Általános információk, a diplomaterv szerkezete

A diplomaterv szerkezete a BME Villamosmérnöki és Informatikai Karán:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervező nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések
9. A tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása
10. A megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek
11. Esetleges köszönetnyilvánítások
12. Részletes és pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következő oldaltól kezdődő LaTeXdiplomatervsablon dokumentum tartalma.

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2,5 cm, baloldalon 1 cm-es kötéssel). Az alapértelmezett betűkészlet a 12 pontos Times New Roman, másfeles sorközzel, de ettől kismértékben el lehet térni, ill. más betűtípus használata is megengedett.

Minden oldalon – az első négy szerkezeti elem kivételével – szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerű fejezetenként külön számozni (pl. 2.4. ábra, 4.2. táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetőleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztő segítségével írják le (A LaTeX ehhez kézenfekvő megoldásokat nyújt).

Az irodalomjegyzék szövegközi hivatkozása történhet sorszámozva (ez a preferált megoldás) vagy a Harvard-rendszerben (a szerző és az évszám megadásával). A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben (a LaTeX-sablon a BibTeX segítségével mindezt automatikusan kezeli). Minden publikáció a szerzők után a következő adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóiratok címét csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internetes hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Fontos:

- A szakdolgozatkészítő / diplomatervező nyilatkozata (a jelen sablonban szereplő szövegtartalommal) kötelező előírás, Karunkon ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhető!
- Mind a dolgozat, mind a melléklet maximálisan 15 MB méretű lehet!

Jó munkát, sikeres szakdolgozatkészítést, ill. diplomatervezést kívánunk!

FELADATKIÍRÁS

A feladatkiírást a tanszéki adminisztrációban lehet átvenni, és a leadott munkába eredeti, tanszéki pecséttel ellátott és a tanszékvezető által aláírt lapot kell belefűzni (ezen oldal *helyett*, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell beleszerkeszteni ezt a feladatkiírást.

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Development of Navigation Methods in Dynamic Environment for an Intelligent Model Car

Bachelor's Thesis

Author
Soma Veszelovszki

Advisor
Domokos Kiss

May 13, 2019

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Veszelovszki Soma*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2019. május 13.

Veszelovszki Soma
hallgató

Kivonat

A diplomatervezési feladat egy, a tanszéken futó kutatási projekthez kapcsolódik, amelyben egy intelligens autó tesztplatform létrehozása a cél. A platform alapja egy csökkentett méretű (kb. 1:3 méretarányú) autómodell, melyet egy távirányítható játékautóból alakítunk át. A tesztplatform létrehozásának célja különböző navigációs algoritmusok fejlesztésének és valós környezetben történő tesztelésének elősegítése.

A sikeres navigáció és az ütközések elkerülése érdekében fontos, hogy a jármű valós időben detektálni tudja a környező objektumokat, és megfelelően reagáljon rájuk. Ennek elősegítése érdekében az autóplatformon elhelyezésre került egy-egy vízszintes síkban pásztázó lézeres távolságszkenner (lidar) a jármű elején és hátulján.

A feladat keretében ezekre a szenzorokra építve kell megvalósítani mozgó objektumok felismerését, azok méretének és sebességvektorának becslésével együtt. Fontos, hogy a jármű el tudja különíteni a statikus és a mozgó akadályokat egymástól. További feladat egy olyan akadályelkerülési módszer megvalósítása az autón, amely az objektumok mozgását is figyelembe véve hozza meg a megfelelő navigációs döntést minden időpillanatban.

A megvalósított algoritmusok működését mind szimulált, mind valós környezetben szükséges ellenőrizni. A valós tesztelés történhet a tanszéki járműplatformon, vagy egy kisméretű, egyedileg felépített modellautón is.

Abstract

The theme of the thesis is a subtask of a research project at the Department of Automation and Applied Informatics, with the purpose of designing an intelligent car testing platform. The platform is based on a decreased-size (1:3 scale) remote control car model, that has been modified to support self-driving program control. The platform aims to support the development of different navigation algorithms and helping the testing in real environment.

For a successful navigation and obstacle avoidance, the detection of the surrrounding objects and reacting accordingly are essential. For the purpose of detection, two horizontal distance scanning sensors (lidars) have been placed on the car - one on the front and one on the back.

Part of the task is to implement the detection of the moving obstacles and estimate their sizes and speed vectors. It is important for the car to be able to separate static and moving objects from each other. The other part is developing an obstacle-avoidance method, implemented as an application on the car, that takes the moving objects into consideration while making navigational decisions at every time step.

The implemented algorithms need to be tested both in simulation and in real environment, that may be executed on the department's vehicle platform or on a small-scale, custom-build model car.

Chapter 1

Introduction

Nowadays, self-driving cars gain more and more attention, both their technology and their effect on people's daily routines and their lives overall. Several articles are published every year about how these cars will change the way people commute to work, visit their friends or go on a family vacation. These articles often point out the decrease of the number of accidents, an optimized load of traffic and thus a reduced fuel consumption as the major advantages of this technology-to-come.

The release date of these cars, however, is still a matter of question. In 2015, Mark Fields, president and CEO of Ford at the time estimated their first fully autonomous car in 2020. 2 years later, at CES 2017 Nvidia announced that with the partnership of Audi they would develop a self-driving vehicle - also, in market by 2020. Both of these statements are considered too ambitious guesses today, as there is a high probability that we need to wait until at least 2025 for reliable fully autonomous vehicles to hit the roads. Claiming that there are no viable signs of these cars in traffic would be a false statement, as there are several companies who have been testing there vehicles on public roads for the last years, but these prototypes are very far from reliable products yet. The company that seems to be ahead of the competition in this race is Tesla. Their self-drving software is already in their products, but it still needs millions of hours of testing and the responsibility is still the driver's if an accident happens.

But why is this delay of release dates? One possible answer is that manufacturerers have the tendency to exaggerate when asked about new products, and thus the users' need and the other competitors development speed urged them to make such estimations they could not keep up with. Another theory is that the companies at that time didn't acknowledge how many hours and kilometers of testing is needed to finalize a self-driving product.

However, the spreading of autonomous vehicles is blocked by several legislational and technological obstacles. As this thesis describes an engineering problem an its solution, I will reflect on the technological blockers that both car manufacturers and other self-driving software developer companies need to face. Just to list some of these problems, we can name reliable object detecion, error insensitive, robust decision-making, fast, well-tuned physical control, and to meet the safety and quality requirements set by the market, determinant scheduling and redundancy throughout the whole software are also essential.

Chapter 2

Mapping

2.1 Method

This chapter decribes the process of building a separate static and a dynamic map (containing non-moving and moving obstacles, respectively) based on radial distance measurements (in this case provided by a LIDAR). Isolating the static and the moving obstacles from each other has its difficulties but also its advantages. First of all, the implemented local track planner calculates safer, more optimized paths if it is informed of both the static and the dynamic obstacles along the path. Secondly, SLAM (Simultaneous localization and mapping) algorithms work better if the their input contains only static objects in the space, because they build an internal map of the world. Passing detections of moving obstacles to a SLAM algorithm may lead to worse localization quality, as they may ruin this map.

As a first subtask of the mapping project, I checked if any implementation is avaiable already, that can handle dynamic objects. The only possible candidate was gmapping, which is a popular ROS package, used in a wide variety of applications that require map-building and localization. Its SLAM algorithm takes LIDAR measurements as its input and generates an occupancy grid (a 2D map) of the car's environment. By using this map it is able to make corrections to the car's odometry-based position and orientation, which is usually inaccurate. I tried out the package, and the result maps were promising, the generated map was insensitive to the car's longitudinal movements and its rotations. But unfortunately, gmapping's SLAM does not support dynamic objects. See Section 2.4.1 for further details. Therefore, gmapping could not be used as the producer of the static map. But that didn't mean it couln't be used for its second feature, localization. Note, that the mapping implementation I made is not a SLAM algorithm, it is not able to make corrections to the car's pose. So for that purpose, I still needed the help of gmapping, which proved to be very reliable at localization. But the static map-building needed to be implemented internally.

In order to create two disjunct maps, one static and one dynamic, the key element of the process is the separation of the moving and non-moving obstacles of the measured points. After determining these two disjunct set of points, the maps can be converted to any desired or required format. Static maps are usually published as occupancy grids, while dynamic obstacles need to present information about their speed vector. Occupancy grids do not group the grid points according to their probabilities, therefore they do not

know about the obstacles' borders and areas[1]. Dynamic obstacles however can be either represented as separate points with their own speed vectors, or groups of points, each groups having one speed vector. I chose the latter representation, thus publishing groups that contain a set of points (all the points, ideally) of the same obstacle. This way there is a one-to-one relationship between moving obstacles and groups. The separation and grouping methods of the mapping process is shown on diagram 2.1.

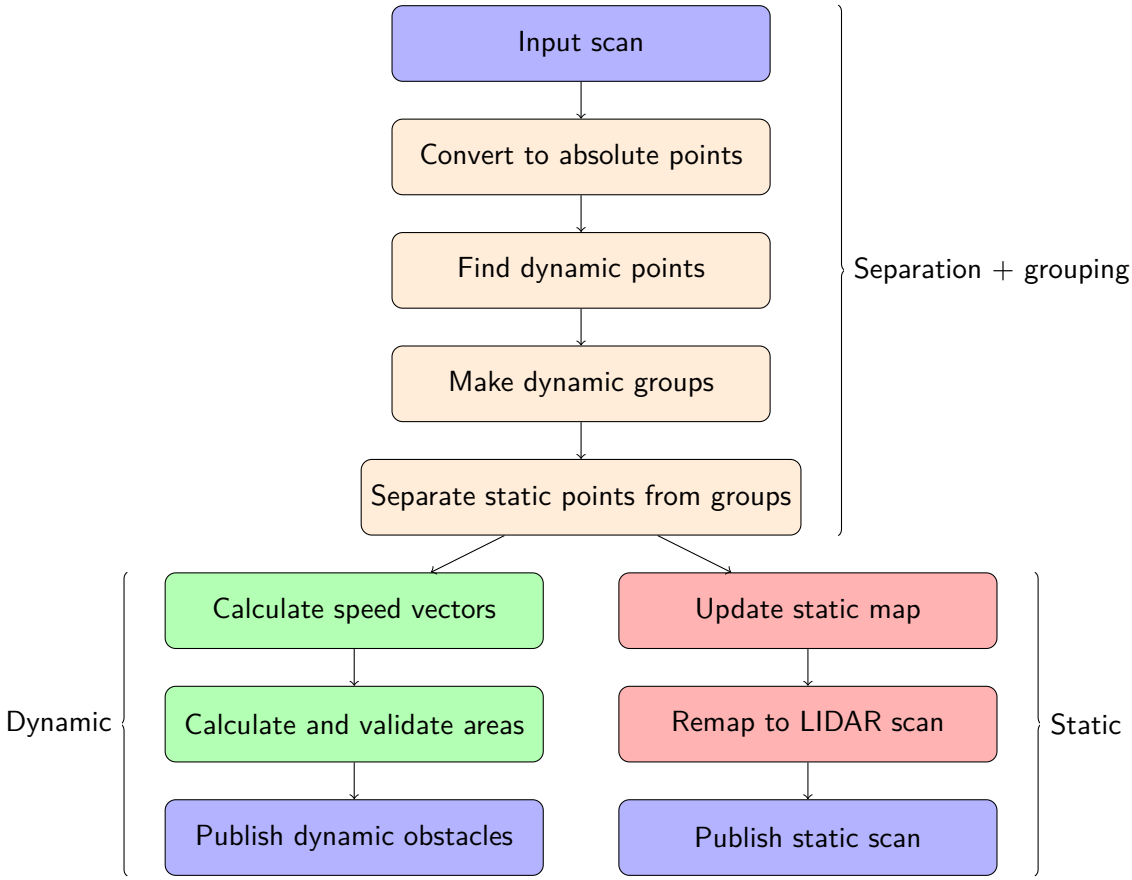


Figure 2.1: Mapping method

The diagram consists of 3 subgraphs - these are also marked on the diagram. The first, and most important is the separation and grouping of dynamic points. The second section describes the additional calculations that need to be done for the dynamic obstacles, and consists the documentation for the message structure definining these dynamic obstacles. The third section is about the static map that is built from the static points. The next sections are going to explain these subgraphs in detail.

2.2 Separation and grouping

This section describes the method of separating dynamic and static points of the input scan. This step is essential in the pipeline of creating two disjunct maps.

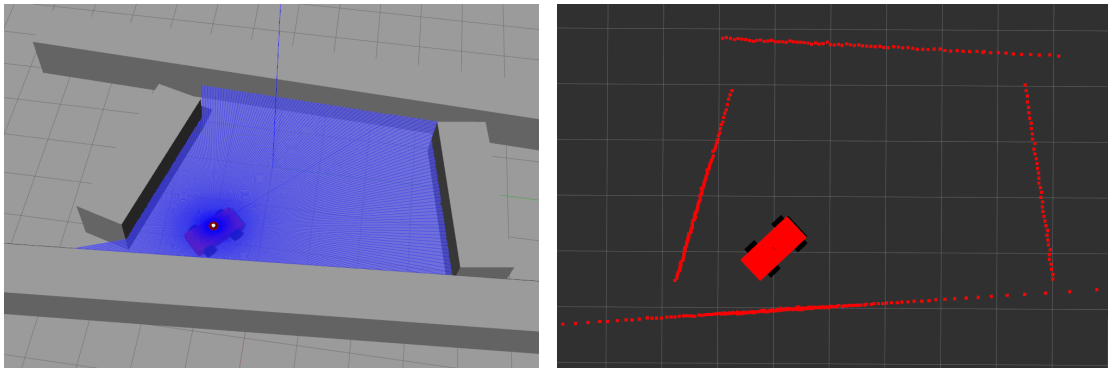
[1]In this project, 2D LIDARs were used, therefore the measured objects were seen as 2D shapes that have areas, not 3D objects that have volumes

2.2.1 Input scan

The input of the mapping algorithm is a 2D LIDAR scan, consisting of radial distance measurements. Two types of LIDARs were used in the project, RPLidar A1 and RPLidar A2. Both types have the following specifications:

Scan rate	10 Hz
Sample rate	8000 samples/sec
Distance resolution	0.2 centimeters
Angular resolution	1°
Detection range	12 meters

The devices proved to be reliable, and for a project of this volume, their frequencies, resolutions and ranges were adequate. Their output after each measurement sequence is an array of radial distances, that can be visualized easily using rviz.



(a) Gazebo simulation **(b)** rviz visualization

Figure 2.2: LIDAR scan

2.2.2 Absolute points

For static map-building, absolute points[2] are needed in the space, and static-dynamic point separation also uses absolute points, so firstly, these radial distances need to be transformed. However, the internal static map and the static-dynamic separation use different coordinate systems. In order to understand the need for this, let's take a look at figure 2.3.

As the figure shows, there are 3 coordinate frames that are important to the current case. The *odom* frame is the car odometry's coordinate system. It is calculated from values measured on the vehicle, such as servo position and speed. The *scanner* frame is the LIDAR's coordinate system. The transformation between *odom* and *scanner* is basically the pose of the LIDAR, relative to the car. The *map* frame is the output of gmapping's localization. Basically, gmapping takes the car odometry and the LIDAR scans as its input, and corrects the odometry using SLAM. As a result the difference between frames *map* and *odom* will increase with time.

The internal static map in my imlpementation uses this corrected *map* frame as the base for its points, so that its error is minimized. But unfortunately, the static-dynamic separation

[2] Absolute points are not relative to the car, but to a fix base point.

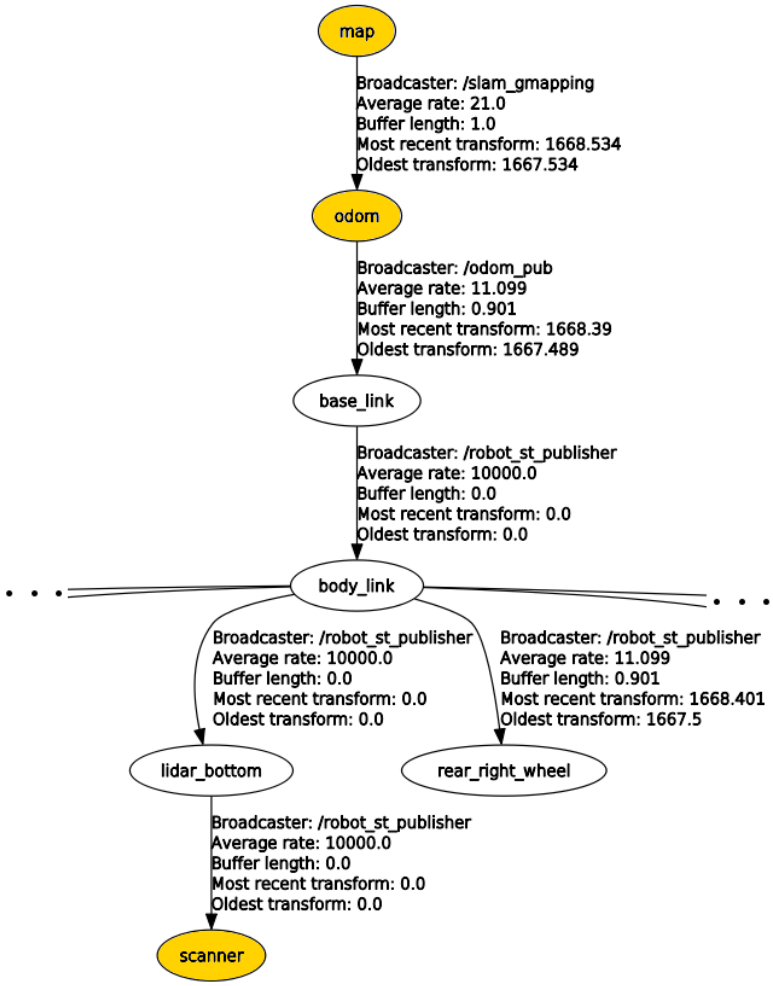


Figure 2.3: Coordinate frames

method cannot use this frame. The reason is that this frame does not get updated on every new scan, but with a much slower frequency. Therefore there are 'jumps' in the *map* frame, which would cause false dynamic point detections (see Section 2.2.3). To avoid this undesired situation, the static-dynamic separation uses the *odom* frame as its base, which is more continuous than the *map* frame.

Therefore, two transformations are needed for each input point. The transformations are calculated using tf, which maintains the relationship between coordinate frames in a tree structure (see figure 2.3) buffered in time, and makes the transformation of points, vectors, etc possible at any desired point in time.

2.2.3 Dynamic points

The next step is similar to a creating a subtraction image in image processing, where subtracting two images, taken from the same position but at a different time, results in an image that amplifies the movements between the snapshots. The aim of this algorithm unit is basically the same: selecting the points from the input that are likely to be part of a moving mass. This is done by finding the points among the current measurements that have not been present in the previous ones.

This step is best explainable in practice. Let's assume that the position and orientation of the LIDAR is fix, and one object (e.g. a car) in the detected area is moving. Figure 2.4 shows this situation. On the image, the pale contour represents the previous pose of the object, and the current state is blue.

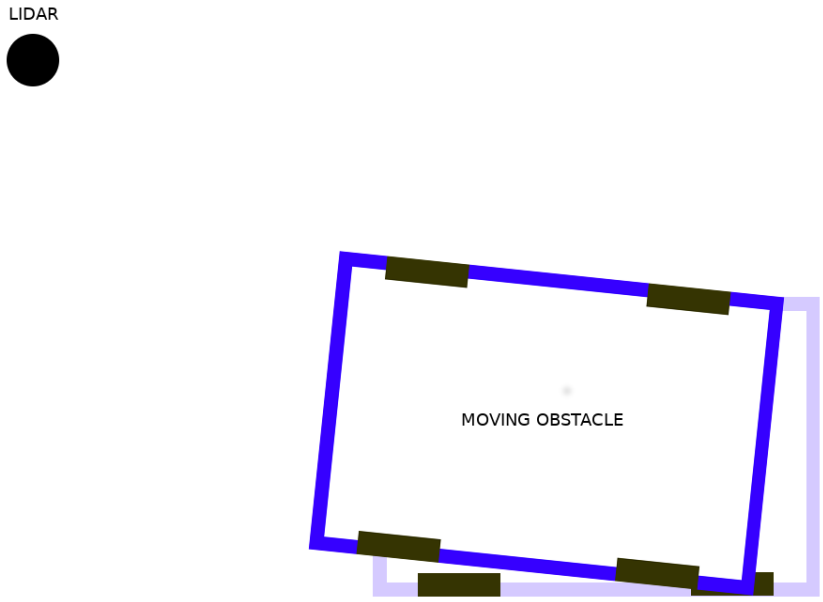


Figure 2.4: The obstacle is moving

Figure 2.5 shows how the LIDAR detects the moving object in two different timesnaps. I marked the points corresponding to the current position with red color, and the ones of the previous measurement are pale red. As it is suggested in the figure, the measurements are far from ideal, the detected points are noisy.

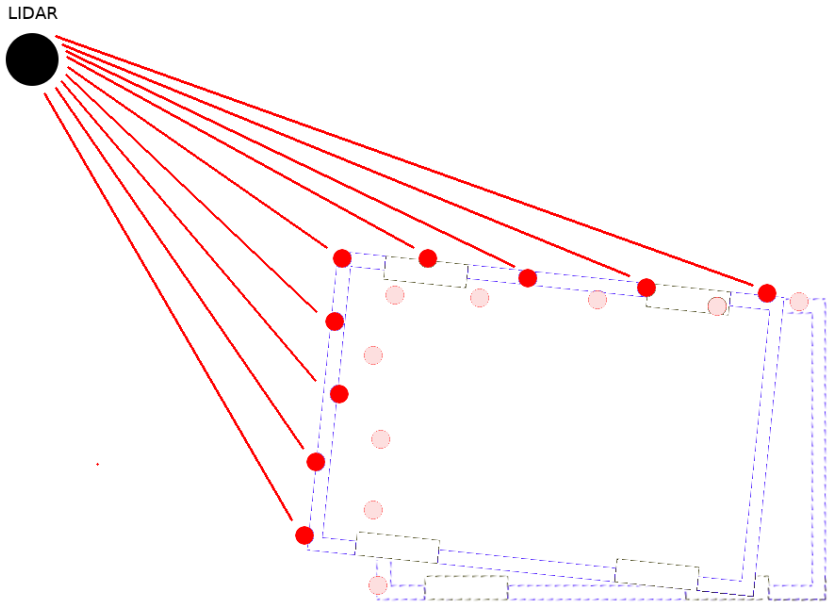


Figure 2.5: The detected points of the moving obstacle

To understand the method of dynamic point detection, I removed the LIDAR, its virtual rays and the obstacle's contour from the image, thus leaving leaving only the measured points of the two timesnap. The result, which is basically a time-buffered array of 2D

points, is presented on figure 2.6. The algorithm iterates through each point in the current measurement, and checks if any point in the previous measurements[3] is within its compliance radius[4].

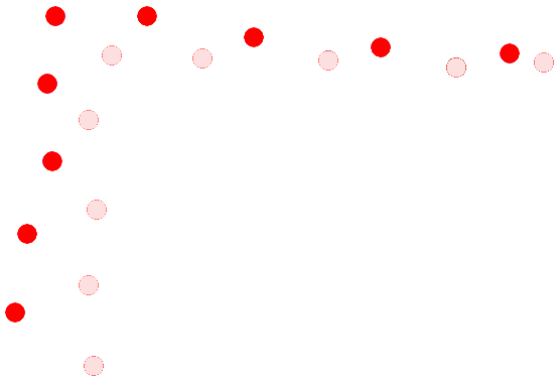


Figure 2.6: The detected points of the two timesnaps

The measured points with their compliance radiuses are presented in figure 2.7. The possible dynamic points, that have no points from the previous measurement within their radius, are marked with blue, and their compliance radius with yellow. The possible static points are marked with red, along with their radiuses. Note, that these points are *possible* dynamic and static points. The final classification will be preceded be multiple filtering mechanisms and point grouping, but this is the base for finding dynamic objects.

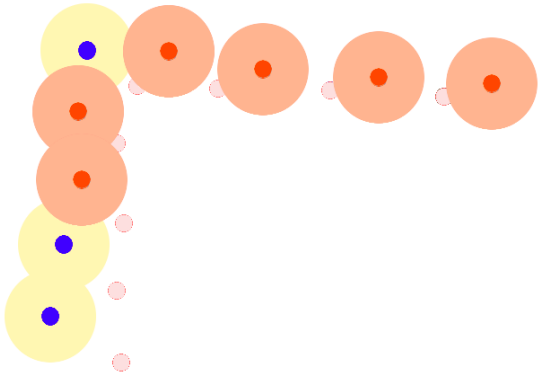


Figure 2.7: The compliance radiuses and the dynamic points

Several conclusions can be drew from figure 2.7. The one, and probably most important is that with right parameterizing (number of 'look-up' measurements, compliance radius, etc) this method amplifies the positive[5] and negative[6] changes of the scans. The second remark is that the method does not detect all the points of a moving object. However, due to measurement noise false detections may happen, non-moving points may be marked as

[3]The number of measurements to 'look up' is configurable.

[4]The compliance radius is a maximum allowed distance between measurements representing the same physical point but measured in different timesnaps. If the distance between two measurements is greater than this value, the measurements are assumed to represent different points of the space. The compliance radius is calculated for each measurement separately, and it is proportional to the distance of the LIDAR and the measured point.

[5]Previously a distant background was detected, but now a closer object appears.

[6]Previously a close object was detected, but now it disappears, and the distant background becomes visible.

dynamic. Therefore, this separation needs to be followed by several filtering and validating steps.

2.2.4 Dynamic groups

2.2.5 Separation of static points and dynamic groups

2.3 Dynamic obstacles

After the dynamic groups have been separated from the static points, additional information needs to be calculated for them, so that the local track planner algorithm can use their data for its obstacle-avoidance feature.

2.3.1 Speed vectors

2.3.2 Area validation

2.3.3 Publishing dynamic obstacles

2.4 Static points

Static points used for two separate purposes. The first one is building a static map, the second is localization with the help of gmapping.

2.4.1 Static map

After separating the dynamic groups from the scan points, only the static points are left. These points do not change their positions with time[7], so they can be added to the static map.

For static map-building, gmapping has already been mentioned as a candidate, but unfortunately, its algorithm does not recognize changes in the environment. In practice, if an object has been detected and placed in its map, it will not be erased from there, even after the object has moved away. Figures 2.8 and 2.9 demonstrate the problem.

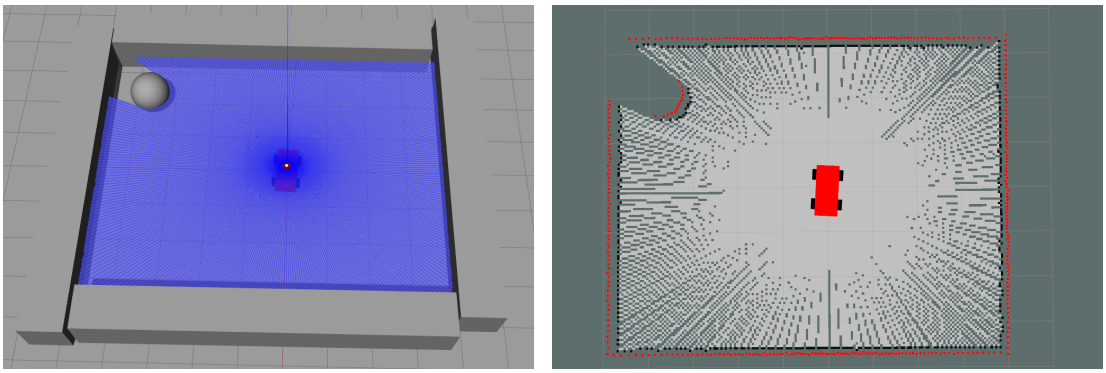
Figure 2.8 shows a situation where the ball is standing still. A the mapping algorithm of gmapping successfully creates a map (in the form of an occupancy grid[8]) that marks the place of the ball as occupied. However, when the ball starts moving (see figure 2.9), the map does not change, because the algorithm cannot handle moving obstacles.

2.4.2 Remapping to LIDAR scan

2.4.3 Publishing static scan

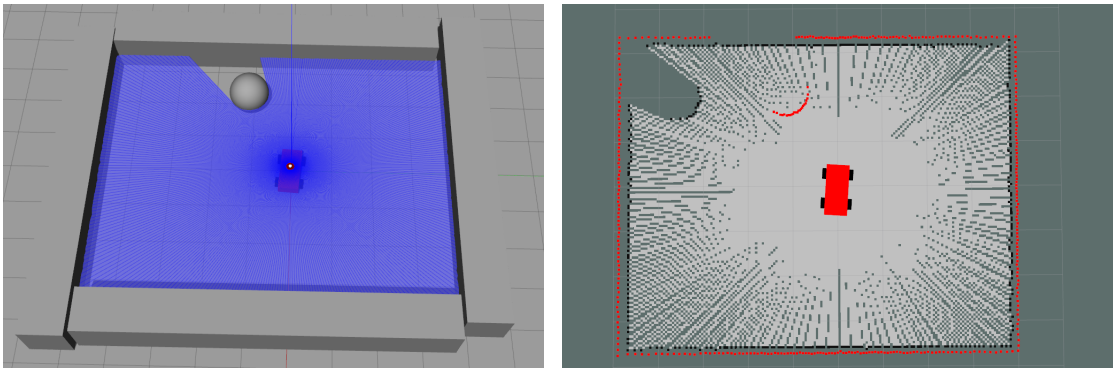
[7] Except those obstacles that start moving only after they have already been detected as sets of static points. But these objects do not ruin the quality of the map, either.

[8] An occupancy grid represents a map in an evenly spaced field of probability values representing if the grid points are occupied by an obstacle



(a) Gazebo simulation **(b)** Map created by gmapping

Figure 2.8: Initial state - ball is still



(a) Gazebo simulation **(b)** Map created by gmapping

Figure 2.9: Moving state - ball has changed position

Chapter 3

LaTeX-eszközök

3.1 A szerkesztéshez használatos eszközök

Ez a sablon TeXstudio 2.8.8 szerkesztővel készült. A TeXstudio egy platformfüggetlen, Windows, Linux és Mac OS alatt is elérhető LaTeX-szerkesztőprogram számtalan hasznos szolgáltatással (figure 3.1). A szoftver ingyenesen letölthető[1].

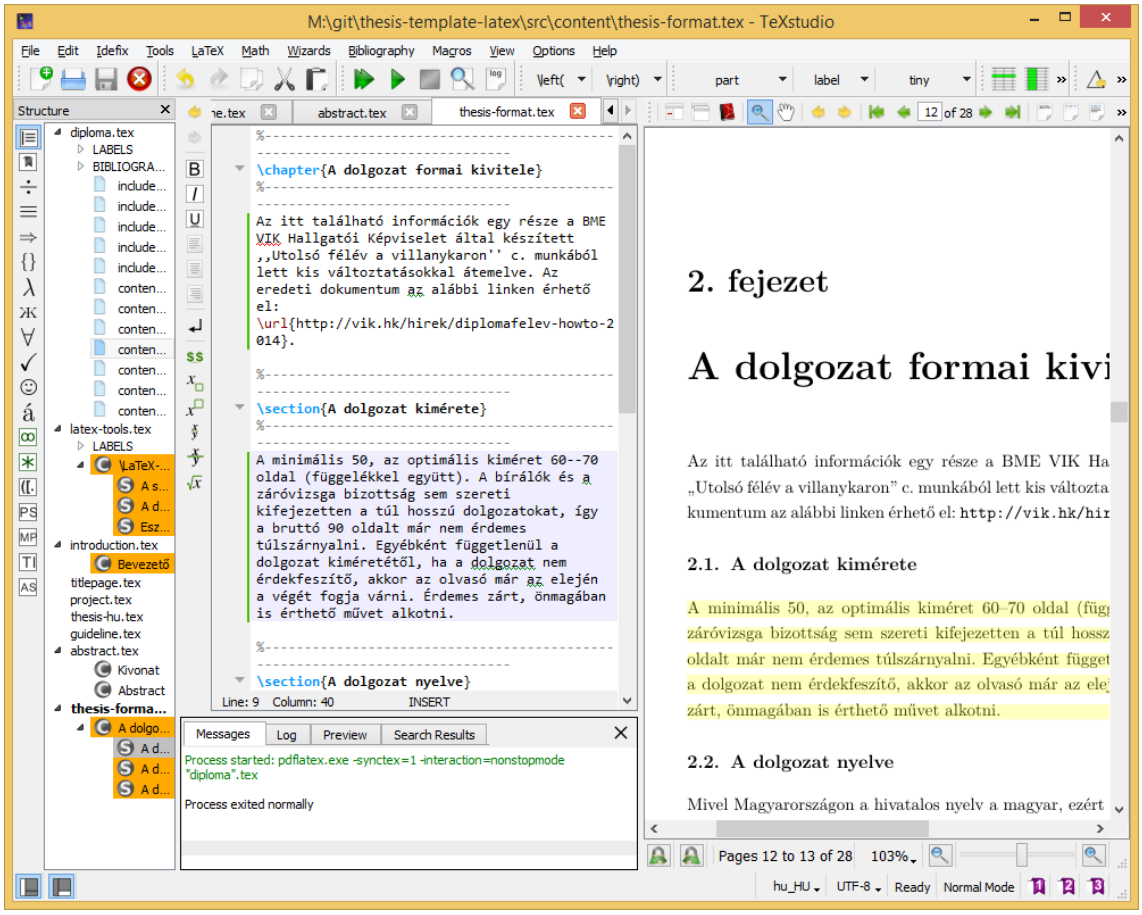


Figure 3.1: A TeXstudio LaTeX-szerkesztő.

[1] A TeXstudio hivatalos oldala: http://texstudio.sourceforge.net/

A TeXstudio telepítése után érdemes még letölteni a magyar nyelvű helyesírásellenőrző-szótárakat hozzá. A TeXstudio az OpenOffice-hoz használatos formátumot tudja kezelni. A TeXstudio beállításainál a `General` fülön a `Dictionaries` résznél tudjuk megadni, hogy melyik szótárat használja.

Egy másik használható Windows alapú szerkesztőprogram a LEd[2] (LaTeX Editor), a TeXstudio azonban stabilabb, gyorsabb, és jobban használható.

3.2 A dokumentum lefordítása Windows alatt

A TeXstudio és a LEd kizárólag szerkesztőprogram (bár az utóbbiban DVI-nézegető is van), így a dokumentum fordításához szükséges eszközöket nem tartalmazza. Windows alatt alapvetően két lehetőség közül érdemes választani: MiKTeX (`http://miktex.org/`) és TeX Live (`http://www.tug.org/texlive/`) programcsomag. Az utóbbi működik Mac OS X, GNU/Linux alatt és Unix-származékokon is. A MiKTeX egy alapcsomag telepítése után mindig letölti a használt funkciókhoz szükséges, de lokálisan hiányzó TEX-csomagokat, míg a TeX Live DVD ISO verzóban férhető hozzá. Ez a dokumentum TeX Live 2008 programcsomag segítségével fordult, amelynek DVD ISO verziója a megadott oldalról letölthető. A sablon lefordításához a disztribúcióban szereplő `magyar.ldf` fájlt a `http://www.math.bme.hu/latex/` változatra kell cserélni, vagy az utóbbi változatot be kell másolni a projekt-könyvtárba (ahogy ezt meg is tettük a sablonban) különben anomáliák tapasztalhatók a dokumentumban (pl. az ábra- és táblázat-aláírások formátuma nem a beállított lesz, vagy bizonyos oldalakon megjelenik alapértelmezésben egy fejléc). A TeX Live 2008-at még nem kell külön telepíteni a gépre, elegendő DVD-ről (vagy az ISO fájlból közvetlenül, pl. DaemonTools-szal) használni.

Ha a MiKTeX csomagot használjuk, akkor parancssorból a következő módon tudjuk újrafordítani a teljes dokumentumot:

```
$ texify -p thesis.tex
```

A `texify` parancs a MiKTex programcsomag `miktex/bin` alkönyvtárában található. A parancs gondoskodik arról, hogy a szükséges lépéseket (fordítás, hivatkozások generálása stb.) a megfelelő sorrendben elvégezze. A `-p` kapcsoló hatására PDF-et generál. A fordítást és az ideiglenes fájlok törlését elvégezhetjük a sablonhoz mellékelt `manual_build.bat` szkript segítségével is.

A TEX-eszközöket tartalmazó programcsomag binárisainak elérési útját gyakran be kell állítani a szerkesztőprogramban, például TeXstudio esetén legegyszerűbben az `Options / Configure TeXstudio... / Commands` menüponttal előhívott dialógusablakban tehetjük ezt meg.

A PDF-LATEX használata esetén a generált dokumentum közvetlenül PDF-formátumban áll rendelkezésre. Amennyiben a PDF-fájl egy PDF-nézőben (pl. Adobe Acrobat Reader vagy Foxit PDF Reader) meg van nyitva, akkor a fájlleírót a PDF-néző program tipikusan lefoglalja. Ilyen esetben a dokumentum újrafordítása hibaüzenettel kilép. Ha bezárjuk és újra megnyitjuk a PDF dokumentumot, akkor pedig a PDF-nézők többsége az első oldalon nyitja meg a dokumentumot, nem a legutóbb olvasott oldalon. Ezzel szemben például az egyszerű és ingyenes Sumatra PDF nevű program képes arra, hogy a megnyitott dokumentum megváltozását detektálja, és frissítse a nézetet az aktuális oldal megtartásával.

[2] A LEd hivatalos oldala: `http://www.latexeditor.org/`

3.3 Eszközök Linuxhoz

Linux operációs rendszer alatt is rengeteg szerkesztőprogram van, pl. a KDE alapú Kile jól használható. Ez ingyenesen letölthető, vagy éppenséggel az adott Linux-disztribúció eleve tartalmazza, ahogyan a dokumentum fordításához szükséges csomagokat is. Az Ubuntu Linux disztribúciók alatt például legtöbbször a `texlive-*` csomagok telepítésével használhatók a LaTeX-eszközök. A jelen sablon fordításához szükséges csomagok (kb. 0,5 GB) az alábbi paranccsal telepíthetők:

```
$ sudo apt-get install texlive-latex-extra texlive-fonts-extra texlive-fonts-recommended
    texlive-xetex texlive-science
```

Amennyiben egy újabb csomag hozzáadása után hiányzó fájlra utaló hibát kapunk a fordítótól, telepítenünk kell az azt tartalmazó TeX Live csomagot. Ha pl. a `bibentry` csomagot szeretnénk használni, futtassuk az alábbi parancsot:

```
$ apt-cache search bibentry
texlive-luatex - TeX Live: LuaTeX packages
```

Majd telepítsük fel a megfelelő TeX Live csomagot, jelen esetben a ‘texlive-lualatex‘-et. (Egy LaTeX csomag több TeX Live csomagban is szerepelhet.)

Ha gyakran szerkesztünk más LaTeXdokumentumokat is, kényelmes és biztos megoldás a teljes TeX Live disztribúció telepítése, ez azonban kb. 4 GB helyet igényel.

[1]

```
sudo apt-get install texlive-full
```

Acknowledgements

Bibliography

[1] Gábor Jeney. Hogyan néz ki egy igényes dokumentum? Néhány szóban az alapvető tipográfiai szabályokról, 2014. http://www.mcl.hu/~jeneyg/kinezet.pdf.

Appendix

A.1 A TeXstudio felülete

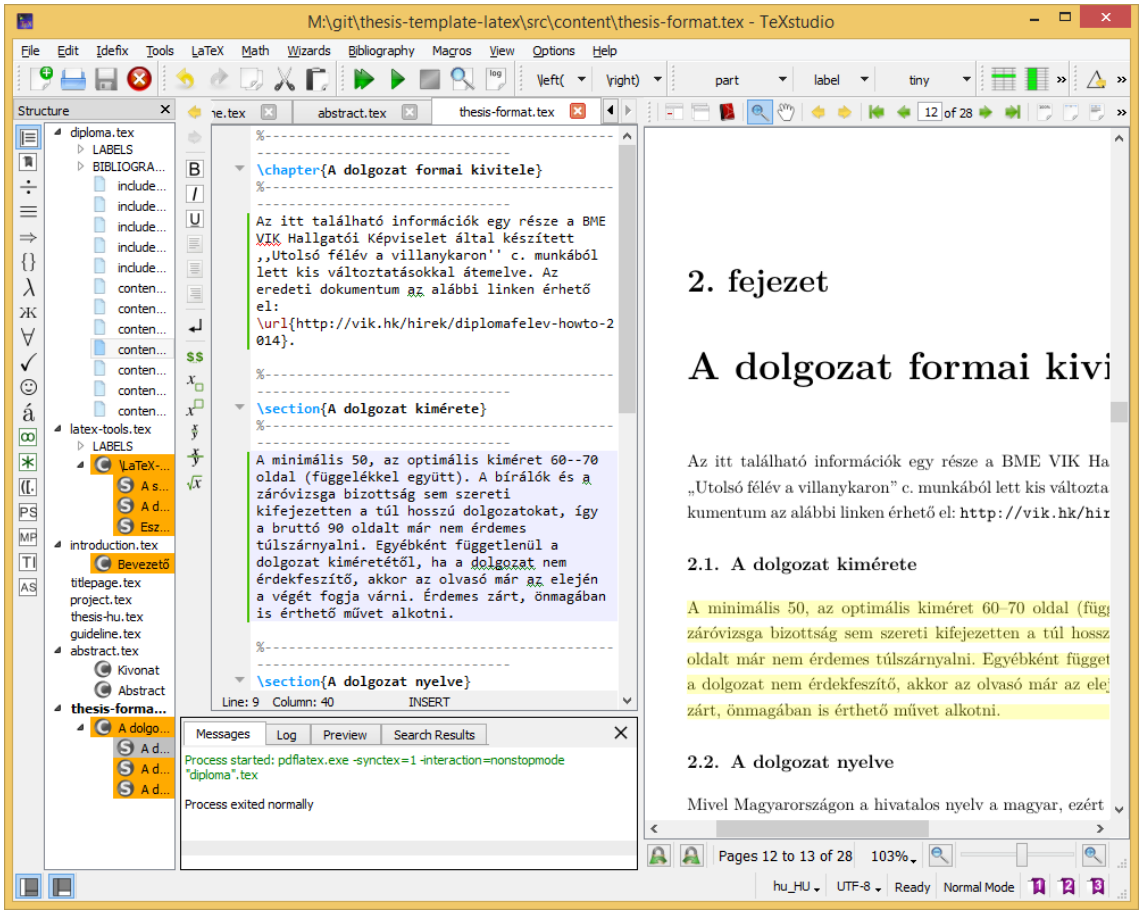


Figure A.1.1: A TeXstudio LaTeX-szerkesztő.

A.2 Válasz az „Élet, a világmindenség, meg minden” kérdésére

A Pitagorasz-tételből levezetve

$$c^2 = a^2 + b^2 = 42. \tag{A.2.1}$$

A Faraday-indukciós törvényből levezetve

$$\operatorname{rot} E = -\frac{dB}{dt} \qquad \longrightarrow \qquad U_i = \oint\limits_{\mathbf{L}} \mathbf{Edl} = -\frac{d}{dt} \int\limits_{A} \mathbf{Bda} = 42. \tag{A.2.2}$$