KOKKUVÕTE

Antud lõputöö eesmärgiks oli arendada 3D müüri mudelist telliste koordinaatide ja orientatsioonide tuvastamise tarkvara. Tarkvara kirjutati programmeerimiskeeles Python. Valminud programmil on graafiline kasutajaliides (joonis 2.4.2), programm on pakitud kokku EXE failiks ja tarkvara väljastab leitud asukohad CSV failina. Loodud tarkvara täidab kõiki talle püstitatud eesmärke.

3D failitüüpide uuringust lähtuvalt otsustati valida sisendfailiks STL failiformaat. Neutraalne failiformaat STL lubab müüri mudelit luua paljude erinevate CAD programmidega. Lähtudes vabavaraliste 3D failide lugemisega tegelevate teekide ja programmide võrdlusest, otsustati STL faile lugeda kasutades Pythoni teeki NumPy-STL.

Loodud programm tuvastab STL formaadis 3D müüri mudelist risttahuka kujuliste telliste koordinaadid ja orientatsioonid. Tarkvara töötab nii sirgete kui ka nurki ja kaari sisaldavate müüridega (joonis 2.6.1). Programm töötab ka juhul, kui müür sisaldab erineva suurusega telliseid. Loodud GUI lubab kasutajal valida väljund- ja sisendfaile, muuta väljundfailide sisu ning lubab näha ka tuvastatud koordinaatide ja orientatsioonide eelvaadet.

Tuleviku edasiarendustes tuleks täiendada risttahukate tuvastamise funktsiooni selliselt, et tuvastamine töötaks ka siis, kui kaks identset risttahukat on sarnase küljega tihedalt teineteise vastas. Sellise olukorra puhul ei tee pragune algoritm vahet, kus üks tellis algab ja teine lõppeb. Praktikas ei ole see eripära probleemiks, sest tellismüürides laotakse tellised reeglina sidematerjalist vuugiga. Juhul kui tellised peavad ilmtingimata vahetus kontaktis olema, tuleb telliste vahele jätta vähemalt ühe mikromeetrine vahe. Samas aga selleks, et tellismüüri oleks üldse võimalik tööstusrobotiga laduda, peab vuuk telliste vahel olema vähemalt millimeeter.

Edasiarendustes saaks ka tõsta programmi töökiirust ning suurenda toetatavate eri kujuga telliste ja toetatavate 3D failiformaatide hulka. Pragune programm töötab vaid risttahuka kujuliste tellistega, kuid tellistel võivad olla ka süvendid, augud, kaarjad või kiilukujulised küljed ning väljaulatuvad servad. Toetatavate 3D failiformaatide hulga suurendamiseks tuleks valida mõni teine töös mainitud 3D faile lugevatest teekidest, mis toetab nii IGES, STEP kui ka STL formaate, nagu näiteks OCCT või PythonOCC. Et tõsta programmi töökiirust, tuleks programmi muuta selliselt, et see suudaks töötada paralleelselt mitme protsessori tuuma peal. Peaks ka uurima, kui suurt ajavõitu saaks kui osa arvutustest loovutataks graafikaprotsessorile.

SUMMARY

The aim of this thesis was to develop software for identifying locations and orientations of bricks from a 3D wall model. The software was developed using the Python programming language and compressed into an EXE file. It has a graphical user interface (Figure 2.4.2) and outputs the locations and orientations as a CSV file.

Based on the study of 3D file types, it was decided to use STL files as input. The neutral file format STL allows for the creation of a wall model using a variety of CAD programs. Based on the study of open source libraries and programs for reading 3D files, it was decided to read STL files using the Python library NumPy-STL.

The created program detects the locations and orientations of rectangular bricks from a 3D wall model in STL format and works with both straight walls and walls containing corners and arcs (Figure 2.6.1). The program works even if the wall contains bricks of different sizes. The GUI allows the user to easily select output and input files, change the contents of the output files, and visually preview the detected brick locations and orientations.

Future developments should enhance the cuboid detection function so that it works even when two identical cuboids are perfectly aligned. In such situations, the current algorithm does not distinguish where one brick starts and the other ends. In practice, this peculiarity is not a problem, as bricks do not usually overlap in brick walls. If it is necessary for them to be aligned, a gap of at least one micrometer must be left between the bricks. If an industrial robot is used for brick laying, most robots require a side gap of at least one millimeter between the bricks regardless.

Enhancements can also be made to increase the speed of the program, the number of supported brick shapes, and the number of supported 3D file formats. The current program only works with rectangular bricks, but bricks can also have recesses, holes, protruding edges, round corners, and can be wedge-shaped. To increase the number of supported 3D file formats, one of the other 3D file reading libraries that supports IGES, STEP, or STL formats should be selected, such as OCCT or PythonOCC. For the software to be able to calculate the brick locations of an entire house within a reasonable amount of time, the program must be rewritten so that it can run in parallel on multiple processor cores. It should also be considered whether some of the calculations could be handed over to the graphics processor and whether that would speed up the program.