 4.4. The collision map is a Boolean matrix representing the collision check state between all the moving objects in the workcell. The rows and columns of the matrix are named after the moving objects in the workcell. The entries in the matrix display the collision check state of a pair of moving objects in the workcell – the user can activate or deactivate this collision check state [40].

	LASER_CAD	Fanuc M-20iA/35M (J0)	Fanuc M-20iA/35M (J1)	Fanuc M-20iA/35M (J2)	Fanuc M-20iA/35M (J3)	Fanuc M-20iA/35M (J4)	Fanuc M-20iA/35M (J5)
LASER_CAD	☒	☒	☒	☒	☒	☒	☒
Fanuc M-20iA/35M (J0)	☒	☒	☒	☒	☒	☒	☒
Fanuc M-20iA/35M (J1)	☒	☒	☒	☒	☒	☒	☒
Fanuc M-20iA/35M (J2)	☒	☒	☒	☒	☒	☒	☒
Fanuc M-20iA/35M (J3)	☒	☒	☒	☒	☒	☒	☒
Fanuc M-20iA/35M (J4)	☒	☒	☒	☒	☒	☒	☒
Fanuc M-20iA/35M (J5)	☒	☒	☒	☒	☒	☒	☒
Fanuc M-20iA/35M (J6)	☒	☒	☒	☒	☒	☒	☒
DIECAST	☒	☒	☒	☒	☒	☒	☒
Diecast15	☒	☒	☒	☒	☒	☒	☒
holder_cylinder	☒	☒	☒	☒	☒	☒	☒

Figure 4.4: Example of a collision map

4.7 RoboDK post processors

Post processors generate robotic arm programs for robotic arm controllers from robotic arm simulations. Post processors are essential for offline programming of robotic arms. A post processor defines the vendor-specific rules a robotic arm program must follow. A robotic arm must be linked to a post processor in RoboDK to be able to generate

a robotic arm controller program. A post processor in RoboDK is a Python script (`.py` extension). All the RoboDK post processors are located in the `C:/RoboDK/Posts/` folder (in case of the Windows operating system). To use a post processor it must be placed in the `C:/RoboDK/Posts/` folder. The user can modify an existing post processor or create a new one from scratch. The modification of post processors is described in [Chapter 5](#) in more detail [\[41\]](#).

Chapter 5

Post processor implementation

Python is the most powerful language you can still read.

Paul Dubois

Lead developer for Numerical Python and Pyfort

This chapter familiarises the reader with the creation of a curve follow project in RoboDK. Firstly, it focuses on the specifics of programming FANUC robotic arms. Secondly, it gives a short introduction to setting up a curve follow project in RoboDK. Subsequently, it describes the RoboDK API, and in the end, it deals with the problematics of post processors and the modification of post processors.

5.1 FANUC robotic arms programming specifics

5.1.1 FANUC Roboguide

FANUC Roboguide is a proprietary robotic arm simulation and offline programming tool developed by the FANUC corporation. FANUC Roboguide is in many ways similar to RoboDK. Like RoboDK, it supports the creation of robotic arm stations, importing of CAD files and CAD-to-path features. The main difference is that Roboguide is limited to FANUC robotic arms and FANUC related technologies and procedures. In contrast, RoboDK is not limited to one robotic arm manufacturer and is universal and expandable. FANUC Roboguide is used in this thesis to compile the created robotic arm controller programs and upload them to the robotic arm controller. FANUC Roboguide offers

several simulation software options tailored for specific robotic arm applications:

- FANUC Roboguide HandlingPRO – simulating material handling applications including loading/unloading, packaging, assembly and material removal,
- FANUC Roboguide PaintPRO – simulating painting applications,
- FANUC Roboguide WeldPRO – simulating robotic arc welding processes,
- FANUC Roboguide PalletPRO and PalletTool – simulating palletizing applications.

An example of a FANUC Roboguide station is shown in Figure 5.1. The version of FANUC Roboguide used in this study is 8.30104.00.35 (Rev. K) [42].

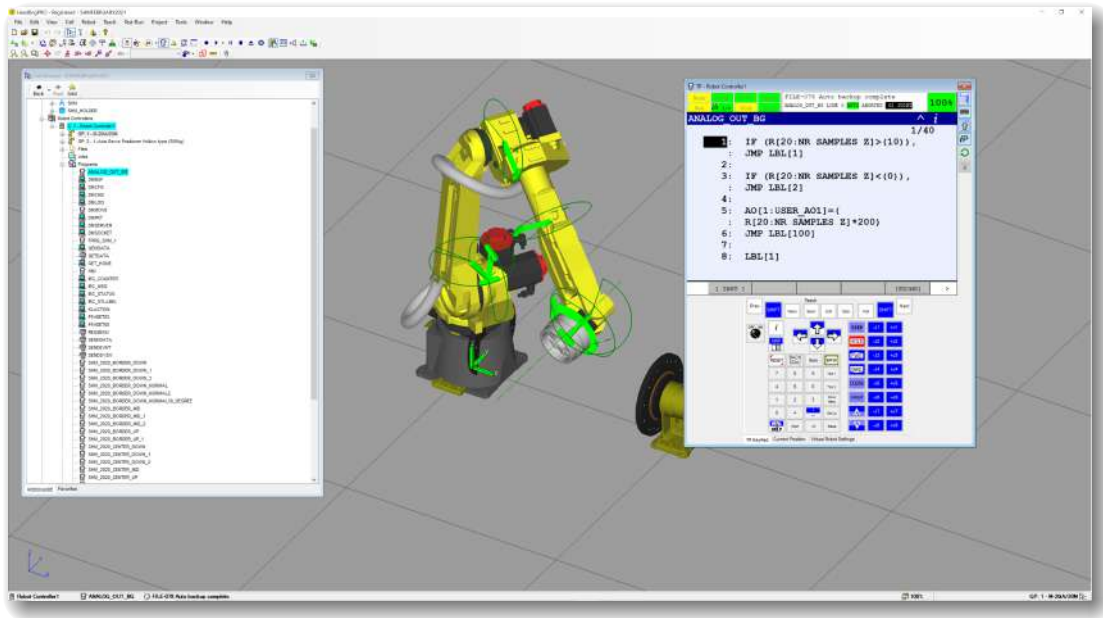


Figure 5.1: FANUC Roboguide workcell – user interface example

5.1.2 FANUC robotic arm controller programming languages

The FANUC company implements two programming languages for programming their robotic arm controllers: teach pendant (TP) or KAREL. The TP language is mainly used for motion control of the robotic arm and the programs are usually edited via the pendant. TP programs are either binary files (.tp file extension) or can be human-readable ASCII files (.ls file extension). The syntax of the TP language is described in great detail

in the *SYSTEM R-30iB and R-30iB MATE Handling Tool Setup and Operations Manual* [10]. The KAREL language is a high-level language and does not support robotic arm movement instructions. KAREL is mainly used to implement program logic. KAREL programs can not be edited using a pendant. The syntax of the KAREL language is covered in the *FANUC Robotics SYSTEM R-30iA Controller KAREL Reference Manual* [43].

5.1.3 Teach pendant program structure

A program element is a segment of a program. A TP program is a collection of program elements with the purpose of executing an application. Figure 5.2 shows an example of a TP program with some typical program elements. A TP program is typically composed of the following program elements:

- program header information – includes information such as program name, comment, group mask, program type, application mask, write protection setting, and cycle time,
- UTOOL and FRAME number entries – specify the frame and tool used,
- line numbers – assigned to each program instruction,
- motion instructions – include commands that tell the robotic arm where and how to move,
- program instructions for logic, I/O, data handling, program control, advanced functions,
- remarks – annotate the program,
- program end marker – indicates that there are no more instructions in the program.

A motion instruction is the most important type of program element. A motion instruction directs the robotic arm to move in a specified way to a specific location in the workcell using a specified speed. Figure 5.3 shows a typical example of a motion instruction. A motion instruction consists of:

- motion type – how the robotic arm moves to the position,

```

PROGRAM NAME → TEMPL_SINGLE_SHOT_LITRON ^ i
1/9
FRAME SELECTION → 1: UFRAME_NUM=1
2: WAIT 2.00(sec)
PROGRAM INSTRUCTION → 3: WAIT DI[1]=OFF
4: WAIT DI[1]=ON
REMARK → 5: IF R[1]=10,CALL ANALOG_OUT_BG
6: !REMARK
MOTION INSTRUCTION → 7: L P[1] 100mm/sec CNT100
PROGRAM END MARKER → 8: DO[9]=PULSE,0.1sec
[End]

```

Figure 5.2: Example of a TP program with some typical program elements [10]

- position indicator symbol – indicates that the robotic arm is at the taught position,
- positional information – where the robotic arm moves,
- termination type – how the robotic arm ends the move to the position,
- speed – how fast the robotic arm moves to a position,
- motion options – additional commands that perform specific tasks during robotic arm motion.

5.1.4 Compiling a FANUC teach pendant program

Only a TP program in binary format can be run on FANUC robotic arm controllers. Because RoboDK creates TP programs as human-readable ASCII files, the TP programs need to be converted to binary format before uploading them to the robotic arm controller. Two options to convert `.ls` programs to `.tp` programs exist:

1. the ASCII Upload option must be loaded on the robotic arm controller. After uploading an `.ls` file to the controller, it is automatically converted to a `.tp` file,
2. the program is compiled and uploaded either using the WinOLPC tools via Roboguide or using the WinOLPC tools directly [10]. The WinOLPC tools are located in `C:/Program File(x86)/FANUC/WinOLPC/bin` in the Windows operating system.

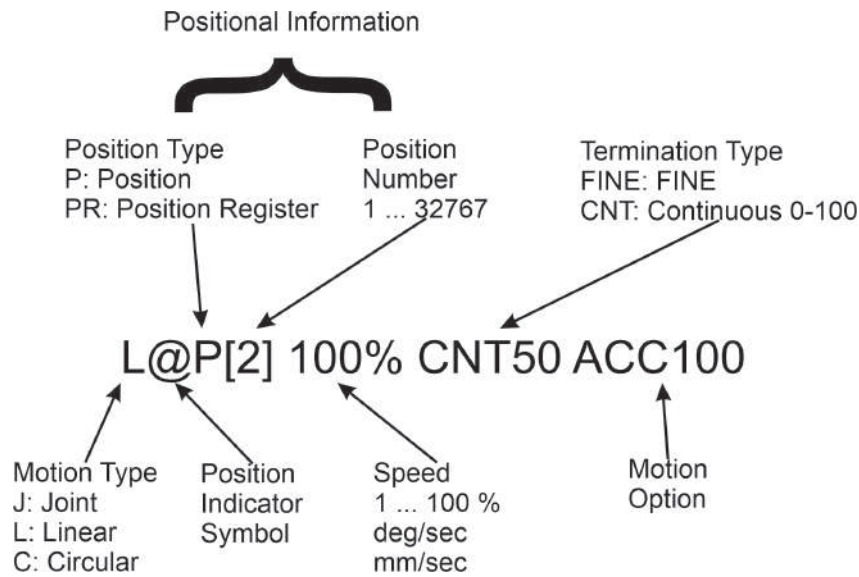


Figure 5.3: Example of motion instruction [10]

5.2 Robot machining projects in RoboDK

The applications of robotic arm machining in industry are numerous. Some applications include:

- milling,
- drilling,
- chamfering,
- deburring.

RoboDK supports three types of robot machining projects:

- robot machining projects – a robotic arm path is created from an Numerical Control (NC) file,
- curve follow projects – a robotic arm follows a curve path in 3D space,
- point follow projects – a robotic arm follows a path defined by points in 3D space.

The process of setting up a curve follow project in RoboDK is described in the following sections. In laser shock peening, the laser (representing the tool) is static, and the robotic arm holds the object. Therefore, a curve follow project with a constant tool orientation is set up in RoboDK [44].

5.2.1 Setting up a curve follow project in RoboDK

A curve follow project in RoboDK must at least consist of one robotic arm, one tool, and one reference frame. The situation described is a so-called remote TCP situation, i.e., the Tool Center Point (TCP) is fixed in the station, and the robotic arm holds the object. The TCP in our project is represented by the laser beam. The user has to execute the following steps to set up a basic curve follow project and to generate a vendor-specific robotic arm program:

1. create and import the robotic arm path to RoboDK using preferred CAD software,
2. mount the robotic arm path as a tool in RoboDK,
3. create reference frame (optionally import CAD of manufactured part) in RoboDK station,
4. create curve follow project: `Utilities` → `Curve follow project`,
5. open the curve follow project settings. A window similar to the one displayed in Figure [5.4](#) will open,
6. modify curve follow project settings:
 - `Robot` – the robotic arm holding the object,
 - `Reference` – the frame representing the remote TCP,
 - `Tool1` – the robotic arm path.
7. change `Select algorithm` option to: `Robot holds object & follows path`,
8. select `Update project`. RoboDK automatically generates a preprocessed robotic arm program for the actual station,
9. right-click the preprocessed program, select post processor and generate program. The `.ls` program for the actual station will be opened in the default RoboDK text editor,
10. congratulations. The curve follow project is now set up,
11. compile the program and export the program to the FANUC robotic arm controller using FANUC Roboguide or WinOLPC tools [45](#).

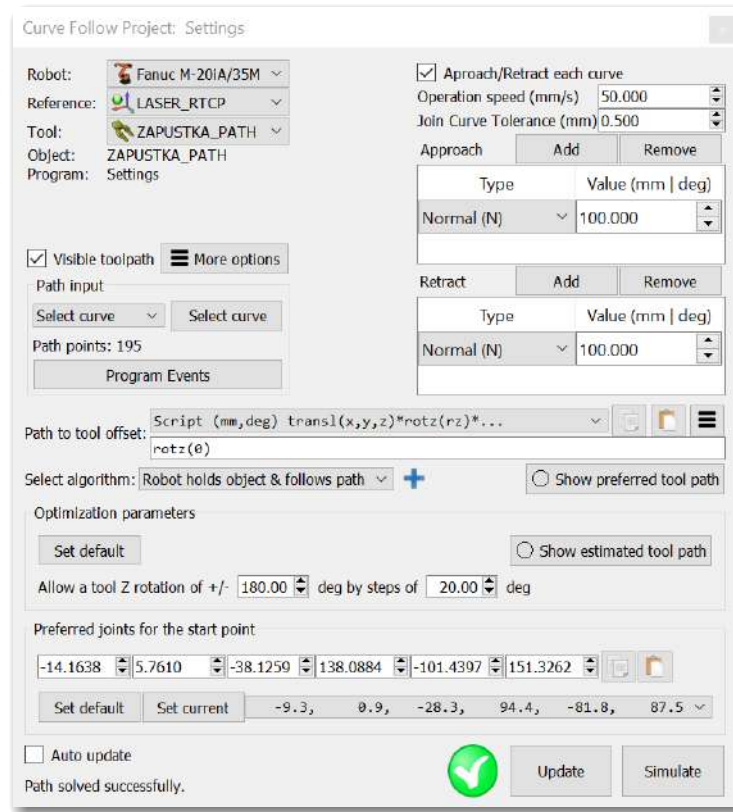


Figure 5.4: Curve follow project settings in RoboDK

5.3 RoboDK API

An application interface (API) is a set of features an application provides to the user via a high-level programming language. The RoboDK API dramatically expands the potential of RoboDK. The RoboDK API is implemented in Python, C#, C++, Visual Basic (.NET) and Matlab. The Python API is the most extensive and is used in this thesis. The RoboDK API facilitates the execution of the following tasks:

- automating simulations – macro scripts for repetitive tasks such as tool changing, pick and place applications.
- offline programming – using the RoboDK API methods to create programs for specific robotic arm controllers. The pre-processed .pyc file created in RoboDK’s GUI is executed directly by the post processor. The post processor defines the rules for vendor-specific robotic arm controller program generation.
- online programming – using RoboDK API to move robotic arms and retrieve their current position in real time. RoboDK connects to the robotic arm controller using

robotic arm drivers.

The elegance of this approach is that the same program written in a high-level programming language can be used for all three of the above-mentioned tasks [46].

5.3.1 Python API for RoboDK

Python is an interpreted high-level programming language. The Python API for RoboDK uses Python 3, although most features are compatible with Python 2 also. The RoboDK API for Python consists of the following modules:

- `roboLink` module – this module represents the link between RoboDK and the high-level programming language,
- `item` module – any item from the RoboDK item tree can be retrieved. An item can be a robotic arm, a reference frame, a tool, an object, or a specific project,
- `roboDK` module – a module with a robotics toolbox for pose operations. All post processors depend on the `roboDK` module .

The modules are located in the folder `C:/RoboDK/Python/` in the Windows operating system. This folder is included in the `PYTHONPATH` system variable. The Python API is accompanied with examples. These examples are located in the folders `C:/RoboDK/Library/Scripts` and `C:/RoboDK/Library/Macros` in the Windows operating system [47].

5.4 RoboDK post processors

A post processor specifies how robotic arm programs must be generated for a specific robot controller and are used when robotic arm programs are generated offline. Post processors serve as tools to convert the simulation to vendor-specific robotic arm programs. All RoboDK post processors are placed in the `C:/RoboDK/Posts` folder in the Windows operating system. All post processors rely on the `roboDK` module. The `roboDK` module is a robotics toolbox for Python, based on [Peter Corke's Robotics Toolbox](#) [47].

5.4.1 RoboDK post processor workflow

The creation of a program that is compatible with a FANUC robotic arm controller consists of three steps. These steps are shown in Figure 5.5:

1. create a program in a curve follow project in RoboDK,
2. generate a preprocessed/universal `.pyc` file using RoboDK,
3. generate a vendor-specific robotic arm `.ls` program using the post processor.

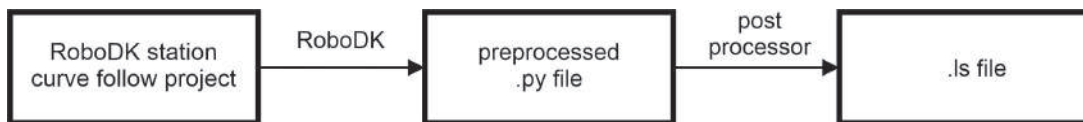


Figure 5.5: RoboDK post processor workflow

5.4.2 RoboDK preprocessed file

After creating a program in the curve follow project, but before generating it using a post processor, the program is saved as a preprocessed/universal Python program and saved in a local temporary folder. The post processor defines a `RobotPost` class that generates the desired code. In the Windows operating system, the preprocessed Python files are saved in the user's temporary folder (e.g.: `C:/Users/username/AppData/Temp`). These programs can also be used for debugging new post processors. A code snippet of a preprocessed file is shown in Listing 5.1 [48]. The three dots (...) denote parts of the code that have been intentionally omitted to keep the source code listing shorter.

```

import sys
import os
sys.path.append(os.path.abspath(r"C:/RoboDK/Posts/")) # temporarily add path to
↳ POSTS folder

from FANUC_R30iA_decompile3 import *

def p(x,y,z,r,p,w):
    a~ = r*math.pi/180.0

```

```

b = p*math.pi/180.0
c = w*math.pi/180.0
ca = math.cos(a)
sa = math.sin(a)
cb = math.cos(b)
sb = math.sin(b)
cc = math.cos(c)
sc = math.sin(c)
return Mat([[cb*ca,ca*sc*sb--cc*sa,sc*sa+cc*ca*sb,x],
           ↪ [cb*sa,cc*ca+sc*sb*sa,cc*sb*sa--ca*sc,y], [--sb,cb*sc,cc*cb,z],
           ↪ [0.0,0.0,0.0,1.0]])

print('Total instructions: 140')
r = RobotPost(r""FANUC_R30iA_decompyle3"",r""Fanuc M--20iA/35M"",6,
           ↪ axes_type=['R','R','R','R','R','R'], ip_com=r""127.0.0.1"", api_port=20500,
           ↪ prog_ptr=2172227036784, robot_ptr=2172252511552)

r.ProgStart(r""Settings"")
r.RunMessage(r""Program generated by RoboDK v5.2.2 for Fanuc M--20iA/35M on
           ↪ 02/07/2021 17:12:29"",True)
r.RunMessage(r""Using nominal kinematics."",True)
r.setZoneData(1.000)
r.setSpeed(1000.000)
r.setFrame(p(1000.3,--445,552.3,0,10.19,--90),--1,r""LASER_RTCP"")
r.setTool(p(0,0,0,0,0,0),--1,r""ZAPUSTKA_PATH"")
r.RunMessage(r""Show ZAPUSTKA_PATH"",True)
r.MoveJ(None,[--9.2878,0.949114,--28.3094,94.4347,--81.8311,87.4811],None)
r.MoveL(p(--14.8303,41.4662,85,105.917,0,180),
           ↪ [--14.8823,2.45519,--28.2667,97.1731,--76.9261,87.0176], [0,0,0])
r.setSpeed(10.000)
r.MoveL(p(0.866853,42.1343,85,94.455,0,180),
           ↪ [--14.6627,3.59506,--28.447,97.1044,--77.1403,75.3963], [0,0,0])

...
...
...

r.setSpeed(1000.000)
r.MoveL(p(--16.1785,41.074,185,107.509,0,180),
           ↪ [--9.29955,0.844572,--28.2678,94.4344,--81.8176,449.114], [0,0,0])
r.ProgFinish(r""Settings"")

```

```
r.ProgSave(r""C:/Users/marek.boehm/Documents/RoboDK/"" , r""Settings"" , False,
↔ r""C:/RoboDK/Other/VSCodium/VSCodium.exe"" )
```

Listing 5.1: Code snippet of preprocessed program to be executed by post processor

Let us dive into the source code of this preprocessed file and explain the functionality of some of its methods. The most important method is the `MoveL` function:

```
MoveL(pose, joints, conf_RLF=None)
```

The `MoveL` method generates manufacturer-specific code for linear movement. The parameters of the `MoveL` method are:

- `pose` (`roboDK.Mat()`) – pose target of the tool with respect to the reference frame, pose can be `None` if the target is defined as a joint target,
- `joints` (`float list`) – robotic arm joints as a list,
- `conf_RLF` (`int list`) – pass additional arguments to function.

```
ProgStart(progname)
```

The `ProgStart` method is called when a new program is generated. The parameters of the `ProgStart` method are:

- `progname` (`str`) – program name.

```
setSpeed(speed_mms)
```

The `setSpeed` method sets the linear robotic arm speed (in mm s^{-1}). The parameters of the `setSpeed` method are:

- `speed_mms` – speed in mm s^{-1} .

```
ProgSave(folder, progname, ask_user=False, show_result=False)
```

The `ProgSave` method saves the program in a format compatible with the corresponding robotic arm controller type. The parameters of the `ProgSave` method are:

- `folder` (`str`) – folder hint to save the program,
- `progname` (`str`) – program name as a hint to save the program,

- `ask_user` (`bool`, `str`) – true if the default settings in RoboDK are set to prompt the user to select the folder,
- `show_result` (`bool`, `str`) – false if the default settings in RoboDK are set to not show the program once it has been saved. Otherwise, a string is provided with the path of the preferred text editor.

`ProgFinish(progname)`

The `ProgFinish` method is triggered after all the instructions of the program and defines the end of the program. The parameters of the `ProgFinish` method are:

- `progname` (`str`) – program name.

Every post processor should implement a set of basic methods to generate programs for robotic arm controllers correctly [49].

5.4.3 Selecting a post processor

Each robotic arm has a post processor assigned to it by default. The following steps have to be taken to change the post processor of a robotic arm:

1. opening the robot panel,
2. selecting `Parameters`,
3. selecting post processor from drop-down list [50].

5.4.4 Post processor example and post processor example output

An example of an output of the unmodified R-30iA post processor output is shown in Listing [5.2].

```
/PROG Diecast15
/ATTR
OWNER    = MNEDITOR;
COMMENT  = "RoboDK sequence";
PROG_SIZE = 0;
```



```
CREATE      = DATE 31-12-14 TIME 12:00:00;
MODIFIED   = DATE 31-12-14 TIME 12:00:00;
FILE_NAME  = Diecast15;
VERSION    = 0;
LINE_COUNT = 462;
MEMORY_SIZE = 0;
PROTECT    = READ_WRITE;
TCD: STACK_SIZE = 0,
      TASK_PRIORITY = 50,
      TIME_SLICE = 0,
      BUSY_LAMP_OFF = 0,
      ABORT_REQUEST = 0,
      PAUSE_REQUEST = 0;
DEFAULT_GROUP = 1,*,*,*,*,*,*,*;
CONTROL_CODE = 00000000 00000000;
/MN
```

```
1: ! Program generated by ;
2: ! RoboDK v5.2.2 for F ;
3: ! anuc M-20iA/35M on 0 ;
4: ! 9/08/2021 16:07:49 ;
5: ! Using nominal kinema ;
6: ! tics. ;
7: PR[9,1]=1000.300 ;
8: PR[9,2]=-445.000 ;
9: PR[9,3]=552.300 ;
10: PR[9,4]=-90.000 ;
11: PR[9,5]=10.190 ;
12: PR[9,6]=0.000 ;
13: UFRAME[9]=PR[9] ;
14: UFRAME_NUM=9 ;
15: PR[9,1]=0.000 ;
16: PR[9,2]=0.000 ;
17: PR[9,3]=0.000 ;
18: PR[9,4]=0.000 ;
19: PR[9,5]=0.000 ;
20: PR[9,6]=0.000 ;
21: UTOOL[9]=PR[9] ;
22: UTOOL_NUM=9 ;
23: ! Show Diecast15 ;
24: J P[1] 1% CNT0 ;
25: L P[2] 50mm/sec CNT0 ;
```

```

26:L P[3] 10mm/sec CNT0 ;
27:L P[4] 10mm/sec CNT0 ;

...
...
...

461:L P[438] 10mm/sec CNT0 ;
462:L P[439] 50mm/sec CNT0 ;
/POS
P[1]{
  GP1:
    UF : 9, UT : 9,
    J1=  -11.825 deg, J2=  0.745 deg, J3=  -13.905 deg,
    J4=  69.881 deg, J5=  -63.946 deg, J6=  28.200 deg
};
P[2]{
  GP1:
    UF : 9, UT : 9,      CONFIG : 'N U T, 0, 0, 0',
    X =   1.632 mm,  Y =  -67.788 mm,  Z =   73.203 mm,
    W = -165.352 deg, P =  -21.879 deg, R =   21.809 deg
};
P[3]{
  GP1:
    UF : 9, UT : 9,      CONFIG : 'N U T, 0, 0, 0',
    X =   2.074 mm,  Y =  -68.202 mm,  Z =   73.760 mm,
    W = -165.468 deg, P =  -21.989 deg, R =   21.644 deg
};
P[4]{
  GP1:
    UF : 9, UT : 9,      CONFIG : 'N U T, 0, 0, 0',
    X =   2.535 mm,  Y =  -68.640 mm,  Z =   74.302 mm,
    W = -165.566 deg, P =  -22.104 deg, R =   21.494 deg
};

...
...
...

P[439]{
  GP1:
    UF : 9, UT : 9,      CONFIG : 'N U T, 0, 0, -1',

```

```

X =   -1.638 mm,   Y =  -138.500 mm,   Z =   143.912 mm,
W =  -165.328 deg, P =    21.824 deg, R =   -21.741 deg
};
/END

```

Listing 5.2: Code snippet of unmodified R-30iA post processor output

5.5 FANUC R-30iA RoboDK post processor

The FANUC R-30iA post processor is located in the `C:/RoboDK/Posts/vXX` folder. XX is a two-digit number and denotes the post processor version. The FANUC R-30iA post processor comes in the form of a `.pyc` file (compiled bytecode) and needs to be decompiled to a `.py` file with the help of a decompiler. The decompiler used in this study is `decompyle3` [51]. The FANUC R-30iA post processor is compatible with FANUC R-30iB robotic arm controllers.

5.5.1 Modifying the FANUC R-30iA post processor methods

The user can create a new post processor from scratch or modify an existing post processor. A post processor for a specific robotic arm controller is a single `.py` file. All post processors are placed in the `C:/RoboDK/Posts` folder. A new post processor is added to RoboDK by creating a `.py` file or renaming an existing `.py` file in this folder. By deleting a `.py` file in this folder, the post processor is deleted from the list of post processors.

The experimental setup of the LSP station at the HiLASE centre links the robotic arm controller to the laser sources. The objective of the post processor modification for the Litron LPY ST 7875-10 2HG laser can be expressed as:

“The laser source is operating at a 1 Hz repetition rate. The post processor for the R-30iA controller will be modified in such a way that when the robotic arm with the sample is in a position, where the LSP process (individual laser impact area) should be placed, the robotic arm controller sends a command to the laser source to fire exactly one laser pulse onto the desired area and then move on to the next position. This is accomplished by controlling the digital inputs (DIs) and digital outputs (DOs) of the robotic arm controller.”

The modification involves the `MoveL` method. The unmodified `MoveL` method generates the following code:

```
n: L P[1] 4mm/sec CNT100;
```

where:

- `n` – number of program line,
- `L` – linear movement,
- `P[1]` – designation of point,
- `4mm/sec` – movement speed,
- `CNT100` – zone data.

The modified `MoveL` method should output the following robotic arm controller code to meet the requirements of the laser shock peening process:

```
n: WAIT DI[1:SYNCHRONIZATION_IN]=OFF;
n+1: WAIT DI[1:SYNCHRONIZATION_IN]=ON;
n+2: L P[1] 4mm/sec CNT100 TB 0.00sec,DO[5]=PULSE,1.0sec;
n+3: WAIT DO[5:LASER ON] = OFF;
n+4: WAIT 1.00(sec);
```

where:

- `n` – number of program line,
- `WAIT CONDITION` – wait condition, block code execution until condition is `True`,
- `WAIT DI[1:SYNCHRONIZATION_IN]=OFF; WAIT DI[1:SYNCHRONIZATION_IN]=ON;` – detect rising edge of incoming synchronisation signal,
- `TB 0.00sec,DO[5]=PULSE,1.0sec` – execute command at a certain time before reaching the point,
- `P[1]` – designation of point,
- `DO[5]=PULSE,1.0sec` – rectangular pulse at digital output, pulse duration is one second, this command fires the laser,
- `WAIT 1.00(sec)` – block code execution for a certain amount of time.

Listing 5.3 contains the original MoveL method of the FANUC R-30iA post processor and Listing 5.4 contains the modified MoveL method 49.

```
def MoveL(self, pose, joints, conf_RLF=None):
    """Add a~linear movement"""
    if self.LAST_POSE is not None:
        if pose is not None:
            if distance(pose.Pos(), self.LAST_POSE.Pos()) < 0.001:
                if pose_angle_between(pose, self.LAST_POSE) < 0.001:
                    return
        self.page_size_control()
    if pose is None:
        target_id = self.add_target_joints(pose, joints)
        move_ins = 'P[%i] %s %s ;' % (target_id, self.SPEED, self.CNT_VALUE)
    else:
        target_id = self.add_target_cartesian(pose, joints, conf_RLF)
        move_ins = 'P[%i] %s %s ;' % (target_id, self.SPEED, self.CNT_VALUE)
    self.addline(move_ins, 'L')
    self.LAST_POSE = pose
```

Listing 5.3: Code snippet of original MoveL method

```
def MoveL(self, pose, joints, conf_RLF=None):
    """Add a~linear movement"""
    if self.LAST_POSE is not None:
        if pose is not None:
            if distance(pose.Pos(), self.LAST_POSE.Pos()) < 0.001:
                if pose_angle_between(pose, self.LAST_POSE) < 0.001:
                    return
        self.page_size_control()
    if pose is None:
        target_id = self.add_target_joints(pose, joints)
        syncro_off = 'WAIT DI[1:SYNCHRONIZATION_IN]=OFF;'
        syncro_on = 'WAIT DI[1:SYNCHRONIZATION_IN]=ON;'
        move_ins = 'P[%i] %s %s TB 0.00sec,DO[5]=PULSE,1.0sec;' % (target_id,
            ↪ self.SPEED, self.CNT_VALUE)
        wait_off = 'WAIT DO[5:LASER ON] = OFF;'
        wait_on = 'WAIT 1.00(sec);'
    else:
```

```

target_id = self.add_target_cartesian(pose, joints, conf_RLF)
syncro_off = 'WAIT DI[1:SYNCHRONIZATION_IN]=OFF;'
syncro_on = 'WAIT DI[1:SYNCHRONIZATION_IN]=ON;'
move_ins = 'P[%i] %s %s TB 0.00sec,DO[5]=PULSE,1.0sec;' % (target_id,
    ↪ self.SPEED, self.CNT_VALUE)
wait_off = 'WAIT DO[5:LASER ON] = OFF;'
wait_on = 'WAIT 1.00(sec);'

self.addline(syncro_off, ' ')
self.addline(syncro_on, ' ')
self.addline(move_ins, 'L')
self.addline(wait_off, ' ')
self.addline(wait_on, ' ')
self.LAST_POSE = pose

```

Listing 5.4: Code snippet of modified MoveL method

5.5.2 Example output of the modified FANUC R-30iA post processor

An example of an output of the modified R-30iA post processor is shown in Listing

5.5.

```

/PROG Diecast15
/ATTR
OWNER    = MNEDITOR;
COMMENT  = "RoboDK sequence";
PROG_SIZE = 0;
CREATE   = DATE 31-12-14 TIME 12:00:00;
MODIFIED = DATE 31-12-14 TIME 12:00:00;
FILE_NAME = Diecast15;
VERSION  = 0;
LINE_COUNT = 2214;
MEMORY_SIZE = 0;
PROTECT  = READ_WRITE;
TCD: STACK_SIZE = 0,
    TASK_PRIORITY = 50,
    TIME_SLICE = 0,
    BUSY_LAMP_OFF = 0,
    ABORT_REQUEST = 0,
    PAUSE_REQUEST = 0;

```

```
DEFAULT_GROUP = 1,*,*,*,*,*,*;
CONTROL_CODE = 00000000 00000000;
/MN

1: ! Program generated by ;
2: ! RoboDK v5.2.2 for F ;
3: ! anuc M-20iA/35M on 1 ;
4: ! 8/09/2021 16:36:30 ;
5: ! Using nominal kinema ;
6: ! tics. ;
7: PR[9,1]=1000.300 ;
8: PR[9,2]=-445.000 ;
9: PR[9,3]=552.300 ;
10: PR[9,4]=-90.000 ;
11: PR[9,5]=10.190 ;
12: PR[9,6]=0.000 ;
13: UFRAME[9]=PR[9] ;
14: UFRAME_NUM=9 ;
15: PR[9,1]=0.000 ;
16: PR[9,2]=0.000 ;
17: PR[9,3]=0.000 ;
18: PR[9,4]=0.000 ;
19: PR[9,5]=0.000 ;
20: PR[9,6]=0.000 ;
21: UTOOL[9]=PR[9] ;
22: UTOOL_NUM=9 ;
23: ! Show Diecast15 ;
24: J P[1] 1% CNTO ;
25: WAIT DI[1:SYNCHRONIZATION_IN]=OFF;
26: WAIT DI[1:SYNCHRONIZATION_IN]=ON;
27: L P[2] 50mm/sec CNTO TB 0.00sec,DO[5]=PULSE,1.0sec;
28: WAIT DO[5:LASER ON] = OFF;
29: WAIT 1.00(sec);
30: WAIT DI[1:SYNCHRONIZATION_IN]=OFF;
31: WAIT DI[1:SYNCHRONIZATION_IN]=ON;
32: L P[3] 10mm/sec CNTO TB 0.00sec,DO[5]=PULSE,1.0sec;
33: WAIT DO[5:LASER ON] = OFF;
34: WAIT 1.00(sec);

...
...
...
```

```

2210: WAIT DI[1:SYNCHRONIZATION_IN]=OFF;
2211: WAIT DI[1:SYNCHRONIZATION_IN]=ON;
2212:L P[439] 50mm/sec CNT0 TB 0.00sec,DO[5]=PULSE,1.0sec;
2213: WAIT DO[5:LASER ON] = OFF;
2214: WAIT 1.00(sec);
/POS
P[1]{
  GP1:
    UF : 9, UT : 9,
    J1=  -11.825 deg, J2=  0.745 deg, J3=  -13.905 deg,
    J4=  69.881 deg, J5=  -63.946 deg, J6=  28.200 deg
};
P[2]{
  GP1:
    UF : 9, UT : 9,      CONFIG : 'N U T, 0, 0, 0',
    X =   1.632 mm,  Y =  -67.788 mm,  Z =   73.203 mm,
    W = -165.352 deg, P =  -21.879 deg, R =   21.809 deg
};

...
...
...

P[439]{
  GP1:
    UF : 9, UT : 9,      CONFIG : 'N U T, 0, 0, -1',
    X =  -1.638 mm,  Y = -138.500 mm,  Z =  143.912 mm,
    W = -165.328 deg, P =   21.824 deg, R =  -21.741 deg
};
/END

```

Listing 5.5: Code snippet of modified R-30iA post processor output

Chapter 6

Testing of robotic arm program

All life is an experiment. The more experiments you make the better.

Ralph Waldo Emerson

American essayist

Firstly, this chapter deals with the simulation of the LSP process in RoboDK. Secondly, it focuses on testing the robotic arm program on the physical robotic arm. The goal of the testing is to evaluate the effectiveness of the solution realised in the CAM program RoboDK. The robotic arm cell used for simulation and testing is depicted in [section 3.2.1 – LSP station layout](#). A forging die is used as a test sample. [Figure 6.1](#) shows a CAD drawing of the forging die. [Figure 6.1](#) also highlights the areas to be treated by the LSP process in black and the approach/retreat motions in green. The area treated by LSP is made up of a line of consecutive laser shots. Information about the location of the areas to be treated by the LSP process is usually provided by the manufacturers based on experience from the factory floor. The actual testing described in this chapter was done with the Litron LPY ST 7875-10 2HG laser and the FANUC M-20iA/20M robotic arm.

6.1 Simulation

The simulation process can be broken down into several steps:

1. create a robot machining project in RoboDK, so it mirrors the actual robotic arm workcell,

2. set up the curve follow project, set appropriate collision checking parameters and successfully solve robotic arm path,
3. set appropriate simulation parameters and run simulation, generate robotic arm program with RoboDK,
4. if simulation is successful, continue with testing on physical robotic arm.

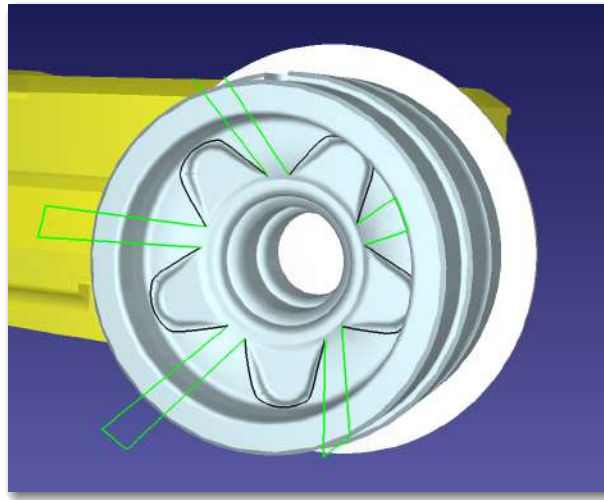


Figure 6.1: CAD drawing of forging die

Figure [6.2](#) shows a picture of the RoboDK workcell with the forging die mounted onto the flange of the FANUC M-20iA/20M robotic arm.

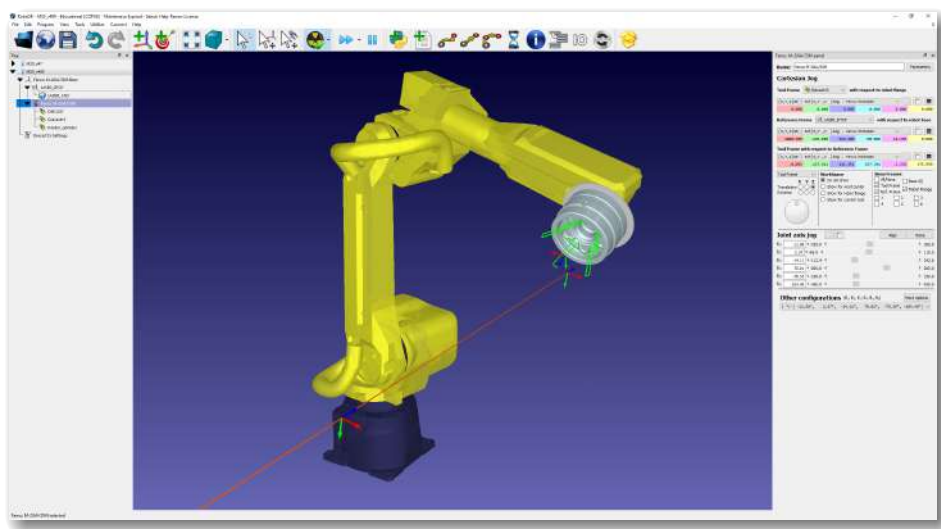


Figure 6.2: RoboDK workcell with forging die

6.2 Testing on the physical robotic arm

The testing of robotic arm programs on physical robotic arm succeeds the simulation process and is roughly divided into the following steps:

1. upload program to robotic arm controller,
2. run program on physical robotic arm in test mode (test mode = mode of robotic arm with reduced maximum speed) and without active laser source,
3. run program on physical robotic arm in production mode (production mode = speed of robotic arm is not restricted) and without active laser source,
4. run program in production mode with active laser source. The energy of the laser source is set to a value that is appropriate for the given LSP application.

The parameters chosen for the forging die experiment are listed in Table [6.1](#).

Table 6.1: Experimental parameters of LSP process on forging die

Parameter	Value
Laser source	Litron LPY ST
Wavelength [ns]	1064
Repetition rate [Hz]	1
Pulse energy at sample [mJ]	3000
Beam size at sample [mm ²]	4.91
Pulse width @1064nm [ns]	12
Power density at sample [GW cm ⁻²]	5.1

6.3 Results of simulation and testing

6.3.1 Results of simulation

The simulation has been a quick way of testing various ideas by changing the parameters of the simulation. The simulation has also allowed the author to create large

programs with the number of points in the order of hundreds. Creating such large programs manually using a teach pendant would be enormously time-consuming.

Initially, the laser beam was parallel to the surface normal of the forging die. Then, the angle between the surface normal and the laser beam was changed to achieve a collision-free simulation. The angle of impact is measured in relation to the surface normal of the forging die. As a rule of thumb, an angle of impact of fewer than 15° does not affect the result of the LSP process. A simulation that is sufficiently close to the actual LSP process was created, and the testing continued on the physical robotic arm.

6.3.2 Results of testing on a physical robotic arm

Generally, the program from the simulation does not transition directly to the physical robotic arm due to minor differences between the simulated and physical workcell. As a consequence, some tweaks to the robotic arm program were necessary. Therefore, the following program parameters were alternated:

- the laser beam is tied to a user frame – this user frame origin was shifted,
- the forging die was rotated around the J6 robotic arm joint (the J6 joint is responsible for the rotation of the robotic arm flange).

The testing on the physical robotic arm has been successful – the LSP affected area on the surface of the forging die has matched the area to be treated by LSP specified in the simulation. Figure [6.3](#) shows the actual LSP process performed on the forging die.

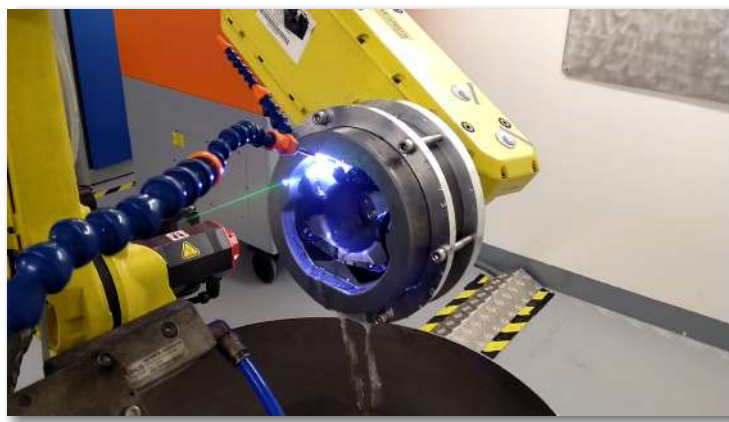


Figure 6.3: LSP process performed on the forging die with plasma emitting from the surface of the material

Chapter 7

Conclusions and future work

Our imagination is the only limit to what we can hope to have in the future.

Charles F. Kettering

American inventor, engineer and businessman

This chapter deals with the conclusions of this graduation thesis and the future work on this graduation thesis. For the sake of clarity, the goals set in the [graduation thesis assignment](#) and the goals set in [Chapter 1 – Introduction](#) are listed separately. The future work also mentions features that could be incorporated into the post processor to improve its functionality.

7.1 Conclusions of the thesis and future work on thesis

Firstly, let us evaluate the goals set in the [graduation thesis assignment](#):

1. get familiar with industrial robotic arm programming methods – goal achieved in [Chapter 2 – Basics of industrial robotics](#),
2. get familiar with the laser shock peening process and identify the inputs, outputs and parameters of this process – goal achieved in [Chapter 3 – Laser shock peening](#)
3. design functions that extend the post processor of your chosen CAM software – goal achieved in [section 5.5.1 – Modifying the FANUC R-30iA post processor methods](#)

4. program the functions that extend the post processor in a higher programming language of your choice and simulate the robotic arm program in a CAM software – goal achieved in [section 5.5.1 – Modifying the FANUC R-30iA post processor methods](#) and [Chapter 6 – Testing of robotic arm program](#).
5. verify and document the functionality of the post processor – goal achieved in [Chapter 6 – Testing of robotic arm program](#).
6. write the graduation thesis in English – goal achieved,
7. prepare the graduation thesis in a problem-based structure corresponding to a scientific document – goal achieved.

Let us recapitulate the goals of this graduation thesis from [Chapter 1 – Introduction](#):

1. creating a simulation of a LSP process in CAM software,
2. developing a post processor for the LSP process,
3. generating a program for the robotic arm controller from simulation,
4. test simulation of LSP process on actual robotic arm controller.

The first goal has been achieved in [section 5.2.1 – Setting up a curve follow project in RoboDK](#) the second goal in [section 5.5.1 – Modifying the FANUC R-30iA post processor methods](#) The third goal has been achieved in [section 5.5.2 – Example output of modified FANUC R-30iA post processor](#) and finally, simulation and testing on a physical robotic arm has been demonstrated in [Chapter 6 – Testing of robotic arm program](#).

This thesis addresses the problem of modifying a FANUC robotic arm controller post processor for LSP applications. One of the main contributions of this thesis is to simplify the generation of robotic arm controller programs. In this thesis, programs are generated using the CAM program RoboDK. The creation of a robotic arm program manually is a tedious process, especially for parts with complex geometries. From an experimental point of view, the contribution lies in testing the programs generated with the help of the CAM program RoboDK.

In the future, the post processor could be improved by incorporating the following features:

- implementation of Remote Tool Centre Point functionality (RTCP). RTCP ensures a constant robotic arm speed with respect to the surface of the sample in case a static tool is used,
- constrain the angular speed of robotic arm joints to a specific limit so that the robotic arm motions will be smoothed.

Bibliography

- [1] K. Ding and L. Ye. *Laser shock peening performance and process simulation*. Woodhead Pub., 2006.
- [2] R. M. White. “Elastic Wave Generation by Electron Bombardment or Electromagnetic Wave Absorption”. In: *Journal of Applied Physics* 34.7 (1963), pp. 2123–2124. DOI: [10.1063/1.1729762](https://doi.org/10.1063/1.1729762).
- [3] Frank Neuman. “Momentum Transfer And Cratering Effects Produced By Giant Laser Pulses”. In: *Applied Physics Letters* 4.9 (1964), pp. 167–169. DOI: [10.1063/1.1754017](https://doi.org/10.1063/1.1754017).
- [4] R. Fabbro et al. “Physical study of laser-produced plasma in confined geometry”. In: *Journal of Applied Physics* 68.2 (1990), pp. 775–784. DOI: [10.1063/1.346783](https://doi.org/10.1063/1.346783).
- [5] Air Force Research Laboratory. *Laser Shock Peening – The Right Technology at The Right Time*. URL: <http://www.dtic.mil/dtic/tr/fulltext/u2/a487687.pdf>. Online; accessed 08-07-2021.
- [6] Y. Sano. *Progress in Laser Peening Technology for Applications to Infrastructure and Energy Systems*. Paper presented at the conference of 4th Int. Conf. on Laser Peening, Madrid. 2013.
- [7] J. Skařupa. *Průmyslové roboty a manipulátory*. Ediční středisko VŠB – TUO, 2007. ISBN: 978-80-248-1522-0.
- [8] Gareth J. Monkman. *Robot grippers*. Wiley-VCH, 2007.
- [9] Alex Owen-Hill. *The Pros and Cons of 5 Robot Programming Methods*. URL: <https://robodk.com/blog/robot-programming-methods-pros-and-cons/>. Online; accessed 28-06-2021.
- [10] *SYSTEM R-30iB and R-30iB MATE Handling Tool Setup and Operations Manual*. 8th ed. FANUC America Corporation. 3900 W. Hamlin Road Rochester Hills, Michigan 48309–3253, 2014.

- [11] *Robot simulation and offline programming*. URL: <https://www.delfoi.com/delfoi-robotics/offline-programming/> Online; accessed 28-06-2021.
- [12] C. Brent Dane et al. “High Power Laser for Peening of Metals Enabling Production Technology”. In: *Materials and Manufacturing Processes* 15.1 (2000), pp. 81–96. DOI: [10.1080/10426910008912974](https://doi.org/10.1080/10426910008912974).
- [13] B.P. Fairand. “Quantitative assessment of laser-induced stress waves generated at confined surfaces”. In: *Journal of Applied Physics* 25.8 (1974), pp. 431–433. ISSN: 0021-8979.
- [14] Marek Bohm et al. “Robotic Arm Human-Machine Interface For Laser Shock Peening Applications”. In: *MM Science Journal* 2019.05 (2019), pp. 3643–3646. DOI: [10.17973/mmsj.2019_12_2019115](https://doi.org/10.17973/mmsj.2019_12_2019115)
- [15] R. Fabbro et al. “Physics and applications of laser-shock processing”. In: *Journal of Laser Applications* 10.6 (1998), pp. 265–279. DOI: [10.2351/1.521861](https://doi.org/10.2351/1.521861)
- [16] Allan H. Clauer and David F. Lahrman. “Laser Shock Processing as a Surface Enhancement Process”. In: *Key Engineering Materials* 197 (2001), pp. 121–144. DOI: [10.4028/www.scientific.net/kem.197.121](https://doi.org/10.4028/www.scientific.net/kem.197.121)
- [17] Xin Hong et al. “Confining medium and absorptive overlay”. In: *Optics and Lasers in Engineering* 29.6 (1998), pp. 447–455. DOI: [10.1016/S0143-8166\(98\)80012-2](https://doi.org/10.1016/S0143-8166(98)80012-2)
- [18] Jan Kaufman. “Influence of Laser Shock Peening on corrosion resistance and stress corrosion cracking of aluminum alloy 5083”. MA thesis. Czech Technical University, 2018.
- [19] Allan H. Clauer, John H. Holbrook, and Barry P. Fairand. “Effects of Laser Induced Shock Waves on Metals”. In: *Shock Waves and High-Strain-Rate Phenomena in Metals* (1981), pp. 675–702. DOI: [10.1007/978-1-4613-3219-0_38](https://doi.org/10.1007/978-1-4613-3219-0_38)
- [20] J. A. Bolger, C. S. Montross, and A. V. Rode. “Shock waves in basalt rock generated with high-powered lasers in a confined geometry”. In: *Journal of Applied Physics* 86.10 (1999), pp. 5461–5466. DOI: [10.1063/1.371546](https://doi.org/10.1063/1.371546)
- [21] Jean-Eric Masse and Gérard Barreau. “Laser generation of stress waves in metal”. In: *Surface and Coatings Technology* 70.2-3 (1995), pp. 231–234. DOI: [10.1016/0257-8972\(95\)80020-4](https://doi.org/10.1016/0257-8972(95)80020-4).

- [22] L. Berthe et al. “Wavelength dependent of laser shock-wave generation in the water-confinement regime”. In: *Journal of Applied Physics* 85.11 (1999), pp. 7552–7555. DOI: [10.1063/1.370553](https://doi.org/10.1063/1.370553)
- [23] D. Devaux et al. “Generation of shock waves by laser-induced plasma in confined geometry”. In: *Journal of Applied Physics* 74.4 (1993), pp. 2268–2273. DOI: [10.1063/1.354710](https://doi.org/10.1063/1.354710)
- [24] C. Brent Dane et al. “High Power Laser for Peening of Metals Enabling Production Technology”. In: *Materials and Manufacturing Processes* 15.1 (2000), pp. 81–96. DOI: [10.1080/10426910008912974](https://doi.org/10.1080/10426910008912974).
- [25] Vaccari J.A. “Laser shocking extends fatigue life”. In: *American Machinist* 62–4. (1992).
- [26] Allan H. Clauer. “Laser Shock Peening, the Path to Production”. In: *Metals* 9.6 (2019), p. 626. DOI: [10.3390/met9060626](https://doi.org/10.3390/met9060626).
- [27] HiLASE centre. *Laser system 'BIVOJ'*. 2021. URL: https://www.hilase.cz/wp-content/uploads/bivoj_visualization-no-text-1170x484.jpg. Online; accessed 28-06-2021.
- [28] B. Saumyabrata. “DiPOLE: a 10 J, 10 Hz cryogenic gas cooled multi-slab nanosecond Yb:YAG laser”. In: *Optics Express* 23.15 (2015), pp. 19542–19551. ISSN: 1094-4087.
- [29] *Litron LPY SERIES High and Ultra-High Energy Q-Switched Nd:YAG Lasers*. URL: <https://litron.co.uk/wp-content/uploads/2020/11/LPY-Series-November-2020.pdf> Online; accessed 02-04-2021.
- [30] *LPY600 Laser Series Operators handbook version*. 1st ed. Litron Lasers. 8 Consul Road, Rugby, Warwickshire, CV21 1PB England, 2004.
- [31] *FANUC M-20iA/20M*. URL: <https://www.robots.com/robots/fanuc-m-20ia-20m> Online; accessed 02-04-2021.
- [32] *FANUC I/O Unit-MODEL A. Connection and maintenance manual*. 4th ed. FANUC America Corporation. 3900 W. Hamlin Road Rochester Hills, Michigan 48309–3253, Apr. 2005.
- [33] RobotWorx. *FANUC M-20iA/20M*. 2021. URL: https://www.robots.com/images/robots/Fanuc/M-Series/FANUC_M-20iA_20M_cutout.png Online; accessed 28-06-2021.

- [34] Dan Mihai. *RoboDK: An Offline Programming and 3D Simulation Software for Industrial Robots*. 2015. URL: <https://www.smashingrobotics.com/robodk-industrial-robot-offline-programming-simulation-software/> Online; accessed 28-06-2021.
- [35] Alex Owen-Hill. *9 Powerful RoboDK Features You Might Not Know About*. 2020. URL: <https://robodk.com/blog/powerful-robodk-features/> Online; accessed 28-06-2021.
- [36] RoboDK. *RoboDK documentation: Getting started – Select a robot*. 2021. URL: <https://robodk.com/doc/en/Getting-Started-Select-robot.html> Online; accessed 28-06-2021.
- [37] RoboDK. *RoboDK documentation: Basic guide*. 2021. URL: <https://robodk.com/doc/en/Basic-Guide.html> Online; accessed 28-06-2021.
- [38] RoboDK. *RoboDK documentation: Getting started – New project*. 2021. URL: <https://robodk.com/doc/en/Getting-Started-Select-robot.html> Online; accessed 28-06-2021.
- [39] RoboDK. *RoboDK documentation: RoboDK Add-In for SolidWorks*. 2021. URL: <https://robodk.com/doc/en/Plugin-SolidWorks.html#PluginSolidWorks> Online; accessed 28-06-2021.
- [40] RoboDK. *RoboDK documentation: Collision Detection*. 2021. URL: <https://robodk.com/doc/en/Collision-Avoidance.html> Online; accessed 05-07-2021.
- [41] RoboDK. *RoboDK documentation: Post Processors*. 2021. URL: <https://robodk.com/doc/en/Post-Processors.html#PostProcessor> Online; accessed 28-06-2021.
- [42] FANUC America. *FANUC Roboguide simulation software*. 2021. URL: <https://www.fanucamerica.com/products/robots/robot-simulation-software-FANUC-ROBOGUIDE> Online; accessed 05-07-2021.
- [43] *FANUC Robotics SYSTEM R-30iA Controller KAREL Reference Manual*. 7th ed. FANUC America Corporation. 3900 W. Hamlin Road Rochester Hills, Michigan 48309-3253, 2010.
- [44] RoboDK. *RoboDK documentation: Robot Manufacturing – Robot Machining project*. 2021. URL: <https://robodk.com/doc/en/Robot-Machining.html#MachiningProject> Online; accessed 05-07-2021.

- [45] RoboDK. *RoboDK documentation: Robot Manufacturing – Curve Follow Project*. 2021. URL: <https://robodk.com/doc/en/Robot-Machining.html#CurveFollow>. Online; accessed 05-07-2021.
- [46] RoboDK. *RoboDK documentation: RoboDK API*. 2021. URL: <https://robodk.com/doc/en/RoboDK-API.html#RoboDKAPI>. Online; accessed 28-06-2021.
- [47] RoboDK. *RoboDK documentation: RoboDK API – Python API*. 2021. URL: <https://robodk.com/doc/en/RoboDK-API.html#PythonAPI>. Online; accessed 05-07-2021.
- [48] RoboDK. *RoboDK documentation: Post Processors – Reference*. 2021. URL: <https://robodk.com/doc/en/Post-Processors.html#RefPostProc>. Online; accessed 05-07-2021.
- [49] RoboDK. *RoboDK API documentation: Post Processors – Post Processor Methods*. 2021. URL: <https://robodk.com/doc/en/PythonAPI/postprocessor.html#post-processor-methods>. Online; accessed 05-07-2021.
- [50] RoboDK. *RoboDK API documentation: Post Processors – Select a post processor*. 2021. URL: <https://robodk.com/doc/en/Post-Processors.html#SelectPost>. Online; accessed 05-07-2021.
- [51] R. Bernstein. *python-decompile3*. Version 1.0. July 2021. URL: <https://github.com/rocky/python-decompile3>. Online; accessed 05-07-2021.

Appendices

Appendix A

Content of the enclosed CD/DVD

This thesis is accompanied by a CD/DVD with the following directory structure.

Graduation thesis in \LaTeX 2e

RoboDK project

RoboDK FANUC post processor source code

3D CAD model of forging die

Bohm_AP_2021_2022.pdf

FANUC R-30iB robotic arm controller program for LSP process

Appendix B

Used software and software libraries

L^AT_EX 2_ε [⟨https://miktex.org⟩](https://miktex.org)

RoboDK [⟨https://robodk.com/⟩](https://robodk.com/)

Python [⟨https://www.python.org/⟩](https://www.python.org/)

decompile3 [⟨https://github.com/rocky/python-decompile3/⟩](https://github.com/rocky/python-decompile3/)

SolidWorks Professional 2019 [⟨https://www.solidworks.com/⟩](https://www.solidworks.com/)

VSCodium [⟨https://vscodium.com/⟩](https://vscodium.com/)

Thonny [⟨https://thonny.org/⟩](https://thonny.org/)

FANUC Roboguide [⟨https://www.fanucamerica.com/⟩](https://www.fanucamerica.com/)

The software from the above list is either freely available or its license is owned by the HiLASE centre, Dolní Břežany, Za Radnicí 828, where the author worked at the same time and created this graduate thesis.

Appendix C

Graduation thesis timetable

Activity	Duration	Closing date	Fulfilled
Ordering RoboDK software	2 weeks	13.09.2021	26.09.2021
Designing electrical connection between controller and laser	2 weeks	27.10.2021	10.11.2021
Building of electrical connection between controller and laser	2 weeks	11.10.2021	24.10.2021
Creating of RoboDK project	3 weeks	25.10.2021	14.11.2021
Modifying RoboDK post processor	4 weeks	15.11.2021	12.12.2021
Graduation thesis: chapter Introduction	2 weeks	13.12.2021	27.12.2021
Graduation thesis: complete text	15 weeks	30.04.2022	30.04.2022

Appendix D

English-Czech glossary

English	Czech
cavitation erosion	kavitační opotřebení
collision detection	detekce kolizí
decompiler	dekompilátor
dielectric breakdown	průraz dielektrika
end effector	efektor
fatigue life	únavová životnost
flashlamp	výbojka
forging die	kovací zápustka
Gaussian temporal profile	gaussovský časový průběh laserového impulsu
GUI (Graphical User Interface)	grafické uživatelské rozhraní
laser beam spot size	velikost laserového svazku
laser power density	intenzita záření laseru
laser pulse temporal profile	časový průběh laserového impulsu
laser shock peening	laserové vyklepávání
laser-induced plasma	laserem generované plazma
Nd:YAG	kryстал YAG dopovaný neodymem
offline programming	offline programování
post processor	postprocesor
Q-switching	Q-spínání

English**Czech**

residual stress

zbytkové napětí

robotic arm gripper

uchopovač robotického ramene

robotic arm teach pendant

manuální ovladač robotického ramene

shock wave

rázová vlna

tool center point

středový bod nástroje

yield strength

mez průtažnosti