KOKKUVÕTE

Kokkuvõtte alustuseks tasub välja tuua, et kõik peatükis 2 esitatud tööülesanded ja eesmärgid said täidetud ning üldises pildis võib lahendust pidada edukaks.

Peatükis 3 käidi üle ning tutvustati kõiki töös kasutatud tarkvarasid: MotoSim, RoboDK ja Python ning põhjendati nende valikut alternatiivide ees. Nagu peatükis mainitud, on käesolevas töös kasutatud tarkvarad valitud osaliselt olude sunnil nagu Yaskawa robotist tulenev MotoSim-i kasutamine. Ülejäänud osas on aga valikud tehtud lähtudes pragmaatilisusest, kasutajasõbralikkusest ning võimalustest. Püüton sai valitud STL failide töötlemiseks mõeldud teegi kättesaadavuse tõttu ning RoboDK püütoni toetamise ja ka ilma tasuta saadud litsentsi tõttu. Need ei ole kindlasti ainsad võimalused, mida sarnase ülesande lahendamiseks kasutada saab, kuid antud hetkel tundusid parimad käesoleva töö jaoks ning tänuväärselt on võimaldanud jõuda ka soovitud lahenduseni.

Peatükk 4 lahkas detailsel tasandil töö lahendust ning seletas lahti iga tegevuse, sammu ja koodi, mida kasutati müüriplokkide asukohtadest funktsionaalse roboti tööprogrammini jõudmiseks. Lahenduse toimimiseks sai RoboDK-s välja arendatud ülikooli Yaskawa „You Teach Me" robotjaama imiteeriv simulatsioonikeskkond ning kasutajaloodud püütonskript, mis kasutades RoboDK API programmiliidest lubas püütoni koodiga järjestada roboti jaoks töökäskluseid. Töö tõenäoliselt kõige keerulisemaks osaks osutuski püütonskripti loomine, mis hõlmas CSV sisendfailist plokkide asukohtade lugemist ning õiges formaadis programmi esitamist. Veel olulisem oli API funktsioone kasutades programmi peatsükli loomine, sest just see lõi seose simulatsioonikeskkonna elementide ja koodis olevate muutujate ja asukohtade vahel. Samuti sõltus kogu robotprogrammi loomine ja kõigi liikumiste ja tegevuste järjekord üksnes sellest tsüklist, mis tähendas et kogu tegevus tuli koodi kirjutades väga hoolikalt algusest lõpuni läbi mõelda. Peatüki lõpus demonstreeriti lahenduse ootuspärast toimimist Yaskawa ametlikus MotoSim tarkvaras.

Lõputöö peatükis 5 anti tehtud tööle majanduslik hinnang, kõrvutades lahendust standardse käsitsi roboti programmeerimise metoodikaga. Nagu peatükis välja toodud, on RoboDK litsentsi omandamine ligi kaks korda odavam kui Yaskawa ametlik tarkvara MotoSim. Vahe oleks veel suurem kasutades ABB robotit, sest RobotStudio ja selle pakettide tellimus maksaks üksi sama palju iga aasta kui terve RoboDK litsentsi ühekordne makse. Pidades silmas, et RoboDK toetab sisuliselt iga robotitootja roboteid, on selle omandamine majanduslikult põhjendatud. Samuti sai majanduslikku tasuvust hinnatud tööle kuluva aja seisukohast. Jõuti järeldusele, et vähekogenud ja alustava

roboti programmeerija korral võib RoboDK lahendus tööle kuluvat aega teatud juhtudel vähendada kuni 12 korda, mõnel juhul isegi rohkem. See tuleb kasuks nii roboti programmeerimisteenuse sisse ostjatele kui ka tööandjatele, sest lahendus vähendab tööle kuluvat aega ja tõstab tootlikust. Seega on lahendusel ka selge majanduslik kasu.

Kogu tehtud tööd tagantjärgi analüüsides on seejuures ka ideid, mille abil annaks süsteemi täiustada. Lisaks töös mainitud plokkide lähenemispunktidele oli lahenduse arendamise käigus vahepeal kasutuses veel lisanduv ruumipunkt *midway_point* ehk vahepunkt. See oli lisanduv punkti, mida robot läbis peale klotsi üles korjamist ning müüri juurde liikudes ning oli mõeldud taaskord kokkupõrgete vältimiseks. Töö käigus kaotati see punkt aga ära lähtudes sellest, et õigesti paika pandud plokkide lähenemispunktid tegid sama töö ära ning teiseks selle punkti läbimine aeglustas roboti tööd. Tuleviku perspektiivis seejuures kui peaks juhtuma, et robotiga soovitakse laduda kõrgemaid müüre, tasub kaaluda selle punkti taas kasutusele võtmist ning selle läbimist müüri kõrgemate kihtide ladumisel. Vastasel korral võib tekkida olukord, kus müür on nii kõrge, et oma asetusest sõltuvalt võib robotkäsi klotsitorni ja müüri vahel liikudes vaatamata plokkide lähenemispunktidele juba laotud müürile otsa sõita.

Sama teemat edasi arendades tasub mõelda RoboDK-s kokkupõrke olukordade kontrolli sisaldamise peale, sest praegusel hetkel tuleb seda kontrollida inseneril või roboti operaatoril. Ka MotoSim on varustatud tööriistadega kontrollimaks kokkupõrkeolukordi, kuid MotoSim-is kontrolli ülesehitamine ei ole mõistlik ei ajaliselt ega rahaliselt, sest nagu töös mainitud tuleb iga programmi simulatsioon MotoSim-is nii-öelda käsitsi üles ehitada. Teiseks on kunagi tulevikus RoboDK süsteemi eduka töötamise korral võimalik MotoSim sootuks ahelast välja jätta, liikudes otse roboti peale ning hoides seejuures kokku litsentsile ja selle toe uuendamisele kuluva rahasumma.

Vaatamata kõigele võib isiklikul hinnangul tehtud tööga rahule jääda. Seda eelkõige seetõttu, et lähteülesande tasandil oli tegu võrdlemisi keerulise probleemiga, mis eeldas mitme keeruka süsteemi tundmist ja seejärel automatiseerimist. Samuti oli töö jaoks taustauuringut teha äärmiselt keeruline, sest sarnaseid lahendusi võib öelda et lihtsalt ei ole tehtud. Sellest olenemata on raske töö ja sihikindlusega jõutud välja lahenduseni, mis vähemalt prototüübina toimib täpselt nii nagu palutud.

SUMMARY

To begin with, it shall be pointed out that all tasks and goals as outlined in chapter 2 have been completed and reached, and generally the work can be considered successful.

Chapter 3 went over and introduced all the software used for the work: MotoSim, RoboDK and Python as well as provided argumentation for their selection over the alternatives. As it is pointed out in the chapter, some of the software has been chosen for the lack of an alternative, such as the use of MotoSim due to using a Yaskawa robot. The rest of the choices however have been based on pragmatism, user friendliness and opportunities. Python got chosen for the availability of the STL format manipulation library and RoboDK for its support of Python and user scripts, as well as the free to us license. These are definitely not the only options for such a task but for the time being and with the given conditions seemed the best option for the task at hand, and thankfully have enabled me to reach the desired outcome.

Chapter 4 provided a detailed explanation to the work's technical solution and went over every action, step and code which was used to turn a list of brick locations into a functional robot work program. For the solution, a simulation environment was created inside RoboDK to imitate the university's Yaskawa "You Teach Me" robot station, as well as a custom Python script which by using RoboDK's API allowed for creating and ordering commands for the robot using Python code. Creating the Python script turned out to be likely the hardest part of the solution, as it involved reading data from the CSV input file and presenting it to the program in a specific and correct format. Even more challenging was creating the main program cycle using the API functions because this was the crucial link between the physical objects and coordinate systems in the simulation and the variables inside the Python code. The entire generation of the robot work program, every command and their order depended on this cycle, which meant that the entire process had to be considered very carefully from beginning to end. In the end of the chapter a demonstration was provided, which showed the solution's expected behaviour in Yaskawa's official MotoSim software.

In the 5th chapter an economic evaluation was given to the work, comparing the developed solution side by side with the standard methodology of programming robots by hand. As it was outlined in the chapter, obtaining RoboDK license is nearly two times cheaper than Yaskawa's official software MotoSim. The difference would be even greater when using an ABB robot because the license subscription with all the required packages would cost nearly as much every year as the entire one-time purchase of RoboDK

license. Due to RoboDK supporting robots from nearly all robot manufacturers, obtaining its license is economically justified. The economic impact was also considered from the time and work force perspective. It was concluded that in case of an inexperienced and starting robot programmer, the RoboDK solution may under some circumstances reduce the time spent on the task up to 12 times, under some conditions perhaps even more. This is beneficial both for the clients of robot programming services as well as employers because the solution lowers the time required for the task and therefore promotes productivity. As a result of all that, the economic grounds for the solution have been proven as well.

Analyzing the work done in hindsight, there are however ideas for improving the solution. In addition to the building block approach points as mentioned in the thesis, some time along the way during development there was an extra point used in the simulation called the "midway_point". It was an additional space point which the robot would move through after picking up the block and before moving to the wall. It was designed with the same purpose of preventing and avoiding any collisions between the robot and the wall. During development however this point was left out based on the conclusion that correctly defined approach points already took care of that danger. Additionally, moving the robot through this point slowed down the robot program itself. In future perspective however, if there is ever a desire to lay down higher walls, it would be wise to consider reintroducing this point and passing through it when laying down the higher layers of the wall. Otherwise, one might end up in a situation where the wall is so high that depending on its position the robot arm may still collide with it regardless of the approach points.

Following the same logic, it may be wise to consider developing a collision detection mechanism inside the RoboDK solution because currently that will have to be checked by the engineer or robot operator. MotoSim as well is equipped with the tools for collision detection, however building up the check in MotoSim is not viable neither from a time or financial point of view, because as pointed out in the thesis, every single simulation in MotoSim will have to be built up by hand. Additionally, in future perspective should the RoboDK solution prove to be working well, it may be possible to leave MotoSim out of the loop altogether and move on directly to the robot, all while sparing the money for the license and updating its support.

Despite everything, from a personal point of view the work done has been satisfactory. This is so mainly because on a basic level the task posed a rather complicated problem, which expected the knowledge of several sophisticated systems and following their

automation. Secondly, doing research for the task proved to be more than challenging because one could say projects similar to this simply have not been done. Despite all that, through hard work and determination a solution has been achieved, which at the very least as a prototype works exactly as desired.