DHBW
Duale Hochschule
Baden-Württemberg

Autonome Navigation (im Indoor-Bereich) der selbstbalancierenden Plattform RobStep

Studienarbeit

für die Prüfung zum

Bachelor of Engineering

des Studiengangs Informationstechnik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Philip Hug & Simon Simon

Mai 2016

Bearbeitungszeitraum	5tes & 6tes Semester
Matrikelnummer, Kurs	4815162342, TINF13B3
Ausbildungsfirma	Firma GmbH, Firmenort
Betreuer	Prof. Dr. Markus Strand
Gutachter	-

Sperrvermerk

Die vorliegende Studienarbeit mit dem Titel *Autonome Navigation (im Indoor-Bereich) der selbstbalancierenden Plattform RobStep* enthält unternehmensinterne bzw. vertrauliche Informationen der Firma GmbH, ist deshalb mit einem Sperrvermerk versehen und wird ausschließlich zu Prüfungszwecken am Studiengang Informationstechnik der Dualen Hochschule Baden-Württemberg Karlsruhe vorgelegt. Sie ist ausschließlich zur Einsicht durch den zugeteilten Gutachter, die Leitung des Studiengangs und ggf. den Prüfungsausschuss des Studiengangs bestimmt. Es ist untersagt,

- den Inhalt dieser Arbeit (einschließlich Daten, Abbildungen, Tabellen, Zeichnungen usw.) als Ganzes oder auszugsweise weiterzugeben,
- Kopien oder Abschriften dieser Arbeit (einschließlich Daten, Abbildungen, Tabellen, Zeichnungen usw.) als Ganzes oder in Auszügen anzufertigen,
- diese Arbeit zu veröffentlichen bzw. digital, elektronisch oder virtuell zur Verfügung zu stellen.

Jede anderweitige Einsichtnahme und Veröffentlichung – auch von Teilen der Arbeit – bedarf der vorherigen Zustimmung durch den Verfasser und Firma GmbH.

Karlsruhe, Mai 2016

Philip Hug & Simon Simon

Erklärung

Ich erkläre hiermit ehrenwörtlich:

1. dass ich meine Studienarbeit mit dem Thema *Autonome Navigation (im Indoor-Bereich) der selbstbalancierenden Plattform RobStep* ohne fremde Hilfe angefertigt habe;
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;
3. dass ich meine Studienarbeit bei keiner anderen Prüfung vorgelegt habe;
4. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Karlsruhe, Mai 2016

Philip Hug & Simon Simon

Abstract

Abstract normalerweise auf Englisch. Siehe: http://www.dhbw.de/fileadmin/user/public/Dokumente/Portal/Richtlinien_Praxismodule_Studien_und_Bachelorarbeiten_JG2011ff.pdf (8.3.1 Inhaltsverzeichnis)

Ein „Abstract“ ist eine prägnante Inhaltsangabe, ein Abriss ohne Interpretation und Wertung einer wissenschaftlichen Arbeit. In DIN 1426 wird das (oder auch der) Abstract als Kurzreferat zur Inhaltsangabe beschrieben.

Objektivität soll sich jeder persönlichen Wertung enthalten

Kürze soll so kurz wie möglich sein

Genauigkeit soll genau die Inhalte und die Meinung der Originalarbeit wiedergeben

Üblicherweise müssen wissenschaftliche Artikel einen Abstract enthalten, typischerweise von 100-150 Wörtern, ohne Bilder und Literaturzitate und in einem Absatz.

Quelle: http://de.wikipedia.org/wiki/Abstract Abgerufen 07.07.2011

Diese etwa einseitige Zusammenfassung soll es dem Leser ermöglichen, Inhalt der Arbeit und Vorgehensweise des Autors rasch zu überblicken. Gegenstand des Abstract sind insbesondere

- Problemstellung der Arbeit,
- im Rahmen der Arbeit geprüfte Hypothesen bzw. beantwortete Fragen,
- der Analyse zugrunde liegende Methode,
- wesentliche, im Rahmen der Arbeit gewonnene Erkenntnisse,
- Einschränkungen des Gültigkeitsbereichs (der Erkenntnisse) sowie nicht beantwortete Fragen.

Quelle: http://www.ib.dhbw-mannheim.de/fileadmin/ms/bwl-ib/Downloads_alt/Leitfaden_31.05.pdf, S. 49

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Theorieinhalte

1.1 Kinect

Bei der Microsoft Kinect handelt es sich um einen kombinierten Bildsensor für den Consumermarkt. Der von Microsoft ursprünglich konzipierte Einsatzzweck ist für die Integration von Gestensteuerung in Videospielen in Kombination mit einer Microsoft Xbox 360 Konsole. Erste Details des Gerätes wurden zunächst noch unter dem Workingtitle Project Natal veröffentlicht. Die Prototypen entstanden in Zusammenarbeit mit PrimeSense. Der sehr günstige Preis, mittlerweile 20 Euro für ein gebrauchtes Gerät, sowie quelloffene Treiber und Libraries machen die Kinect zu einem sehr beliebten Baustein zahlreicher Hobby- und Forschungsprojekten.

1.1.1 Fotosensor

Die Kinect verfügt über eine Kombination von Farb- und Tiefenkamera. Beim Farbsensor handelt es sich um einen herkömmlichen VGA-Sensor welcher Farbwerte im RGB-24Bit Farbraum liefert. Bilder werden zunächst mit einer Auflösung von $1280 * 960$ Pixel aufgenommen, anschließend allerdings auf $640 * 480$ Pixel per downsampling herunterreduziert. Das Aufnahmefeld erstreckt sich über 57 Grad Horizontal und 43 Grad Vertikal. [**kinect-georg**]

1.1.2 Tiefenbild-System

Tiefenbilder werden durch eine Kombination von IR-Laser Emitter und Sensor erzeugt. Der Emitter bestrahlt die Umgebung in einem Winkel von 58Grad Horizontal und 45Grad Vertikal mit einem pseudorandom IR-Muster. Diese Technik funktioniert in Bereichen mit viel natürlich Licht nur mit schweren Einschränkungen. Natürliches Licht enthält ebenfalls Infrarote Strahlung, welche zu Interferenzen mit dem IR-Muster des Emitters führt. Der Sensor kann somit das erzeugte Mesh nicht mehr eindeutig identifizieren und erzeugt somit ein extrem chaotisches Bild, welches sich durch starkes Rauschen äußert. Der Laser des Emitters arbeitet mit einer Leistung von 70mW und ist somit nit eyesafe. PrimeSense

hat eine patentierte Methode entwickelt welche den Einsatz ohne Schutzbrille ermöglicht. Laserlicht wird in der Regel mit einer Diode erzeugt. Dioden erzeugen punktgerichtete Strahlen. Dadurch ist bei der Betrachtung des Meshes im Bildzentrum ein Punkt mit höherer Intensität als die restlichen Lichtpunkte zu erkennen. Durch die scattering genannte Methode wird dieser zentrale Punkt auf 9 unterschiedliche Punkte durch Streuung verteilt. Dadurch wird auch die Intensität der einzelnen Scatter-Points auf $\frac{1}{9}$ reduziert, was es für das menschliche Auge unbedenklich macht. Der Emitter erzeugt zunächst ein Mesh mit einer Auflösung von $1200 * 960$ Pixeln. Im Nachhinein wird die Auflösung allerdings durch downsampling auf $640 * 490$ Pixel reduziert. Dies ist nötig da der USB-Stack das begrenzende Medium darstellt, und ansonsten die zeitgerechte Übertragung des Farb sowie des Tiefenbildes nicht sichergestellt werden kann. Microsoft gibt einen Funktionsbereich von 0,8m bis 3,5m an, welcher sich allerdings in praktischen Anwendungen auf 0,5 bis 3m eingependelt hat. Die Monochrom Kamera arbeitet mit einer nativen Auflösung von $1280 * 1024$ Pixel, welche allerdings bereits vor dem downsampling verkleinert wurde. Die Tiefenbilder enthalten Tiefenwerte im Wertebereich von 13Bit. [**kinect-hacking**]

1.1.3 Funktionsweise

light coding

Das Tiefensystem der Kinect unterstützt keine Laufzeitmessung. Stattdessen wird *structured light* verwendet. Dabei wird ein vorher definiertes Muster durch einen Emitter auf eine Fläche projiziert. Das Muster ist aus Punkten zusammengesetzt. Innerhalb des Musters gibt es keine Wiederholungen, was dazu führt dass jeder Punkt innerhalb des Bildes eindeutig identifziert werden kann. Bei der Kinect kommt entgegen des herkömmlichen Vorgehens zwei versetzte Bildsensoren zu verwenden nur ein einzelner zum Zug. Das Bild des fehlenden Sensors wird durch ein hart-codiertes Bild substituiert. Dieses Bild wird als Referenzobjekt für den Wertevergleich mit dem vom Sensor empfangenen Muster genutzt. Dabei werden per stereo triangulationsverfahren die räumlichen Unterschiede der in beiden bildern bekannten Punkte berechnet. Hierbei dienen die Intensitätsunterschiede der Punkte als Berechnungsgrundlage. Ist der vom Sensor empfangene Punkt intensiver als der selbe Punkt im Referenzbild, ist der Punkt näher am Sensor. Bei schwächerer Intensität liegt der reale Punkt tiefer im Raum als der Referenzpunkt.
[**kinect-georg**] [**alpha-centauri-ueberlicht**] [**kinect-uug-chem**]

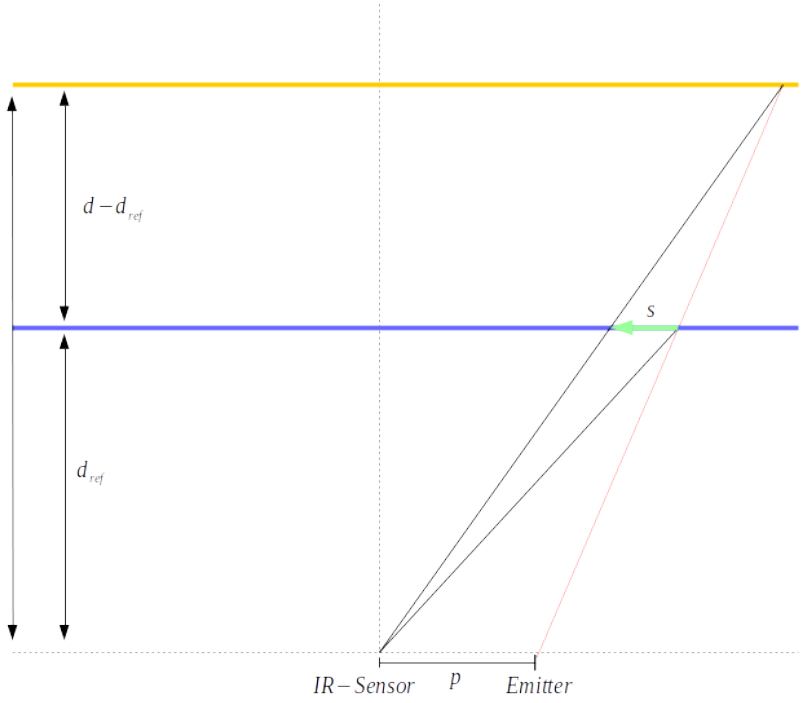

Abbildung 1.1: Illustration des light coding Prizips des Kinect Tiefenbildsystems.

Die Werte für d_ref (Tiefe des Referenzbildes) und p (horizontale Distanz zwischen Emitter und Monochromer Kamera) sind bekannt. Die Strecke s (horizontale Verschiebung zwischen Referenz- und Emitterbild) kann durch die eindeutig identifizierbaren Punkte errechnet werden. Die Tiefe d des Punktes im Emitterbild kann durch $d = \frac{p*d_ref}{s-p}$ berechnet werden. [**kinect-uug-chem**]

1.2 OpenCV

1.3 Robot Operating System

1.3.1 basics

1.3.2 Topics & Nodes

1.4 Robstep

1.5 Robotino

2 Problemstellung

2.1 Vision

2.2 Erwartetes Ziel

2.3 Aufgabenverteilung

2.4 Zeitplan

3 Basis Konzept

- Grundlegende Idee -> Intelligenz aus Roboter herausnehmen in die Umgebung integrieren
- Fokus liegt hier nun auf Kommunikation der beteiligten Komponenten
- Grundmechanik auf ROS aufsetzen -> Eventbus -> liefert Basis für Kommunikation und für Eventhandling -> Erweiterungsfreundlich -> sehr gute Doku -> breite Unterstützung
- Wie soll die Kommunikation von statten gehen?
- ROS-System erklären Netzwerkkommunikation, Eventbus, Nodes Topics
- Welche Rolle spielt die Kinect
- Wie sieht die geplante Umgebung aus? -> Zeichnung

3.1 Kommunikation

- ROS nutzt netzwerktechnologien
- Medium W-LAN -> Flexibel -> Alle Komponenten einfach um WLAN erweiterbar bzw. schon damit ausgestattet
- Kommunikation innerhalb von ROS
- ROS nutzt hierfür Messages in verschiedenen einfach zu definierenden formaten

3.2 Eventhandling

3.3 Wahrnehmen der Umgebung

3.4 Feature Konzepte

Die Konzepte für die Wegfindung, die Analyse von Hindernissen und die Kommunikation mit der Plattform Robotino befinden sich in den jeweiligen Kapiteln.

4 Vorarbeiten

4.1 Betriebssystem

4.2 Installation der ROS Umgebung

4.3 Installation der Packete zur Nutzung des Microsoft Kinect Sensors

4.3.1 Treiberpakete

4.3.2 Frameworks

4.4 Bilder von Kinect bekommen

4.4.1 Ansprechen innerhalb von ROS

Topics & Nodes

zwischenspeichern der frames

image_transport? image_pipeline?

4.5 OpenCV

4.5.1 Installation

- Version 2.4.11 -> letzte Stabile V2.4 Release
- Build from Source

- cMake
- Auswahl der enthaltenen Module für Makefile
- make danach sudo make install

4.6 Robotino

4.6.1 robotino api2

- robotino api2
- eintrag in sources list
- apt-get update / install robotino-api2

4.6.2 robotino_pkg

- catkin workspace bauen
- apt-get install ros-indigo-navigation
- packet in catkin_ws/src
- robotino_node/CMakelists.txt/include_directories -> /usr/local/robotino/api2
- catkin_ws -> catkin_make
- catkin_make install
- source /catkin_ws/devel/setup.bash

5 Wegfindung

6 Hindernisse

Die Interaktion mit seiner Umwelt ist ein essentieller Teil in der Anwendung autonomer Systeme. Besonders der Sicherheitsaspekt spielt hier eine primäre Rolle. Die Akzeptanz der potentiellen Nutzer wäre stark beeinträchtigt wenn die Unversehrtheit von umstehenden Lebewesen und Gegenständen nicht sichergestellt werden kann. Die Komplexität bei Umweltinteraktion zeigt sich besonders beim Auftreten dynamischer Events. Ein Beispiel hierfür ist das Erscheinen von Hindernissen im Bewegungsraum einer mobilen Plattform. Um auf solche Ereignisse angemessen zu reagieren müssen dynamische Objekte erkannt, verfolgt und identifiziert werden. Bei der Entkopptlung der Intelligenz vom Roboter selbst, eröffnet sich ein weiteres Problem. Die Unterscheidung zwischen einem dynamischen Objekt und der mobilen Plattform. Im folgenden Kapitel soll ein Konzept für ein solches Szenario erarbeitet und umgesetzt werden.

6.1 Definition von Hinderniss

Die Definition lautet wie folgt: Ëtwas, was das direkte Erreichen eines Ziels, das Weiterkommen be- oder verhindert.: [**duden-hinderniss**] Dabei wird offengelassen welche Ursache das Hinderniss sein kann, Gegenstände und Lebewesen werden hier nicht unterschieden, somit wird im weiteren Verlauf lediglich die neutrale Form Objekt genutzt. Genauer von Objekten welche den Bewegungsraum der Mobilen Plattform betreten bzw. sich bereits darin befinden. Zusätzlich müssen Objekte anhand ihres Verhaltens Klassifiziert werden. Dies ist notwendig da nicht auf jede Situation mit dem selben Vorgehen reagiert werden kann. Dadurch entsteht die Möglichkeit auf Basis der definierten Klassen eine Fallunterscheidung mit dafür angepassten Reaktionsroutinen zu entwerfen.

- dynamisches Objekt = Lebewesen oder Gegenstand welches den Bildbereich der Kamera betritt. Bewegt sich kontinuierlich auf einem festen Pfad durch den Bewegungsraum der Plattform.
- statisches Objekt = Lebewesen oder Gegenstand welches in den Bewegungsraum der Plattform betritt und dort an einem festen Punkt verweilt.

- chaotisches Objekt = Lebewesen oder Gegenstand welches ohne vorhersehbares Muster im Bewegungsraum der Plattform umherwandelt, und in unregelmäßigen Abständen für eine undefinierbare Zeit an einem fixen Punkt verweilt.
- Blitz Objekt = Lebewesen oder Gegenstand welches den Bewegungsraum der Mobilen Plattform nur sehr flach durchstreift i.e. den Bewegungsraum nicht tief betritt oder ihn lediglich tangiert, aber dennoch von der Raumüberwachung erfasst wird.

6.2 Objekte erkennen

Die Grundlage für jede Interaktion mit seiner Umgebung, unabhängig davon ob nun von einem Roboter oder einem Lebewesen gesprochen wird, bildet die Voraussetzung zu Erkennen und zu Verstehen was darin vorgeht. Bei Lebewesen wie Robotern geschieht dies über die Audio-Visuellen Schnittstellen des Körpers bzw. des Systems.

6.2.1 Konzeptionelle Lösungsansätze

Der Grundsätzliche Ansatz besteht darin die Umgebung mit einem Bildsensor zu überwachen und Veränderungen sichtbar zu machen und hervorzuheben. Dies geschieht durch den Abgleich von Inhalten über eine Serie von Bildern. Dieses Verfahren wird in einigen Publikationen als Frame-Analysis bezeichnet. Die Unterschiede der implementierten Vertreter dieses Verfahrens liegen in den beobachteten Merkmalen der Bilder. Im Verlauf des Projekts wurden drei Varianten verglichen. Für die Implementierung wurde OpenCV genutzt. Beweggründe für diese Entscheidung können dem Theoriekapitel über OpenCV entnommen werden.
(ADD) Verweiss auf OpenCV-ref (ADD)

6.2.2 Sequential Images

- Graustufenbild, monochrom
- vergleicht Bildf_0 mit Nachfolgerf_1
- dabei wird für jeden Pixel die absolute Differenz gebildet.
- Sind die Werte$w_i(f_0)$, $w_i(f_1)$ von Pixel$i$ in f_0 und f_1 identisch wird der Wert für diesen Pixel auf 0 gesetzt. Dabei handelt es sich um einen optimalen Punkt. In der reellen Anwendung dieses Verfahrens werden die Werte nur stark reduziert da durch die Dynamik der Lichtverhältnisse die Grauwerte des Pixels zwar nur marginal dafür

allerdings konstant schwanken und dadurch die Wahrscheinlichkeit für das Eintreffen des Idealfalls $w_i(f_0) = w_i(f_1)$ nur sehr gering ist.

- Durch dieses Verfahren werden dem Informationsgehalt die Gemeinsamkeiten beider Bilder entzogen, und nur die Veränderungen von f_0 zu f_1 verbleiben.
- dadurch bekommt man ein Differenzbild welches die geänderten Pixel isoliert hervorhebt, was das erkennen dynamischer Objekte prinzipiell stark vereinfacht.

6.2.3 find Contour

- RGB Bild
- initialer Prozess = welcher Pixel hat welche Farbe
- verfolgt die Farbänderung jedes Pixels mit Fokus auf Farbwert
- sequential images legt Fokus auf Änderung im Allgemeinen = Änderung Grauwert bedeutet Änderung im Raum
- Colour detection = "Wandern"von Farbwerten
- benachbarter Bildpunkt nimmt Farbe an(Ausbreiten), $C(P_i) = C(P_i + 1)$.
- Farbwert wechselt von Pixel P_i zu $P_i + 1$, dabei nimmt P_i einen anderen Farbwert an(Wandern). $C(P_i + 1(f_1)) == C(P_i(f_0))$
- detektiert Außenkante Konturen engl. contour von Objekten dabei Ausbreiten = Forderseite des Objekts. Wandern = Rückseite des Objekts

(ADD)4er & 8er Nachbarschaft | Illustration der Merkmale(ADD)

6.2.4 Direkte Nutzung von Tiefenbildern

6.2.5 Implementierung

- OpenCV = absdiff(absolute differential); braucht graustufe; benötigt 2 Bilder als eingabe liefert differenzielles Bild entweder in Bild1 zurück oder in übergebenen Ausgabespeicher
- cv-bridge
- ros eigene sensor-msgs images Nachrichten in OpenCV format bringen
- wo liegt der Unterschied?

- Code erklären -> evtl Diagramm
- in CMakelists.txt -> add_executable + add_library
- anpassungen bezüglich graustufenbilder -> in image_converter.cpp

(ADD)relevante Codeschnippsel in listing und erklären; Programmablaufplan zeichnen(ADD)

6.3 Objekte Identifizieren

- lediglich Unterscheidung ob Roboter oder Hindernis
- alles was nicht Roboter ist => Hindernis
- findobjekt ros

6.3.1 Unterscheiden zwischen Roboter und Objekt

- Robotino identifizieren -> Alleinstellungsmerkmal
- spezielles Muster? QR-Code? Abstraktes deutliches schwarz/weiß Design
- Skelett des Robotino anfertigen?
- Wie Perspektivische Unterschiede ausgleichen?

7 Reaktion auf Hindernisse

7.1 Grundsätzliche Verhaltensweise bei auftauchenden Hindernissen

- Hinderniss taucht im Bewegungsraum auf
  - Stop
  - Bewegungsanalyse
  - Hindernis kommt direkt auf Roboter zu -> weiterhin stehen bleiben
  - Hindernis stoppt und bewegt sich nicht mehr weiter -> Umgehung bestimmen, umfahren
  - Hindernis stoppt und bewegt sich weiter -> Roboter bleibt solange stehen bis Hindernis den Bewegungsraum verlassen hat oder sich nicht mehr weiter im Raum bewegt(stillstand)

7.2 Kategorisieren von identifizierten Objekten

7.2.1 fortwährend bewegende Objekte

7.2.2 stillstehende Objekte

8 Interaktion mit mobiler Plattform

8.1 Plattformwechsel

- Robstep
- warum weg von Robstep
- warum eignet sich der Robotino besser?

8.2 Kommunikation mit mobiler Plattform

8.2.1 Integration des Robotino in ROS

8.2.2 Analyse der Topics und Nodes

- robotino_node
- robotino_local_movement
- rosrun robotino_local_move robotino_local_move_client_node x, y, Rotation in Grad, Timeout in Sekunden
- Befehl zum Unterbrechen von Befehlen und sofortiges Anhalten.

8.2.3 Konzeption der Kommunikation

nötige Funktionen

mögliche Befehle & erwartetes Verhalten

Visualisierung des Kommunikationsprozesses

8.3 Abbilden der Steuerbefehle auf Anweisungen des Eventhandlers

9 Prototyp

9.1 Testdurchführung

9.2 Evaluation

10 Fazit

Anhang

(Beispielhafter Anhang)

A. Assignment

B. List of CD Contents

C. CD

B. List of CD Contents

- **Literature/**
  - **Citavi-Project(incl pdfs)/** ⇒ *Citavi (bibliography software) project with almost all found sources relating to this report. The PDFs linked to bibliography items therein are in the sub-directory 'CitaviFiles'*
    - bibliography.bib ⇒ *Exported Bibliography file with all sources*
    - Studienarbeit.ctv4 ⇒ *Citavi Project file*
    - **CitaviCovers/** ⇒ *Images of bibliography cover pages*
    - **CitaviFiles/** ⇒ *Cited and most other found PDF resources*
  - **eBooks/**
  - **JournalArticles/**
  - **Standards/**
  - **Websites/**
- **Presentation/**
  - presentation.pptx
  - presentation.pdf
- **Report/**
  - Aufgabenstellung.pdf
  - Studienarbeit2.pdf
  - **Latex-Files/** ⇒ *editable LaTeX files and other included files for this report*
    - **ads/** ⇒ *Front- and Backmatter*
    - **content/** ⇒ *Main part*
    - **images/** ⇒ *All used images*
    - **lang/** ⇒ *Language files for LaTeX template*