

Fachhochschule Dortmund
No idea of the Department
Das Institut für die Digitalisierung von Arbeits- und Lebenswelten (IDiAL)

Developing a Rover-Following Application for APPSTACLE

RESEARCH PROJECT

Author
Daniel Paredes

Advisor
Robert Höttger

December 4, 2018

Contents

Abstract	3
1 Introduction	4
1.1 APPSTACLE	4
1.2 Eclipse Kuksa	4
1.3 In-vehicle Platform	6
1.4 Automotive Grade Linux - AGL	6
1.5 Rover Services	6
1.6 Rover Application	6
1.7 Related Work	6
2 Design and Implementation	7
2.1 Use case description	7
2.2 Requirements	7
2.3 Camera calibration and the pinhole model	8
2.4 Extrinsic Parameters	10
2.5 Rotation matrix to Euler angles	10
2.6 Use case implementation details	11
3 Results and Discussion	13
3.1 Results	13
4 A Markdown-sablon használata	14
4.1 Ábrák és táblázatok	14
4.2 Felsorolások és listák	14
4.3 Képletek	15
4.4 Irodalmi hivatkozások	16
4.5 A dolgozat szerkezete és a forrásfájlok	18
5 Summary	20
Acknowledgements	21
A Appendix	26
A.1 Calibration of the picam in raspbian	26

A.2	Answer to the “Life, the Universe, and All” questions	26
-----	---	----

Abstract

This document is a skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. The skeleton was implemented in Markdown and can be compiled with Pandoc, using the T_EX Live and or the MiK_TE_X L^AT_EX compiler.

Chapter 1

Introduction

The introductory part contains the analysis of the diploma dissertation, its historical history, the justification of the task (description of the motivation), the solutions so far and the summary of the student's solution.

According to the introductory custom, it is closed with the structure of the diploma, that is, with a brief description of which chapter it deals with.

1.1 APPSTACLE

APPSTACLE or *open standard APplication Platform for carS and TrAnsportation vehiCLEs* is a project that aims to establish a standard car-to-cloud connection, open for external applications and the use of open source software wherever possible without compromising safety and security [11].

No idea what else I should write about APPSTACLE

1.2 Eclipse Kuksa

The result of APPSTACLE project is *Eclipse Kuksa* and it provides the complete tooling stack for the connected vehicle domain [10]. It provides the in-vehicle platform, the cloud platform, and an app development IDE as shown in figure 1.1.

It is possible to collect, store and analyze data through the different kuksa layers of the in-vehicle platform. This platform runs on top of *Automotive Grade Linux* or AGL which is an open-source project from The Linux Foundation. The goal of AGL is to develop a GNU/Linux based operating system and a framework for automotive applications [5].

The development of eclipse Kuksa plug-ins or applications and the deployment of them can be done using the web-browsed based IDE known as *Eclipse Che*. In other words, a complete toolchain is available through *Eclipse Che* which allows not only a fast, but also an independent platform development.

In addition, the cloud platform is built on top of other eclipse frameworks such as Eclipse Hono used in telemetry applications and Eclipse Ditto used to create a digital rover twin; and also provides the Kuksa app-store, so users could download an app and deploy it directly in their rovers.

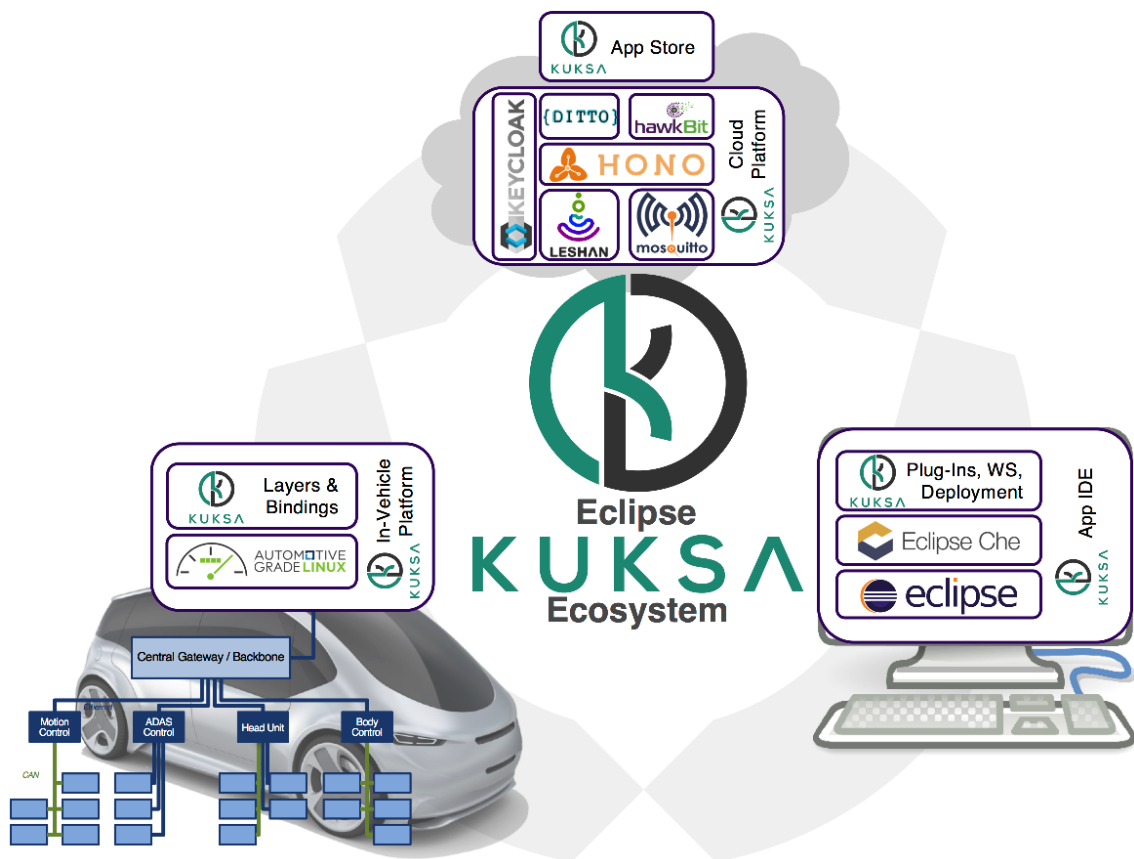


Figure 1.1: Eclipse KUKSA Ecosystem [10]

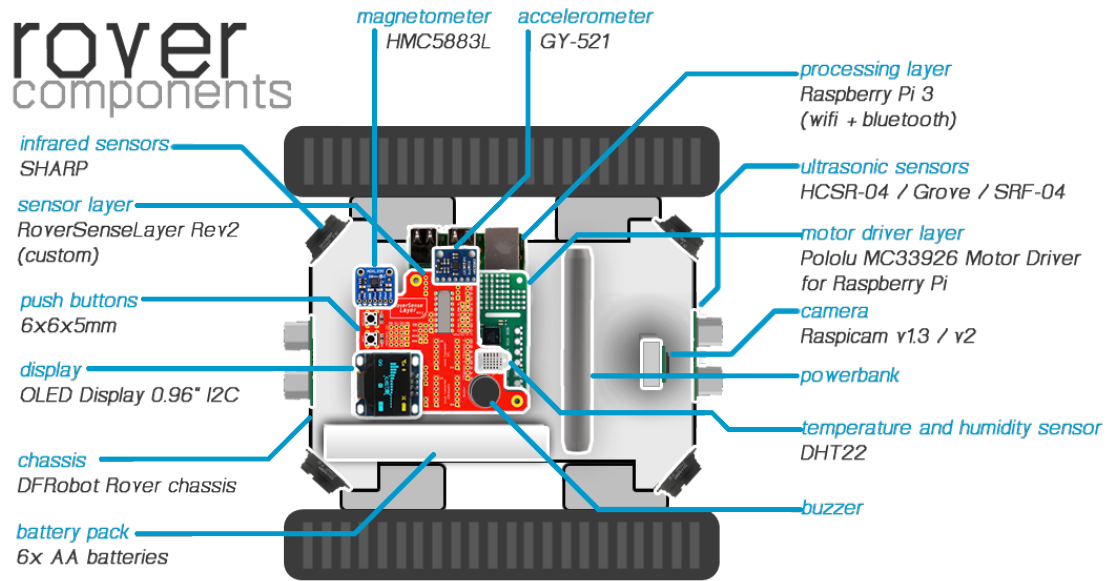


Figure 1.2: Rover Components [1]

1.3 In-vehicle Platform

As it was mentioned in the introduction, the rover vehicle is based on a raspberry pi 3 that can run *Raspbian Jessie* or *AGL* as Operating system. For this use case the follower runs a *Raspbian Jessie* image. The available hardware

1.4 Automotive Grade Linux - AGL

1.5 Rover Services

1.6 Rover Application

1.7 Related Work

Chapter 2

Design and Implementation

2.1 Use case description

In this research project we develop a use case for rover-apps. For the use case we are using two *Rovers*, a **Rover Leader** and a **Rover Follower**. Hereinafter, we will only use **Leader** or **Follower** to refer to them. The leader has a visual marker, the follower should detect it, estimate the angle β and distance d with respect to the leader as is shown in figure 2.1, and follow the leader.

This chapter will be focused on the development of the behavior of the follower because is the only rover that must be completely autonomous. The following sections will describe requirements, main hardware parts involved and software tools required to implement follower's behavior, camera calibration and pose estimation theory, and software implementation details.

2.2 Requirements

Still have no idea how to write them - The system is the follower, but it doesn't actually matter right now.

- Functional requirements?
 - The follower should detect a visual marker.
 - The follower should estimate the distance and angle to the marker.
 - The follower should steer based on the estimated values.
 - The follower should be completely autonomous.
- Software requirements?
 - The follower should run Raspbian as operating system.
 - The follower code should use OpenCV library.
 - The follower code should be based on rover-app libraries.
 - The follower code should be maintainable.
 - The follower code should be reusable.
 - The follower code should be easy to read and understand.

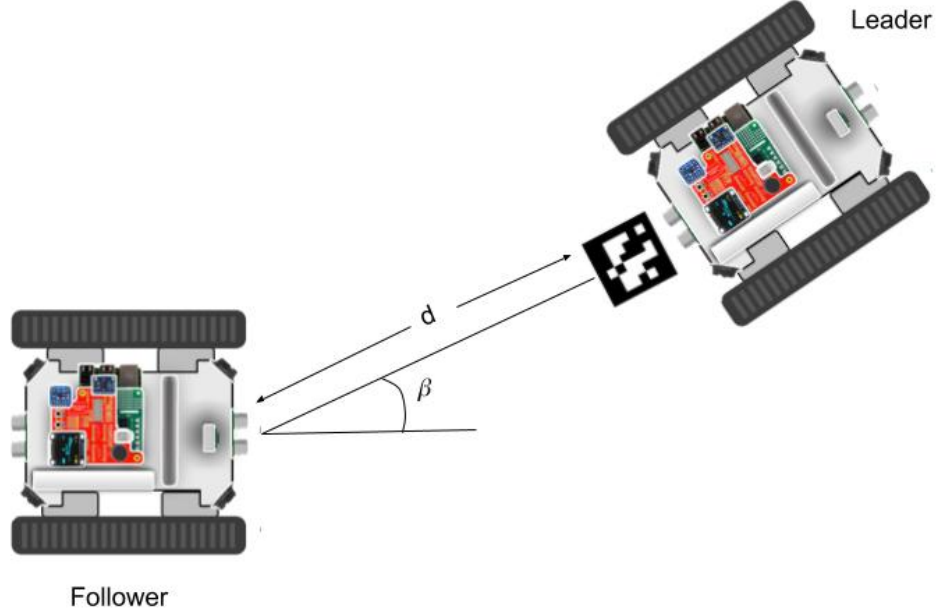


Figure 2.1: Rover Use Case

2.3 Camera calibration and the pinhole model

If you hold that box in front of you in a dimly lit room, with the pinhole facing some light source you see an inverted image appearing on the translucent plate [4]. In figure 2.2, a 3D object (pyramid) is projected first on a scene plane, and then on the image plane. Each point in the scene plane or *world frame* will have its correspondence in the image plane or *camera frame*. The distance from the pinhole to the image plane is called focal length.

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images [15]. The mathematical model of a pinhole camera can be derived using linear algebra and the visual representation shown in figure 2.2.

Let's denote a 2D point $\hat{\mathbf{m}} = [x, y, 1]^T$, a 3D point $\hat{\mathbf{M}} = [X, Y, Z, 1]^T$, there exists a camera projection matrix \mathbf{P} such that $\hat{\mathbf{m}} = \mathbf{P}\hat{\mathbf{M}}$.

$$\hat{\mathbf{m}} = \mathbf{P}\hat{\mathbf{M}} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]\hat{\mathbf{M}} \quad (2.1)$$

The camera intrinsic matrix \mathbf{A} contains information about the internal parameters of the camera: focal length, image sensor format and principal point or image center. The coordinates of the principal point is described by (x_0, y_0) , α_x and α_y represent the focal length in terms of pixels on the axis x and y , and γ is the skew of image.

$$\mathbf{A} = \begin{bmatrix} \alpha_x & \gamma & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

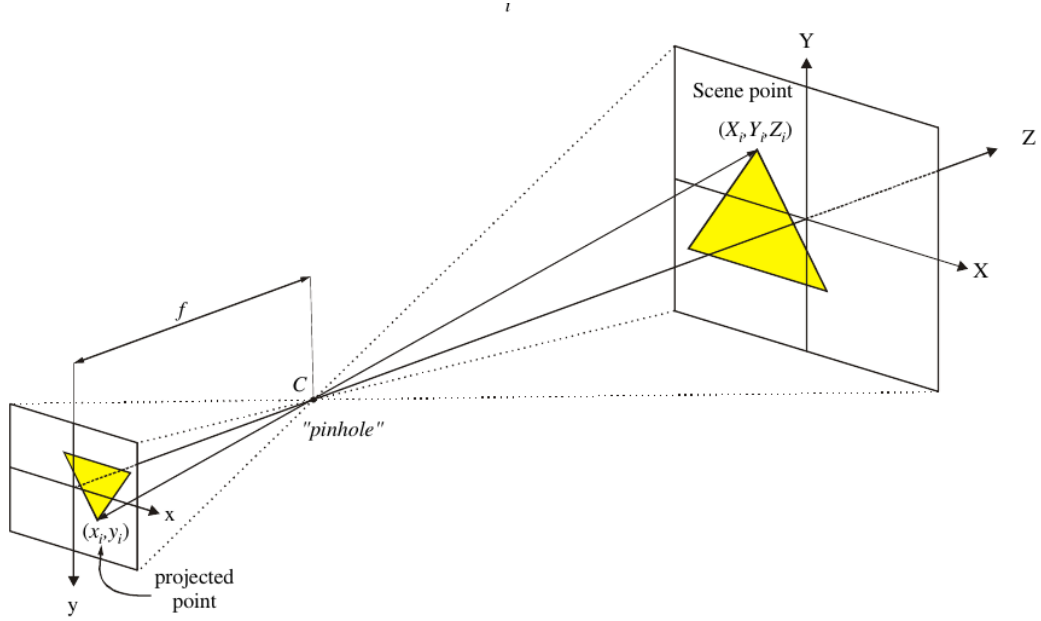


Figure 2.2: The pinhole imaging model [7].

The camera extrinsic parameters are given by the rotation matrix \mathbf{R} and translation vector \mathbf{t} , which are used to project an image on the world frame to camera frame. There is also a scale transformation, but it's already given by α_x and α_y .

Current cameras are equipped with lenses that produce some distortions on the images, however, the pinhole model is still a good approximation for our case since we are using a **PiCamera** which has minimal distortions.

The camera calibration has been done with using **OpenCV**. This library implementation is based on the technique described by [16] and the matlab implementation done by [8]. The calibration technique in [16] requires the camera to observe a planar pattern, usually a chessboard pattern, at different orientations, the more the better the estimation of the intrinsic parameters. The calibration algorithm minimizes the reprojection error which is the distance between observed feature points on the planar pattern and the projected using the estimated parameters.

For calibration we used a *Charuco* board instead of the classical chessboard because it generates a better estimation of the parameters [2].

The procedure to calibrate the PiCamera is straightforward with **OpenCV** and the sample codes found under `opencv_contrib-3.4.1/modules/aruco/samples`. **A detailed explanation can be found in the Appendix (not sure but is a remainder).**

1. Create a charuco board, print it and paste it on a solid and planar surface.
2. Compile the example code `calibrate_camera_charuco.cpp` and run it
3. Place your pattern in different orientations and take pictures
4. When you are done, just close the program

ADD PICTURE OF THE PATTERN ON MY OFFICE

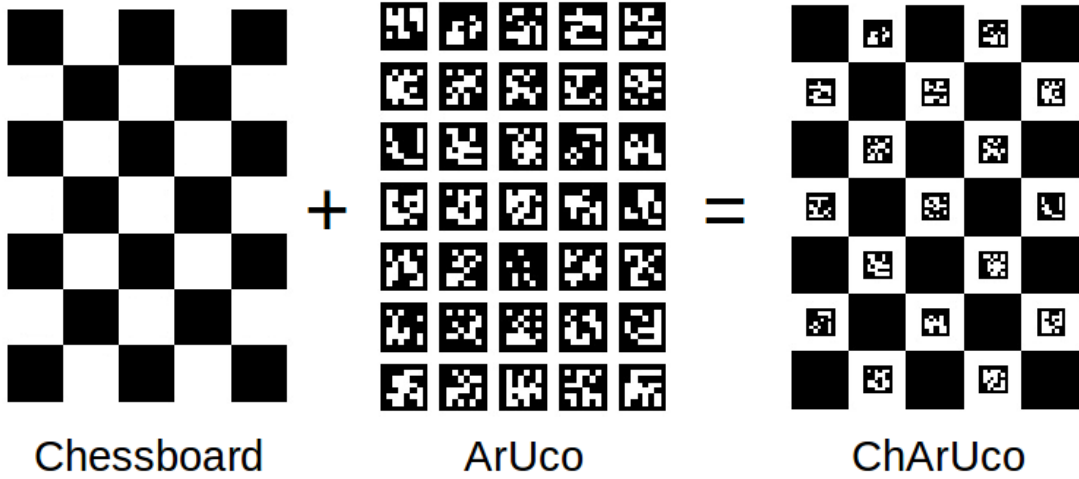


Figure 2.3: Plannar Patterns [2]

In our case, the camera intrinsic matrix \mathbf{A} is as following:

$$\mathbf{A} = \begin{bmatrix} \alpha_x & \gamma & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 6.125e+02 & 0. & 3.216e+02 \\ 0 & 6.122e+02 & 2.365e+02 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

2.4 Extrinsic Parameters

As it was mention before, the camera extrinsic parameters are given by the rotation matrix \mathbf{R} and translation vector \mathbf{t} . A rotation matrix can be formed as the product of three rotations around three cardinal axes, e.g., x , y , and z , or x , y , and x . This is generally a bad idea, as the result depends on the order in which the transforms applies [13].

However, a rotation can be also represented by a rotation axis $\mathbf{k} = [k_x, k_y, k_z]^T$ and an angle θ , or equivalently by a vector $\omega = \theta\mathbf{k}$. In order to do the transformation from axis-angle representation to rotation matrix, the cross-product matrix \mathbf{K} and Rodrigues' rotation formula can be used.

$$\mathbf{K} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \quad (2.4)$$

$$\mathbf{R} = \mathbf{I} + (\sin \theta)\mathbf{K} + (1 - \cos \theta)\mathbf{K}^2 \quad (2.5)$$

2.5 Rotation matrix to Euler angles

In order to get the angles related a rotation whose yaw, pitch and roll angles are ϕ , ρ and ψ . These angles are rotations in z , y and x axis respectively. We will rotate first about the x -axis, then the y -axis, and finally the z -axis. Such a sequence of rotations can be

represented as the matrix product

$$\mathbf{R} = R_z(\phi) R_y(\rho) R_x(\psi) \quad (2.6)$$

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi \\ 0 & \sin \psi & \cos \psi \end{bmatrix} \quad (2.7)$$

$$R_y(\rho) = \begin{bmatrix} \cos \rho & 0 & \sin \rho \\ 0 & 1 & 0 \\ -\sin \rho & 0 & \cos \rho \end{bmatrix} \quad (2.8)$$

$$R_z(\phi) = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

Given the given sequence of rotations and the algorithm described by [6], the angles can be found using algorithm 1.

```

if  $R_{31} \neq \pm 1$  then
     $\phi = \arctan 2(R_{21}, R_{11})$ 
     $\rho = -\arcsin(R_{31})$ 
     $\psi = \arctan 2(R_{32}, R_{33})$ 
else
     $\phi = 0$ 
     $\rho = -R_{31}\pi/2$ 
     $\psi = \arctan 2(-R_{23}, R_{22})$ 
end
Algorithm 1: Slabaugh's algo-
rithm

```

2.6 Use case implementation details

As it was mentioned before, the follower should be completely autonomous. In order to do so, the follower will need to read data from sensors, process that data and generate movement based on the processed data.

In figure 2.1 is shown a diagram of our use case. The main sensor is a PiCamera, the processing part performs the marker detection on the captured frames and the calculation of the euler angles and distance based on the equations described in the last section, and finally the movement is generated using the driving rover services.

Video Processing with OpenCV

Just like in the case of camera calibration, we use **OpenCV** and the submodule **aruco** to capture video and process the frames in order to extract information from the visual markers. To estimate and detect the marker at the beginning we should load the camera intrinsic parameters, saved in a YAML file, and the Aruco dictionary, composed by 256

marker and a marker size of 6x6 bits [2], to memory.

```
cv::FileStorage fs("calibration.yml", cv::FileStorage::READ);
cv::Ptr<cv::aruco::Dictionary> dictionary =
    cv::aruco::getPredefinedDictionary(cv::aruco::DICT_6X6_250);
```

Given a video frame, it is possible to detect Aruco markers if they are visible. When the marker is detected, we extract the four corners of the marker. The first corner is the top left corner, followed by the top right, bottom right and bottom left. The next step is to estimate the extrinsic camera parameters, which means the rotation vector ω and the translation vector \mathbf{t} . The size of the marker is an input parameter of the OpenCV function `cv::aruco::estimatePoseSingleMarkers`. In our case the marker size is 7cm.

```
cv::aruco::detectMarkers(image, dictionary,
    corners, ids);
cv::aruco::estimatePoseSingleMarkers(
    corners, 0.07,
    cameraMatrix, distCoeffs,
    rvec, tvec);

cv::Rodrigues(rvec.row(i), rmat);
rotationMatrixToEulerAngles(rmat, angle)
```

Relevant information

It's only important to check only rotation in Y-Axis because the rover-vehicle is just a car.

GY-521: Gyroscope Reading

The CY-521 sensor is an accelerometer and gyroscope, with that information we could estimate the angular position of follower. However, since the follower has none magnetometer the information given by the accelerometer is useless, since we have no information about the *magnetic north*.

uaehuaaoetnshutns etnahutnaeho uoeantsuhaoetnhun. etauhnsaetoh

Chapter 3

Results and Discussion

3.1 Results

None so far

Chapter 4

A Markdown-sablon használata

Empty section

4.1 Ábrák és táblázatok

Citation of a chapter Implementation

A table

Table 4.1: table title.

Órajel	Frekvencia	Cél pin
CLKA	100 MHz	FPGA CLK0
CLKB	48 MHz	FPGA CLK1
CLKC	20 MHz	Processzor
CLKD	25 MHz	Ethernet chip
CLKE	72 MHz	FPGA CLK2
XBUF	20 MHz	FPGA CLK3

4.2 Felsorolások és listák

Számozatlan felsorolásra mutat példát a jelenlegi bekezdés:

- *első bajusz*: ide lehetne írni az első elem kifejtését,
- *második bajusz*: ide lehetne írni a második elem kifejtését,
- *ez meg egy szakáll*: ide lehetne írni a harmadik elem kifejtését.

Számozott felsorolást is készíthetünk az alábbi módon:

1. *első bajusz*: ide lehetne írni az első elem kifejtését, és ez a kifejtés így néz ki, ha több sorosra sikeredik,
2. *második bajusz*: ide lehetne írni a második elem kifejtését,
3. *ez meg egy szakáll*: ide lehetne írni a harmadik elem kifejtését.

A felsorolásokban sorok végén vessző, az utolsó sor végén pedig pont a szokásos írásjel. Ez alól kivételt képezhet, ha az egyes elemek több teljes mondatot tartalmaznak.

Listákban a dolgozat szövegétől elkülönítendő kódrészleteket, programsorokat, pszeudokódokat jeleníthetünk meg.

```
* *első bajusz:* ide lehetne írni az első elem kifejtését, és ez a kifejtés így néz ki,
* *második bajusz:* ide lehetne írni a második elem kifejtését,
* *ez meg egy szakáll:* ide lehetne írni a harmadik elem kifejtését.
```

A listákban definiálható a használt programozási nyelv (pl. `java`, `latex`, `bash` stb.).

4.3 Képletek

Ha egy formula nem túlságosan hosszú, és nem akarjuk hivatkozni a szövegből, mint például a $e^{i\pi} + 1 = 0$ képlet, *szövegközi képletként* szokás leírni. Csak, hogy másik példát is lássunk, az $U_i = -d\Phi/dt$ Faraday-törvény a $\text{rot} E = -\frac{dB}{dt}$ differenciális alakban adott Maxwell-egyenlet felületre vett integráljából vezethető le. Látható, hogy a L^AT_EX-fordító a sorközöket betartja, így a szöveg szedése esztétikus marad szövegközi képletek használata esetén is.

Képletek esetén az általános konvenció, hogy a kisbetűk skalárt, a kis félkövér betűk (**v**) oszlopvektort – és ennek megfelelően **v**^T sorvektort – a kapitális félkövér betűk (**V**) mátrixot jelölnek. Ha ettől el szeretnénk térni, akkor az alkalmazni kívánt jelölésmódot célszerű külön alfejezetben definiálni. Ennek megfelelően, amennyiben **y** jelöli a mérések vektorát, **ϑ** a paraméterek vektorát és $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\vartheta}$ a paraméterekben lineáris modellt, akkor a *Least-Squares} értelemben optimális paraméterbecslő $\hat{\boldsymbol{\vartheta}}_{LS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ lesz.

Emellett kiemelt, sorszámozott képleteket is megadhatunk, ennél az `equation` és a `eqnarray` környezetek helyett a korszerűbb `align` környezet alkalmazását javasoljuk (több okból, különféle problémák elkerülése végett, amelyekre most nem térünk ki). Tehát

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad (4.1)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}, \quad (4.2)$$

ahol **x** az állapotvektor, **y** a mérések vektora és **A**, **B** és **C** a rendszert leíró paramétermátrixok. Figyeljük meg, hogy a két egyenletben az egyenlőségjelek egymáshoz igazítva jelennek meg, mivel a mindkettőt az & karakter előzi meg a kódban. Lehetőség van számozatlan kiemelt képlet használatára is, például

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u},$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}.$$

Mátrixok felírására az $\mathbf{Ax} = \mathbf{b}$ inhomogén lineáris egyenlet részletes kifejtésével mutatunk

példát:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \quad (4.3)$$

A `\frac` utasítás hatékonyságát egy általános másodfokú tag átviteli függvényén keresztül mutatjuk be, azaz

$$W(s) = \frac{A}{1 + 2T\xi s + s^2T^2}. \quad (4.4)$$

A matematikai mód minden szimbólumának és képességének a bemutatására természetesen itt nincs lehetőség, de gyors referenciaként hatékonyan használhatók a következő linkek:

- http://www.artofproblemsolving.com/LaTeX/AoPS_L_GuideSym.php,
- <http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf>,
- <ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf>.

Ez pedig itt egy magyarázat, hogy miért érdemes `align` környezetet használni: <http://texblog.net/latex-archive/math/eqnarray-align-environment/>.

4.4 Irodalmi hivatkozások

Az irodalmi hivatkozások alkalmazására javasolt a BiBTeX használata, ezért ez a sablon is ezt támogatja. Ebben az esetben egy külön szöveges adatbázisban definiáljuk a forrásmunkákat, és egy külön stílusfájl határozza meg az irodalomjegyzék kinézetét. Ez, összhangban azzal, hogy külön formátumkonvenció határozza meg a folyóirat-, a könyv-, a konferenciacikk stb. hivatkozások kinézetét az irodalomjegyzékben (a sablon használata esetén ezzel nem is kell foglalkoznia a hallgatónak, de az eredményt célszerű ellenőrizni). A felhasznált hivatkozások adatbázisa egy `.bib` kiterjesztésű szöveges fájl, amelynek szerkezetét az alábbi kódrészlet demonstrálja. A forrásmunkák bevitelkor a sor végi vesszők külön figyelmet igényelnek, mert hiányuk a BiBTeX-fordító hibaüzenetét eredményezi. A forrásmunkákat típus szerinti kulcsszó vezeti be (`@book` könyv, `@inproceedings` konferenciakiadványban megjelent cikk, `@article` folyóiratban megjelent cikk, `@techreport` valamilyen egyetem gondozásában megjelent műszaki tanulmány, `@manual` műszaki dokumentáció esetén stb.). Nemcsak a megjelenés stílusa, de a kötelezően megadandó mezők is típusról-típusra változnak. Egy jól használható referencia a <http://en.wikipedia.org/wiki/BibTeX> oldalon található.

```
@BOOK{Wettl104,  
  author = "Ferenc Wettl and Gyula Mayer and Péter Szabó",  
  title = "\LaTeX~kézikönyv",  
  publisher = "Panem Könyvkiadó",
```

```

    year = 2004
}
@ARTICLE{Candy86,
    author = "James C. Candy",
    title = "Decimation for Sigma Delta Modulation",
    journal = "{IEEE} Trans.\ on Communications",
    volume = 34,
    number = 1,
    pages = "72--76",
    month = jan,
    year = 1986,
}
@INPROCEEDINGS{Lee87,
    author = "Wai L. Lee and Charles G. Sodini",
    title = "A Topology for Higher Order Interpolative Coders",
    booktitle = "Proc.\ of the IEEE International Symposium on Circuits and Systems",
    year = 1987,
    vol = 2,
    month = may # "~4--7",
    address = "Philadelphia, PA, USA",
    pages = "459--462"
}
@PHDTHESIS{KissPhD,
    author = "Peter Kiss",
    title = "Adaptive Digital Compensation of Analog Circuit Imperfections for Cascaded D",
    school = "Technical University of Timi\c{s}oara, Romania",
    month = apr,
    year = 2000
}
@MANUAL{Schreier00,
    author = "Richard Schreier",
    title = "The Delta-Sigma Toolbox v5.2",
    organization = "Oregon State University",
    year = 2000,
    month = jan,
    note = "\newline URL: http://www.mathworks.com/matlabcentral/fileexchange/"
}
@MISC{DipPortal,
    author = "Budapesti {M}űszaki és {G}azdaságtudományi {E}gyetem, {V}illamosmérnöki és",
    title = "{D}iplomaterv portál (2011 február 26.)",
    howpublished = "\url{http://diplomaterv.vik.bme.hu/}",
}

```

A stílusfájl egy `.sty` kiterjesztésű fájl, de ezzel lényegében nem kell foglalkozni, mert vannak beépített stílusok, amelyek jól használhatók. Ez a sablon a BiBTeX-et használja, a hozzá tartozó adatbázisfájl a `mybib.bib` fájl. Megfigyelhető, hogy az irodalomjegyzéket a dokumentum végére (a `\end{document}` utasítás elé) beillesztett `\bibliography{mybib}` utasítással hozhatjuk létre, a stílusát pedig ugyanitt a `\bibliographystyle{plain}` utasítással adhatjuk meg. Ebben az esetben a `plain` előre definiált stílust használjuk (a sablonban is ezt állítottuk be). A `plain` stíluson kívül természetesen számtalan más előre definiált stílus is létezik. Mivel a `.bib` adatbázisban ezeket megadtuk, a BiBTeX-fordító is meg tudja különböztetni a szerzőt a címtől és a kiadótól, és ez alapján automatikusan generálódik az irodalomjegyzék a stílusfájl által meghatározott stílusban.

Az egyes forrásmunkákra a szövegből továbbra is a `@[...]` paranccsal tudunk hivatkozni, így a fenti kódrészlet esetén a hivatkozások rendre `[@Wett104]`, `[@Candy86]`, `[@Lee87]`, `[@KissPhD]`, `[@Schreirer00]` és `[@DipPortal]`. Az irodalomjegyzékben alapértelmezésben csak azok a forrásmunkák jelennek meg, amelyekre található hivatkozás a szövegben, és ez így alapvetően helyes is, hiszen olyan forrásmunkákat nem illik az irodalomjegyzékbe írni, amelyekre nincs hivatkozás.

Mivel a fordítási folyamat során több lépésben oldódnak fel a szimbólumok, ezért gyakran többször (TeXLive és TeXnicCenter esetén 2-3-szor) is le kell fordítani a dokumentumot. Ilyenkor ez első 1-2 fordítás esetleg szimbólum-feloldásra vonatkozó figyelmeztető üzenettel zárul. Ha hibaüzenettel zárul bármelyik fordítás, akkor nincs értelme megismételni, hanem a hibát kell megkeresni. A `.bib` fájl megváltoztatáskor sokszor nincs hatása a változtatásnak azonnal, mivel nem mindig fut újra a BibTeX fordító. Ezért célszerű a változtatás után azt manuálisan is lefuttatni (TeXnicCenter esetén `Build/BibTeX`).

Hogy a szövegbe ágyazott hivatkozások kinézetét demonstráljuk, itt most sorban meghivatkozunk a `[14]`, `[3]`, `[9]`, `[?]` és az `[12]` forrásmunkát, valamint az `[?]` weboldalt.

Megjegyzendő, hogy az ékezetes magyar betűket is tartalmazó `.bib` fájl az `inputenc` csomaggal betöltött `latin2` betűkészlet miatt fordítható. Ugyanez a `.bib` fájl hibaüzenettel fordul egy olyan dokumentumban, ami nem tartalmazza a `\usepackage[latin2]{inputenc}` sort. Speciális igény esetén az irodalmi adatbázis általánosabb érvényűvé tehető, ha az ékezetes betűket speciális latex karakterekkel helyettesítjük a `.bib` fájlban, pl. `á` helyett `\'a`-t vagy `ő` helyett `\H{o}`-t írunk.

4.5 A dolgozat szerkezete és a forrásfájlok

A diplomatervsablon (a kari irányelvek szerint) az alábbi fő fejezetekből áll:

- *tájékoztató* a szakdolgozat/diplomaterv szerkezetéről, ami a végső dolgozathoz tartozó, valamint a *feladatkiírás* (`guideline.md`), a dolgozat nyomtatott verziójában ennek a helyére kerül a tanszék által kiadott, a tanszékvezető által aláírt feladatkiírás, a dolgozat elektronikus verziójába pedig a feladatkiírás egyáltalán nem kerüljön bele, azt külön tölti fel a tanszék a diplomaterv-honlapra,
- *címoldal* (generált),
- *tartalomjegyzék* (generált),

- a diplomatervező *nyilatkozata* az önálló munkáról (generált),
- 1-2 oldalas tartalmi *összefoglaló* magyarul és angolul, illetve elkészíthető még további nyelveken is (`abstract.md`),
- *fejezetek: bevezetés* (feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása, `chapter1.md`), a feladatkiírás pontosítása és részletes elemzése, előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések, a tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása, a megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek (`chapter{2..n}.md`),
- összefoglalás (`summary.md`),
- esetleges *köszönetnyilvánítások* (`acknowledgement.md`),
- részletes és pontos *irodalomjegyzék* (generált),
- *függelékek* (`appendices.md`).

Chapter 5

Summary

A summary of the diploma

Acknowledgements

A köszönetnyilvánítás nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

List of Tables

4.1 table title. 14

List of Figures

1.1	Eclipse Kuksa Ecosystem [10]	5
1.2	Rover Components [1]	6
2.1	Rover Use Case	8
2.2	The pinhole imaging model [7].	9
2.3	Plannar Patterns [2]	10

Bibliography

- [1] Eclipse APP4MC. Eclipse app4mc - rover documentation, 2017.
URL: <https://app4mc-rover.github.io/rover-docs/>.
- [2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [3] James C. Candy. Decimation for sigma delta modulation. *IEEE Trans. on Communications*, 34(1):72–76, January 1986.
- [4] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.
- [5] The Linux Foundation. Automotive grade linux, 2016.
URL: <https://www.automotivelinux.org/>.
- [6] Gregory G. Slabaugh. Computing euler angles from a rotation matrix, 01 1999.
Technical Report URL: www.gregslabaugh.net/publications/euler.pdf.
- [7] Rafael Garcia. *A proposal to estimate the motion of an underwater vehicle through visual mosaicking*. PhD thesis, University of Girona. Doctor of Philosophy Girona, 2001.
- [8] J.Y.Bouguet. Matlab calibration tool (2018-11-20).
URL:<http://www.vision.caltech.edu/bouguetj/calibdoc/>.
- [9] Wai L. Lee and Charles G. Sodini. A topology for higher order interpolative coders. In *Proc. of the IEEE International Symposium on Circuits and Systems*, pages 459–462, Philadelphia, PA, USA, May 4–7 1987.
- [10] APPSTACLE Project. Eclipse kuksa, 2016.
URL: <https://www.eclipse.org/kuksa/>.
- [11] APPSTACLE Project. open standard application platform for cars and transportation vehicles, 2016.
URL: <https://itea3.org/project/appstacle.html>.
- [12] Richard Schreier. The delta-sigma toolbox v5.2, January 2000.
URL: <http://www.mathworks.com/matlabcentral/fileexchange/>.
- [13] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

- [14] Ferenc Wettl, Gyula Mayer, and Péter Szabó. *L^AT_EX kézikönyv*. Panem Könyvkiadó, 2004.
- [15] Z. Zhang. Emerging topics in computer vision, chapter 2: Camera calibration. Upper Saddle River, NJ, USA, 2004. Prentice Hall PTR.
- [16] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, November 2000.

Appendix A

Appendix

A.1 Calibration of the picam in raspbian

The `OpenCV` version we are using is 3.4.1.

bla bla bla... more details here

A.2 Answer to the “Life, the Universe, and All” questions

A Pitagorasz-tételből levezetve

$$c^2 = a^2 + b^2 = 42.$$

A Faraday-indukciós törvényből levezetve

$$\text{rot} E = -\frac{dB}{dt} \quad \longrightarrow \quad U_i = \oint_{\mathbf{L}} \mathbf{E} d\mathbf{l} = -\frac{d}{dt} \int_A \mathbf{B} d\mathbf{a} = 42.$$