TECHNISCHE UNIVERSITÄT DRESDEN

Ein Anwenderleitfaden für das Erstellen einer wissenschaftlichen Abhandlung

Falk Hanisch*

29. März 2017

Der Versuch, ein allumfassendes Tutorial für eine wissenschaftliche Arbeit zur Verfügung zu stellen gleicht der beschwerlichen Suche nach einer eierlegenden Wollmilchsau. Es ist quasi nicht möglich, alle potenziellen Anforderungen an eine wissenschaftliche Arbeit in einem Dokument abzudecken, insbesondere weil es für diese fast von jedem Lehrstuhl mehr oder weniger unterschiedliche Vorgaben zu Formatierung und Layout gibt. Dennoch soll hier versucht werden, einen Großteil der für gewöhnlich auftretenden Erfordernisse zu bearbeiten.

Dieses Tutorial hat *nicht* die Intention, LaTeX-Einsteigern sämtliche Grundlagen zu erläutern. Vielmehr wird davon ausgegangen, dass Sie bereits erste Erfahrungen mit LaTeX 2ε gesammelt haben. Dennoch wird versucht, alle Schritte möglichst leicht nachvollziehbar zu gestalten. Sollte Ihnen beim Lesen und Durcharbeiten des Tutorials etwas auf- oder missfallen, so dürfen Sie mich gerne per E-Mail kontaktieren. Auch Anregungen und Wünsche dürfen sie mir gegenüber gerne kommunizieren.

Für absolute Neueinsteiger gibt es einige freie Tutorials, welche die ersten Schritte mit LaTeX 2ε stark erleichtern. Nicola L. C. Talbot stellt sehr gute Tutorials für LaTeX-Novizen[1] [Tal12] sowie Dissertationen[2] [Tal13] zur freien Verfügung.

In erster Linie ist dieser Leitfaden für Anwender gedacht, die für ihre wissenschaftliche Arbeit eine TUD-Script-Dokumentklasse verwenden wollen. Das vorgestellte Vorgehen kann jedoch – natürlich mit gewissen Abstrichen – auch mit anderen Klassen, insbesondere denen aus dem KOMA-Script-Bundle, umgesetzt werden. Viele der hier im Folgenden verwendeten Optionen und Befehle aus dem TUD-Script-Bundle werden lediglich sporadisch in ihrer Grundfunktion erläutert. Allerdings lassen sich detaillierte Erläuterung über die türkisfarbigen Hyperlinks zum TUD-Script-Handbuch jederzeit sehr einfach öffnen. Des Weiteren wird auf eine Vielzahl

*hanisch.latex@outlook.com

[1] http://www.dickimaw-books.com/latex/novices/

[2] http://www.dickimaw-books.com/latex/thesis/

von Paketen verwiesen, deren Dokumentation über die magentafarbigen Links im Comprehensive TeX Archive Network (CTAN) geöffnet werden kann. Alternativ kann dies auch über das Terminal respektive die Kommandozeile mit dem Aufruf `texdoc <Paket>` direkt lokal erfolgen. Ein Großteil dieser Pakete wird zusätzlich kurz im TUD-Script-Handbuch beschrieben.

Der Anwenderleitfaden muss nicht zwingend vollständig nachvollzogen werden. Dieser ist in einzelne Abschnitte untergliedert, damit Sie sich bestimmte Aspekte erarbeiten können. Sollten Querbezüge zu den einzelnen Abschnitten bestehen, werden diese auch genannt. Zu guter Letzt findet sich am Ende dieses Dokumentes das komplette Tutorial als ausführbarer Quelltext.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Zu Beginn werden einige Pakete geladen, die quasi in jedem neu erstellten Dokument verwendet werden sollten. Alle weiteren in diesem Tutorial genutzten und/oder empfohlenen Pakete werden in den relevanten Abschnitten des Tutorials genauer erläutert.

Beim Laden der Klasse mit `\documentclass` können Sie globale Optionen angeben. Diese werden sowohl an die Klasse selbst weitergereicht als auch allen in der Präambel geladenen Pakete zur Verfügung gestellt. Falls das Paket **babel** für die Sprachauswahl im Dokument verwendet wird, sollten auf alle Fälle die verwendeten Sprachen als Klassenoption angegeben werden, wobei die zuletzt angegebene als aktuelle Sprache aktiviert wird. Dadurch werden diese nicht nur an das Paket **babel** sondern auch an andere Pakete übergeben, welche sprachspezifische Einstellungen vornehmen.

```
\documentclass[english,ngerman]{tudscrreprt}
\usepackage{babel}
```

Mit dem Paket **fontenc** lässt sich die Kodierung der für die Ausgabe der verwendeten Schriften im Dokument spezifizieren, wobei die europäische Zeichenkodierung mit der Option `T1` aktiviert wird. Kommt mit **LuaLaTeX** oder **XeLaTeX** ein Unicode-Textsatzsystem zum Einsatz, sollte besser das Paket **fontspec** geladen werden.

Weiterhin kann die Eingabekodierung der kompilierten Datei eingestellt werden. Diese ist standardmäßig `utf8`. Bei der Verwendung von **pdfLaTeX** sind auch andere Kodierungen möglich.[3]

Wurde ein Sprachpaket für die deutsche Sprache geladen (**babel** oder **polyglossia**), so werden für **LuaLaTeX** und **XeLaTeX** erweiterte Trennmuster verwendet. Für **pdfLaTeX** lässt sich eine wesentlich verbesserte Worttrennung für die deutsche Sprache mit dem Paket **hyphsubst** aktivieren. In Unterkapitel 10 können Sie weitere Informationen zur Verwendung von **hyphsubst** sowie zum Thema *Worttrennung* finden.

[3] Sollte die Datei in einer anderen Kodierung gespeichert sein, so kann das Paket **selinput** genutzt werden, um die Kodierung automatisch zu erkennen. Alternativ lässt sich die Eingabekodierung auch mit dem Paket **inputenc** auch manuell festgelegen. In beiden Fällen sollte das Paket **fontenc** *vorher* geladen worden sein.

```
\ifpdftex{
  \usepackage[T1]{fontenc}
  \usepackage[ngerman=ngerman-x-latest]{hyphsubst}
}{
  \usepackage{fontspec}
}
```

Für die KOMA-Script-Klassen wird außerdem das Paket **scrhack** bereitgestellt. Dieses behebt einige Probleme bei der Kompatibilität mit anderen Paketen bezüglich des Erstellens von Verzeichnissen. Deshalb sollte dieses bei Bedarf frühzeitig in der Präambel geladen werden.

```
\usepackage{scrhack}
```

Damit sind die allgemein notwendigen Pakete eingebunden. Es werden zwar weitere benötigt, diese werden allerdings in den einzelnen Abschnitten dieses Tutorials aufgeführt.

2 Makrotypografie

2.1 Satzspiegel

Gleich zu Beginn und bevor das eigentliche Verfassen der Arbeit beginnt, sollte sich der Verfasser Gedanken über das zu nutzenden Layout und den Satzspiegel machen, um bei der Finalisierung des Dokumentes keine böse Überraschung bei Seitenumbrüchen oder der Position von Gleitobjekten zu erleben.

Zuerst gilt es zu entscheiden, ob das Dokument einseitig oder beidseitig gesetzt werden soll. Ist Letzteres der Fall, so sollte die KOMA-Script-Option `twoside` als Klassenoption angegeben werden. Im nächsten Schritt ist der zu verwendende Satzspiegel festzulegen. Hierfür kann die TUD-Script-Option `cdgeometry` verwendet werden, welche im TUD-Script-Handbuch beschrieben ist. Normalerweise wird das Dokument im asymmetrischen Layout des Corporate Designs gesetzt. Dieses Verhalten wird mit `cdgeometry=false` deaktiviert und der Satzspiegel durch das Paket **typearea** nach typografischen Gesichtspunkten konstruiert. Dabei ist für dessen Aufteilung maßgeblich die KOMA-Script-Option `DIV` verantwortlich, mit der KOMA-Script-Option `BCOR` wird die Bindekorrektur angegeben. Diese wird in Abschnitt 2.3 erläutert.

2.2 Änderung des Zeilenabstandes (Durchschuss)

Auf eine Änderung des Zeilenabstandes sollte aus typografischer Sicht bei der Verwendung einer Standardschrift im Normalfall verzichtet werden. Die Gründe hierfür werden im Handbuch von TUD-Script bei der Vorstellung des Paketes **setspace** erklärt, welches für dieses Anliegen den Befehl `\setstretch{`*`<Faktor>`*`}` bereitstellt. Wird die **Open Sans**-Hausschrift der Technischen Universität Dresden verwendet, sollte der Faktor im Bereich `1.05`–`1.1` liegen.

2.3 Bindekorrektur

Falls die Arbeit nach der Fertigstellung gebunden werden soll, so ist auf den notwendigen Binderand zu achten, quasi der Teil einer Seite, welcher durch die Bindung „verschwindet“ und nicht mehr als sichtbarer Teil der Seite vorhanden ist. Als Faustregel gilt, dass die erforderliche

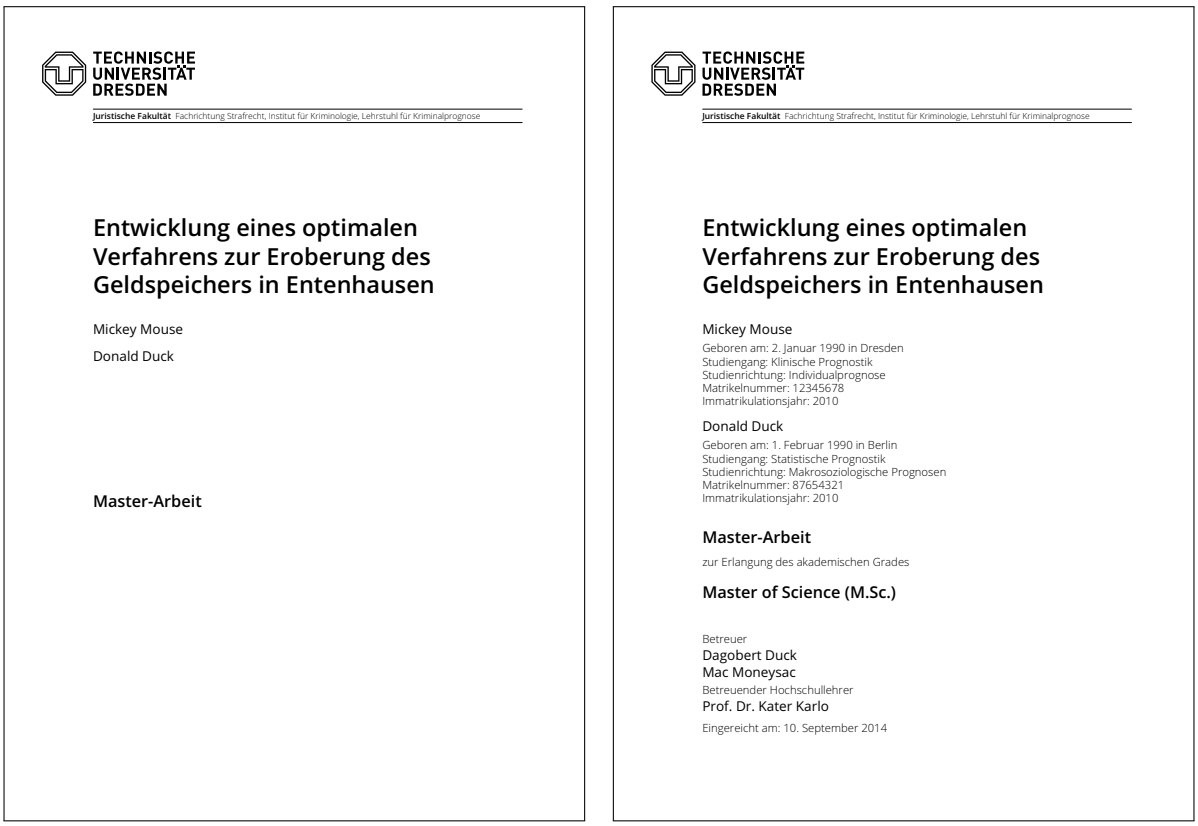


Abbildung 1: Umschlagseite und Titel

Bindekorrektur in etwa der halben Höhe des Buchblocks entsprechen sollte. Dessen Höhe wiederum ist abhängig von der Anzahl der Seiten sowie der Papierdichte.

Wird qualitativ höherwertiges Papier mit einer Dichte von $100\,\mathrm{g/m^2}$ verwendet, so entsprechen 100 Blatt einer Buchblockhöhe von zirka 12 mm. Dementsprechend wäre in diesem Beispiel eine Bindekorrektur von 6 mm notwendig, die sich mit der Klassenoption `BCOR=6mm` einstellen ließe. Sollte die erwartete Seitenzahl davon abweichen, kann die einzustellende Bindekorrektur linear skaliert werden. Für differierende Papierdichten sollte die Buchblockhöhe für die geschätzte Gesamtseitenanzahl in Erfahrung gebracht werden.

3 Umschlagseite und Titel

Die Umschlagseite und der Titel sind sich in ihrer Gestalt sehr ähnlich. Allerdings gibt es ein paar kleine Unterschiede. Einerseits werden auf dem Cover weniger Informationen als auf der Titelseite ausgegeben. Andererseits wird der Titel immer im Satzspiegel des restlichen Dokumentes ausgegeben, wohingegen die Umschlagseite – ohne weitere Optionen – im asymmetrischen Layout des Corporate Designs der Technischen Universität Dresden erscheint. Wie dieses Verhalten geändert werden kann, ist im Handbuch für `\makecover` erläutert. Die resultierende Ausgabe des nachfolgenden Quelltextauszugs ist in Abbildung 1 zu sehen und stellt eine mögliche Ausprägung von Umschlagseite und Titel dar.

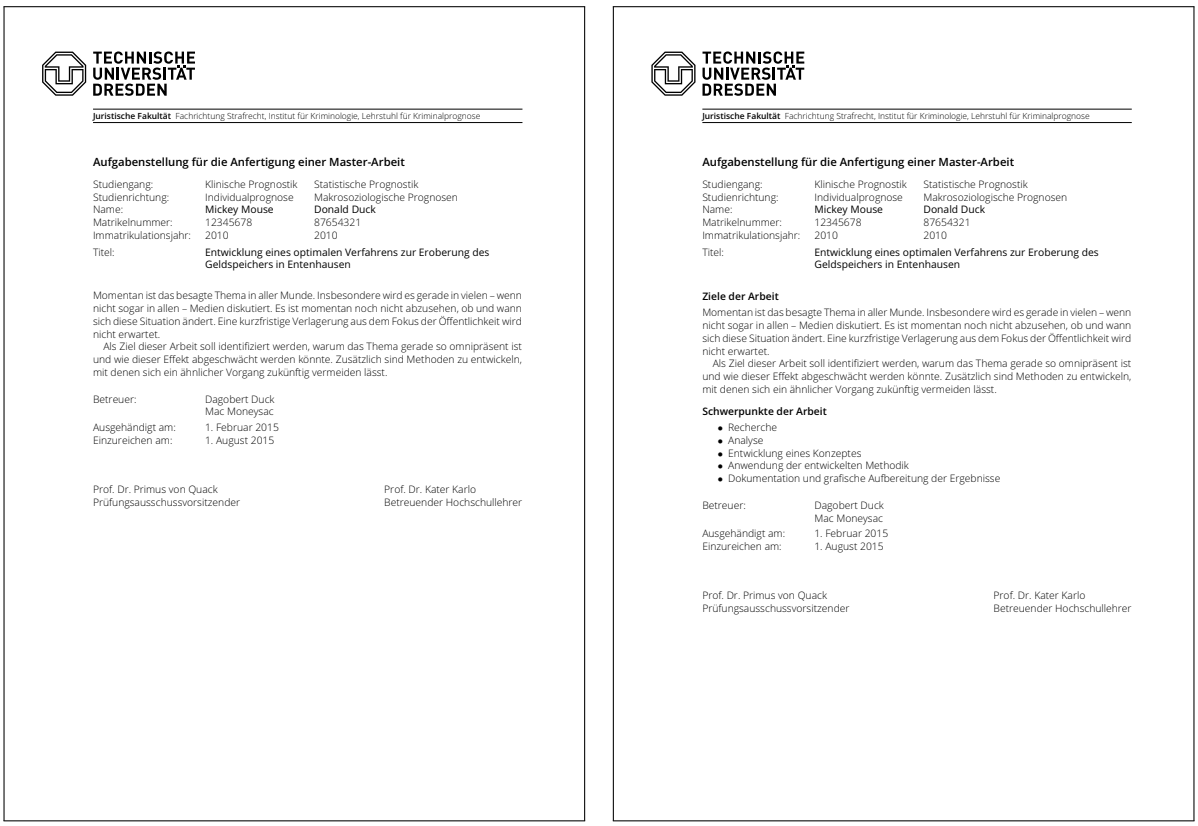
```
\faculty{Juristische Fakultät}
\department{Fachrichtung Strafrecht}
\institute{Institut für Kriminologie}
\chair{Lehrstuhl für Kriminalprognose}
\title{%
  Entwicklung eines optimalen Verfahrens zur Eroberung des
  Geldspeichers in Entenhausen
}
\thesis{master}
\graduation[M.Sc.]{Master of Science}
\author{%
  Mickey Mouse%
  \matriculationnumber{12345678}%
  \dateofbirth{2.1.1990}%
  \placeofbirth{Dresden}%
  \course{Klinische Prognostik}%
  \discipline{Individualprognose}%
\and%
  Donald Duck%
  \matriculationnumber{87654321}%
  \dateofbirth{1.2.1990}%
  \placeofbirth{Berlin}%
  \course{Statistische Prognostik}%
  \discipline{Makrosoziologische Prognosen}%
}
\matriculationyear{2010}
\supervisor{Dagobert Duck \and Mac Moneysac}
\professor{Prof. Dr. Kater Karlo}
\date{10.09.2014}
\makecover
\maketitle
```

4 Vor- und Nachspann

In den folgenden Unterabschnitten werden Elemente vorgestellt, welche häufig als Bestandteil einer wissenschaftlichen (Abschluss-)Arbeit gefordert werden. Die Platzierung oder Position der vorgestellten Elemente innerhalb der Arbeit ist nicht eindeutig durch eine Norm oder dergleichen festgelegt. Vielmehr gibt es meist eine Richtlinie vom verantwortlichen Prüfungsamt oder eine konkrete Vorgabe des betreuenden wissenschaftlichen Mitarbeiters oder Hochschullehrers.

Verwenden Sie für ihre Arbeit eine Buchklasse, so können die entsprechenden Abschnitte mit `\frontmatter` (Vorspann), `\mainmatter` (Hauptteil) und `\backmatter` (Nachspann) gekennzeichnet werden. Einen Anhang starten Sie mit `\appendix`, womit unter anderem die Kapitelnummerierung auf Großbuchstaben umgeschaltet wird. Weitere Hinweise zu den genannten Befehlen sind im KOMA-Script-Handbuch[4] nachzulesen.

[4] http://mirrors.ctan.org/macros/latex/contrib/koma-script/doc/scrguide.pdf



TECHNISCHE UNIVERSITÄT DRESDEN

Juristische Fakultät Fachrichtung Strafrecht, Institut für Kriminologie, Lehrstuhl für Kriminalprognose

Aufgabenstellung für die Anfertigung einer Master-Arbeit

Studiengang:	Klinische Prognostik	Statistische Prognostik
Studienrichtung:	Individualprognose	Makrosoziologische Prognosen
Name:	**Mickey Mouse**	**Donald Duck**
Matrikelnummer:	12345678	87654321
Immatrikulationsjahr:	2010	2010
Titel:	**Entwicklung eines optimalen Verfahrens zur Eroberung des Geldspeichers in Entenhausen**	

Momentan ist das besagte Thema in aller Munde. Insbesondere wird es gerade in vielen – wenn nicht sogar in allen – Medien diskutiert. Es ist momentan noch nicht abzusehen, ob und wann sich diese Situation ändert. Eine kurzfristige Verlagerung aus dem Fokus der Öffentlichkeit wird nicht erwartet.

Als Ziel dieser Arbeit soll identifiziert werden, warum das Thema gerade so omnipräsent ist und wie dieser Effekt abgeschwächt werden könnte. Zusätzlich sind Methoden zu entwickeln, mit denen sich ein ähnlicher Vorgang zukünftig vermeiden lässt.

Betreuer:	Dagobert Duck Mac Moneysac
Ausgehändigt am:	1. Februar 2015
Einzureichen am:	1. August 2015

Prof. Dr. Primus von Quack
Prüfungsausschussvorsitzender

Prof. Dr. Kater Karlo
Betreuender Hochschullehrer

TECHNISCHE UNIVERSITÄT DRESDEN

Juristische Fakultät Fachrichtung Strafrecht, Institut für Kriminologie, Lehrstuhl für Kriminalprognose

Aufgabenstellung für die Anfertigung einer Master-Arbeit

Studiengang:	Klinische Prognostik	Statistische Prognostik
Studienrichtung:	Individualprognose	Makrosoziologische Prognosen
Name:	**Mickey Mouse**	**Donald Duck**
Matrikelnummer:	12345678	87654321
Immatrikulationsjahr:	2010	2010
Titel:	**Entwicklung eines optimalen Verfahrens zur Eroberung des Geldspeichers in Entenhausen**	

Ziele der Arbeit

Momentan ist das besagte Thema in aller Munde. Insbesondere wird es gerade in vielen – wenn nicht sogar in allen – Medien diskutiert. Es ist momentan noch nicht abzusehen, ob und wann sich diese Situation ändert. Eine kurzfristige Verlagerung aus dem Fokus der Öffentlichkeit wird nicht erwartet.

Als Ziel dieser Arbeit soll identifiziert werden, warum das Thema gerade so omnipräsent ist und wie dieser Effekt abgeschwächt werden könnte. Zusätzlich sind Methoden zu entwickeln, mit denen sich ein ähnlicher Vorgang zukünftig vermeiden lässt.

Schwerpunkte der Arbeit

- Recherche
- Analyse
- Entwicklung eines Konzeptes
- Anwendung der entwickelten Methodik
- Dokumentation und grafische Aufbereitung der Ergebnisse

Betreuer:	Dagobert Duck Mac Moneysac
Ausgehändigt am:	1. Februar 2015
Einzureichen am:	1. August 2015

Prof. Dr. Primus von Quack
Prüfungsausschussvorsitzender

Prof. Dr. Kater Karlo
Betreuender Hochschullehrer

Abbildung 2: Aufgabenstellung in freier und standardisierter Form

4.1 Aufgabenstellung

Das Erstellen der Aufgabenstellung einer Abschlussarbeit im Corporate Design der Technischen Universität Dresden muss das Paket **tudscrsupervisor** geladen werden.

```
\usepackage{tudscrsupervisor}
```

Dieses Paket stellt die Umgebung `task` sowie den Befehl `\taskform` bereit. Bei beiden Varianten wird zu Beginn eine Tabelle mit Autoreninformationen erzeugt. Am Ende werden der oder die Betreuer der Arbeit sowie Professor und gegebenenfalls der Prüfungsausschussvorsitzende ausgegeben. Mit der Umgebung `task` kann zwischen Kopf und Fuß ein beliebiger Inhalt gesetzt werden. Der Befehl `\taskform` hingegen erzeugt eine standardisierte Ausgabe, wobei der Inhalt des zweiten obligatorischen Arguments in einer `itemize`-Umgebung verwendet wird und somit `\item` zu nutzen ist. Das Resultat des folgenden Quelltextes ist in Abbildung 2 zu sehen.

```
\faculty{Juristische Fakultät}
\department{Fachrichtung Strafrecht}
\institute{Institut für Kriminologie}
\chair{Lehrstuhl für Kriminalprognose}
\title{%
  Entwicklung eines optimalen Verfahrens zur Eroberung des
  Geldspeichers in Entenhausen
}
```

```
\thesis{master}
\author{%
  Mickey Mouse%
  \matriculationnumber{12345678}%
  \course{Klinische Prognostik}%
  \discipline{Individualprognose}%
\and%
  Donald Duck%
  \matriculationnumber{87654321}%
  \course{Statistische Prognostik}%
  \discipline{Makrosoziologische Prognosen}%
}
\matriculationyear{2010}
\issuedate{1.2.2015}
\duedate{1.8.2015}
\supervisor{Dagobert Duck \and Mac Moneysac}
\chairman{Prof. Dr. Primus von Quack}
\professor{Prof. Dr. Kater Karlo}

\newcommand{\taskcontent}{%
  Momentan ist das besagte Thema in aller Munde. Insbesondere wird es
  gerade in vielen~--- wenn nicht sogar in allen~--- Medien diskutiert.
  Es ist momentan noch nicht abzusehen, ob und wann sich diese Situation
  ändert. Eine kurzfristige Verlagerung aus dem Fokus der Öffentlichkeit
  wird nicht erwartet.

  Als Ziel dieser Arbeit soll identifiziert werden, warum das Thema
  gerade so omnipräsent ist und wie dieser Effekt abgeschwächt werden
  könnte. Zusätzlich sind Methoden zu entwickeln, mit denen sich ein
  ähnlicher Vorgang zukünftig vermeiden lässt.
}

\begin{task}
\smallskip
\par\noindent
\taskcontent
\end{task}

\taskform[pagestyle=empty]{\taskcontent}{%
  \item Recherche
  \item Analyse
  \item Entwicklung eines Konzeptes
  \item Anwendung der entwickelten Methodik
  \item Dokumentation und grafische Aufbereitung der Ergebnisse
}
```

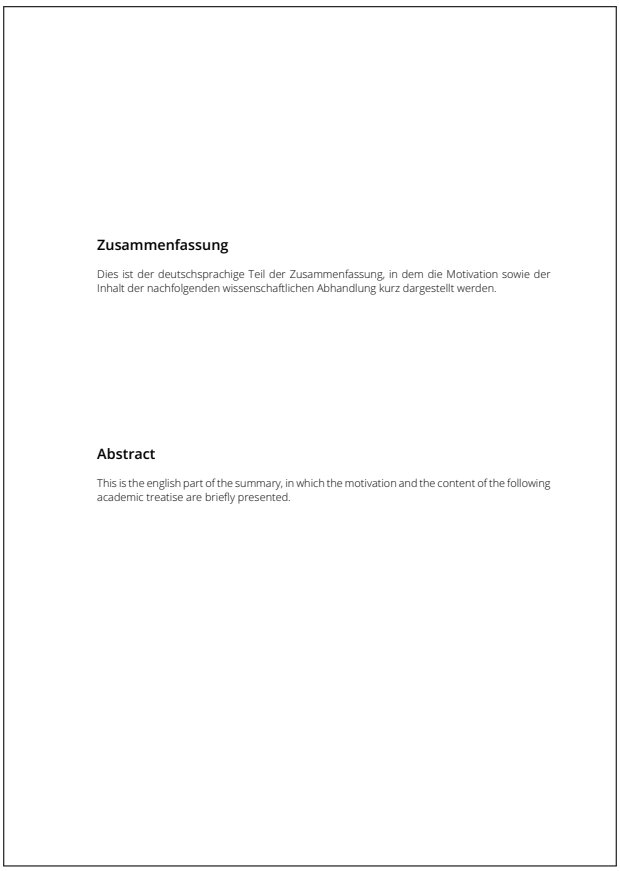
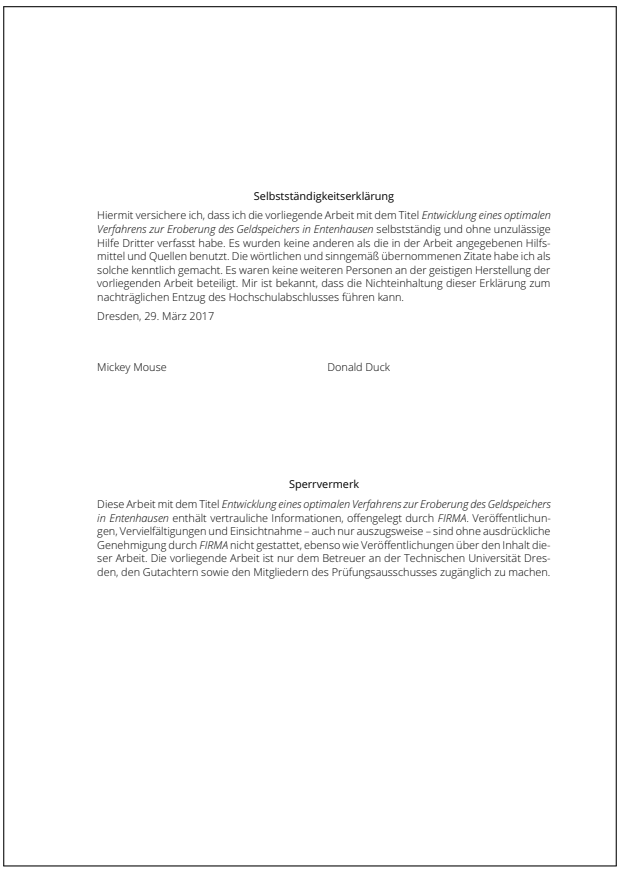


Abbildung 3: Zusammenfassung in deutscher und englischer Sprache

4.2 Zusammenfassung

Häufig wird zu Beginn einer wissenschaftliche Arbeit die Motivation und der Inhalt dieser zusammengefasst, um den Leser die Thematik der Abhandlung vorzustellen. in den meisten Fällen wird diese dabei in deutscher und englischer Sprache verfasst. Hierfür stellt KOMA-Script bereits die Umgebung `abstract` bereit. Vielfach wird der Wunsch geäußert, sowohl die deutsche als auch die englische Zusammenfassung auf derselben Seite zu setzen. Diese Variante kann mithilfe der TUD-Script-Klassen sehr einfach umgesetzt werden, wie der nachfolgende Quelltextauszug zeigt. Die resultierende Ausgabe ist in Abbildung 3 zu sehen.

```
\TUDoption{abstract}{multiple,section}
\begin{abstract}
  Dies ist der deutschsprachige Teil der Zusammenfassung, in dem die
  Motivation sowie der Inhalt der nachfolgenden wissenschaftlichen
  Abhandlung kurz dargestellt werden.
\nextabstract[english]
  This is the english part of the summary, in which the motivation and
  the content of the following academic treatise are briefly presented.
\end{abstract}
```



Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit mit dem Titel *Entwicklung eines optimalen Verfahrens zur Eroberung des Geldspeichers in Entenhausen* selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in der Arbeit angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Es waren keine weiteren Personen an der geistigen Herstellung der vorliegenden Arbeit beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Hochschulabschlusses führen kann.

Dresden, 29. März 2017

Mickey Mouse Donald Duck

Sperrvermerk

Diese Arbeit mit dem Titel *Entwicklung eines optimalen Verfahrens zur Eroberung des Geldspeichers in Entenhausen* enthält vertrauliche Informationen, offengelegt durch *FIRMA*. Veröffentlichungen, Vervielfältigungen und Einsichtnahme – auch nur auszugsweise – sind ohne ausdrückliche Genehmigung durch *FIRMA* nicht gestattet, ebenso wie Veröffentlichungen über den Inhalt dieser Arbeit. Die vorliegende Arbeit ist nur dem Betreuer an der Technischen Universität Dresden, den Gutachtern sowie den Mitgliedern des Prüfungsausschusses zugänglich zu machen.

Abbildung 4: Selbstständigkeitserklärung und Sperrvermerk

4.3 Selbstständigkeitserklärung und Sperrvermerk

Für die meisten Abschlussarbeiten an der Technischen Universität Dresden wird vom Verfasser eine Selbstständigkeitserklärung verlangt. Für diese wird ein Standardtext bereitgestellt. Dieser kann mit dem Befehl `\confirmation` ausgegeben werden. Wurde das Thema in Kooperation mit einem Unternehmen bearbeitet, so wird zumeist auch ein Sperrvermerk gefordert, welcher mit `\blocking` erzeugt werden kann. Mit `\declaration` lassen sich beide Erklärungen direkt nacheinander erzeugen. Die verwendete Überschrift und ein möglicher Eintrag in das Inhaltsverzeichnis können über die Option `declaration` reguliert werden. Eine mögliche Ausprägung der Erklärungen ist in Abbildung 4 abgebildet.

```
\title{%
  Entwicklung eines optimalen Verfahrens zur Eroberung des
  Geldspeichers in Entenhausen
}
\author{Mickey Mouse\and Donald Duck}
\declaration[company=FIRMA]
```

4.4 Inhalts-, Abbildungs-, und Tabellenverzeichnis

Das Inhaltsverzeichnis wird mit `\tableofcontents` erzeugt und führt die Gliederung des erstellten Dokumentes entsprechend der verwendeten Befehle[5] auf. Wurde das Paket **hyperref** geladen, so werden im Inhaltsverzeichnis PDF-Hyperlinks auf die einzelnen Abschnitte erzeugt.

Sowohl Abbildungen als auch Tabellen werden in LaTeX normalerweise mit speziellen Umgebungen – `figure` und `table` – eingebunden. Innerhalb dieser sogenannten Gleitumgebungen kann der Befehl `\caption[<Verzeichniseintrag>]{<Bezeichnung>}` genutzt werden, um diesen eine Bezeichnung hinzuzufügen. Mit `\listoffigures` beziehungsweise `\listoftables` lassen sich Verzeichnisse erstellen, in denen alle Gleitobjekte des jeweiligen Typs ausgegeben werden, falls diese denn eine Bezeichnung hinzugefügt wurde.

Sollen Abbildungen oder Tabellen außerhalb ihrer angestammten Gleitumgebung `figure` beziehungsweise `table` genutzt und benannt werden, kann dies entweder mit dem Befehl `\captionof{<Typ>}[<Verzeichniseintrag>]{<Bezeichnung>}` oder – falls die beiden Pakete **caption** und **hyperref** genutzt werden – auch wie gewohnt mit `\caption` allerdings zusätzlich ergänzt mit einem zuvor aufgerufenem `\captionsetup{type=figure/table}` erfolgen. Weitere Informationen diesbezüglich sind dem KOMA-Script-Handbuch[6] respektive der Anleitung zu **caption** zu entnehmen. In Unterkapitel 6 wird genauer auf Gleitumgebungen eingegangen.

```
\tableofcontents
\listoffigures
\listoftables
```

4.5 Abkürzungs- und Symbolverzeichnis

Für die Auszeichnung von Abkürzungen gibt es zwei sehr gute Pakete, die dieses Unterfangen stark vereinfachen. Die einfachere – jedoch nicht so mächtige – der beiden Varianten ist die Nutzung des Paketes **acro**. Sollen nur Abkürzungen und gegebenenfalls eine sortierte Liste dieser gesetzt werden, ist dieses allerdings absolut ausreichend. Für ein Symbolverzeichnis lässt sich in dieser Variante das Paket **nomencl** nutzen. Dieses bietet meiner Meinung nach jedoch keine großen Vorteile, stattdessen kann auch einfach eine Tabelle händisch erzeugt werden. Das Paket **acro** ist sehr gut und ausführlich dokumentiert. Deshalb wird hier auf eine exemplarische Erläuterung verzichtet und stattdessen auf dessen Dokumentationen verwiesen. Alternativ kann auch das Paket **acronym** verwendet werden, wobei die alphabetische Sortierung der Abkürzungen durch den Anwender erfolgen muss, weshalb ich von diesem eher abrate.

Die andere Möglichkeit ist die Nutzung des Paketes **glossaries**, das eine große Zahl an Einstellmöglichkeiten und Optionen besitzt, allerdings auch etwas Zeit für die Einarbeitung und Studium der Dokumentation benötigt. Der ursprüngliche Einsatzzweck dieses Paketes ist das Setzen eines fachsprachlichen oder technischen Glossars. Es bietet zusätzlich die Mittel zum Erzeugen eines Abkürzungs- sowie Symbolverzeichnis. Es soll folgend hier kurz erläutert werden, wie das Paket zu verwenden ist. Für weiterführende Beispiele sollte die Dokumentationen zu Rate gezogen werden, welche dahingehend keine Wünsche offen lässt.

[5] `\part`, `\addpart`, `\chapter`, `\addchap`, `\section`, `\addsec` etc.
[6] http://mirrors.ctan.org/macros/latex/contrib/koma-script/doc/scrguide.pdf

Achtung!

Das Paket **glossaries** sollte immer *nach* **hyperref** geladen werden. Entweder Sie achten explizit darauf oder Sie verwenden den Befehl `\AfterPackage*` aus dem KOMA-Script-Bundle. Mit diesem können Sie die Quelltext nach dem Laden eines Paketes ausführen. *Es ist allerdings darauf zu achten, dass der Quelltext nur ausgeführt wird, wenn das avisierte Zielpaket auch tatsächlich geladen wird.* Falls Sie demnach auf **hyperref** jedoch nicht auf **glossaries** verzichten möchten, sollte der nachfolgende Quelltext am Ende der Präambel eingefügt werden.

Die für **glossaries** verwendeten Optionen werden kurz erläutert.

```
\usepackage[%
```

Die zwei Paketoptionen `acronym` sowie `symbols` erzeugen die beiden Verzeichnisse für Abkürzungen und Symbole. Die Option `nomain` wird immer dann verwendet, wenn im Dokument kein zusätzliches allgemeines oder technisches Glossar erzeugt werden soll.

```
acronym,% Abkürzungen
symbols,% Formelzeichen
nomain,% kein Glossar
```

Durch `nogroupskip` wird der automatische Abstand zwischen den Einträgen zur Gruppierung innerhalb eines Glossars entfernt, die Option `toc` fügt alle erzeugten Verzeichnisse dem Inhaltsverzeichnis hinzu, `section` bestimmt die Gliederungsebene der Überschrift.

```
nogroupskip,%
toc,%
section=chapter,%
```

Mit der Option `nostyles` kann man gegebenenfalls das Laden der von **glossaries** vordefinierten Stilen verhindern. Dies ist insbesondere sinnvoll, wenn für die einzelnen Glossare und Verzeichnisse ohnehin eigene Stile erstellt werden, wie dies nachfolgend geschieht.

```
nostyles,%
```

Mit der Option `translate` werden die Überschriften aller Glossare in der Dokumentsprache gesetzt, wobei `translate=babel` die Nutzung des Paketes **babel** erzwingt.

```
translate=babel,%
```

Das Paket **glossaries** erstellt bei der Kompilierung des Dokumentes Hilfsdateien, anhand derer die Indexe erzeugt und sortiert werden können. Da **pdfLaTeX** diese Funktionalität nicht bereitstellt, ist hierfür der Aufruf einer passenden Anwendung notwendig. Die meisten Distributionen stellen dafür ***makeindex*** bereit. Dieses Programm kann prinzipiell zwar Einträge alphabetisch sortieren, bietet allerdings weder die Nutzung von Unicode-Symbolen noch eine Sprachunterstützung. Aufgrund der genannten Nachteile ist ***xindy*** zum Sortieren der Einträge zu empfehlen, da dieses Programm sowohl eine Unterstützung von Unicode als auch die Möglichkeit, nach sprachabhängigen Regeln zu sortieren, bietet. Allein für die deutsche Sprache gibt es beispielsweise zwei Varianten der Sortierung – nach DIN und nach Duden.

Ohne weiteres Zutun des Anwenders wird vom Paket **glossaries** in der Grundeinstellung ***makeindex*** genutzt. Soll stattdessen für die Sortierung der Glossar- und Verzeichniseinträge ***xindy*** zum Einsatz kommen, muss die Option `xindy` aktiviert werden. Dieser kann zusätzlich

sowohl die zu nutzende Spracheinstellung zur Sortierung als auch die gewünschte Eingabekodierung an ***xindy*** übergeben werden. Beispielsweise wird mit `xindy={language=german-din}` die deutsche Sortierung nach DIN aktiviert.

```
% mit Tex Live einfach verwendbar
  xindy={language=german-din},
```

Das Erstellen aller Glossare – unabhängig davon, ob ***xindy*** oder ***makeindex*** für die Sortierung zum Einsatz kommt – sollte das Perl-Skript **makeglossaries** verwendet werden, welches alle notwendigen Optionen an die jeweilige Anwendung weiterleitet.

Achtung!

Für den Aufruf von ***xindy*** *selbst als auch für die Ausführung des Skriptes* **makeglossaries** *ist ein Perl-Interpreter notwendig. Dieser wird lediglich von* **TEX Live** *nicht jedoch von* **MiKTEX** *direkt bereitgestellt. Wird die letztgenannte Distribution verwendet, muss der Anwender gegebenenfalls diesen zusätzlich installieren.*

Kann das Skript **makeglossaries** nicht genutzt werden, kann alternativ dazu die Option `automake` aktiviert werden. Diese sorgt dafür, dass am Ende des **pdfLATEX**-Laufs der Aufruf von ***xindy*** oder ***makeindex*** direkt erfolgt. Soll ***xindy*** verwendet werden, sind dafür allerdings erweiterte Schreibrechte notwendig, weshalb für diese Variante **pdfLATEX** mit der Option `--shell-escape` respektive `--enable-write18` aufgerufen werden muss. Für ***xindy*** ist außerdem die Angabe der Sprache über die Option `xindy={language=...}` zwingend notwendig.

Ist weder die Nutzung des Skriptes **makeglossaries** noch der – im Zweifel sicherheitskritische – Aufruf von **pdfLATEX** mit erweiterten Schreibrechten möglich, so muss die gewünschte Anwendung für die Sortierung explizit durch den Anwender aufgerufen werden, wobei auf die Angabe der richtigen Parameter zu achten ist. Genaueres hierzu sowie Lösungen für Probleme beim Erstellen der Glossare und Verzeichnisse sind in der Dokumentation von **glossaries** zu finden.

```
automake,%
```

Damit sind alle verwendeten Optionen erläutert. Schließlich sorgt der Befehl `\makeglossaries` für das Erstellen der optionsabhängigen Stildateien für ***makeindex*** respektive ***xindy*** sowie das Erzeugen der benötigten Hilfsdateien.

```
]{glossaries}
\makeglossaries
```

Damit wäre der erste Teil zur Initialisierung überstanden und wir können zum eigentlichen Problem kommen. Wie wird nun ein Abkürzungs- und/oder Symbolverzeichnis erstellt?

4.5.1 Abkürzungsverzeichnis

Das Paket **glossaries** stellt für die Definition von Abkürzungen einen speziellen Befehl bereit. Mit `\newacronym[<Parameterliste>]{<Label>}{<Abkürzung>}{<Wortgruppe>}` wird eine Abkürzung definiert und kann später über `{<Label>}` genutzt werden. Die möglichen optionalen Parameter können in der Dokumentation zu **glossaries** nachgeschlagen werden. Für ein kleines Beispiel werden drei Abkürzungen erstellt...

```
\newacronym{apsp}{APSP}{All-Pairs Shortest Path}
\newacronym{spsp}{SPSP}{Single-Pair Shortest Path}
\newacronym{sssp}{SSSP}{Single-Source Shortest Path}
```

... und diese in einer kurzen Textpassage mit dem **glossaries**-Befehl `\gls{<Label>}` verwendet.

```
In der Graphentheorie wird häufig die Lösung des Problems des kürzesten
Pfades zwischen zwei Knoten gesucht. Dieses Problem wird häufig auch
mit \gls{spsp} bezeichnet. Es lässt sich auf die Variationen \gls{sssp}
und \gls{apsp} erweitern. Für die Lösung von \gls{spsp}, \gls{sssp}
oder \gls{apsp} kommen unterschiedliche Algorithmen zum Einsatz.
```

In der Graphentheorie wird häufig die Lösung des Problems des kürzesten Pfades zwischen zwei Knoten gesucht. Dieses Problem wird häufig auch mit Single-Pair Shortest Path (SPSP) bezeichnet. Es lässt sich auf die Variationen Single-Source Shortest Path (SSSP) und All-Pairs Shortest Path (APSP) erweitern. Für die Lösung von SPSP, SSSP oder APSP kommen unterschiedliche Algorithmen zum Einsatz.

Gut zu sehen ist, dass sich die Ausgabe der Abkürzung bei der ersten Verwendung mit `\gls` von der zweiten – und jeder weiteren – unterscheidet. Das Verhalten lässt sich über verschiedene Stile mit `\setacronymstyle` anpassen. Die Ausgabe einer Liste aller Abkürzungen erfolgt mit:

```
\printacronyms
```

Akronyme

APSP All-Pairs Shortest Path. 15
SPSP Single-Pair Shortest Path. 15
SSSP Single-Source Shortest Path. 15

Dabei werden die Akronyme in einer `description`-Umgebung gesetzt, was absolut ausreichend ist. Mir persönlich ist allerdings die Darstellung in einer quasi-tabellarischen Form lieber. Dabei soll der Stil mit fettgedruckten Abkürzungen beibehalten werden. Das **glossaries**-Paket stellt zwar auch eine Vielzahl an Stilen in Tabellenform bereit, allerdings nicht in dem gewünschten. Deshalb wird nachfolgend gezeigt, wie sich ein eigener Stil in Tabellenform kreieren lässt.

Es bieten sich die Umgebungen `tabularx` oder `tabu` an, bei denen die Spaltenbreite teilweise automatisch berechnet wird, um sich manuelle Formatierungsarbeiten zu sparen. Folgend werden beide Varianten vorgestellt. Falls Sie noch keine Erfahrungen mit dem Tabellensatz in LaTeX 2_ε haben, lohnt sich vorher ein Blick in Unterkapitel 7, um die verwendeten Befehle und Umgebungen zu verstehen. Wie Sie Stile definieren, die Seitenumbrüche in einer Tabelle zulassen, können Sie in diesem Abschnitt bei der Symbolverzeichniserstellung erfahren.

Eigener Stil mit `tabularx`

Ein eigener Glossarstil kann mit `\newglossarystyle` definiert werden, wobei für den neu definierten Stil die Umgebung `theglossary` umdefiniert wird. Als erstes kommt die Tabellenumgebung `tabularx` aus dem gleichnamigen Paket, welches in Unterabschnitt 7.2.1 vorgestellt wird, zum Einsatz.

Es werden in der Tabelle drei Spalten definiert. Die erste und letzte Spalte sind schlicht linksbündige Standardspalten (`l`). In diesen werden die Abkürzungen selbst sowie die Seitenzahlen eingetragen. Die Verwendung von `@{}` führt dazu, dass der normalerweise vor der ersten und

nach der letzten Spalte eingefügte Abstand von `\tabcolsep` entfällt. Die Breite der Spalte vom Typ `X` wird automatisch berechnet.

Aufgrund der Implementierung von `tabularx` lässt sich diese nicht als verschachtelte Umgebung verwenden. Allerdings kann stattdessen die mit `\tabularx` sowie `\endtabularx` die Low-Level-Variante genutzt werden. Die Definition des neuen Stils `acrotabularx` wird nachfolgend ausgegeben, die weitergehende Erläuterung schließt sich daran an.

```
\newglossarystyle{acrotabularx}{%
  \renewenvironment{theglossary}{%
    \tabularx{\linewidth}{@{}lXl@{}}%
  }{%
    \endtabularx\par\bigskip%
  }%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand*{\glsgroupskip}{}%
  \renewcommand*{\glossentry}[2]{%
    \glsentryitem{##1}% Entry number if required
    \glstarget{##1}{\sffamily\bfseries\glossentryname{##1}} &
    \glsentrydesc{##1} &
    ##2\tabularnewline
  }
}
```

Der Rest des Stils ist schnell erläutert. Zunächst wird auf Tabellenköpfe (`\glossaryheader`) sowie Überschriften und Abstände bei Gruppierungen (`\glsgroupheading` sowie `\glsgroupskip`) verzichtet. Der Befehl `\glossentry` ist verantwortlich für die Formatierung der Einträge im Abkürzungsverzeichnis. Dieser wird intern durch **glossaries** mit zwei obligatorischen Argumenten aufgerufen. Das erste enthält das entsprechende Label, das zweite ein kommaseparierte Liste der Seitenzahlen. Dabei stehen verschiedene Makros zur Auswahl, um anhand des gegebenen Labels die gewünschten Informationen zu extrahieren.[7]

Der Befehl `\glossentry` wurde so definiert, dass für jeden Eintrag eine separate Zeile in der Tabelle erzeugt wird, wo in der ersten Spalte die Abkürzung selbst, in der zweiten die Langform und in der dritten Spalte schließlich die Liste der Seiten, auf welchen die jeweilige Abkürzung mit `\gls{<Label>}` verwendet wurde, ausgegeben wird. Zum Abschluss die resultierende Ausgabe des Abkürzungsverzeichnisses im neuen Stil.

```
\printacronyms[style=acrotabularx]
```

Akronyme

APSP	All-Pairs Shortest Path	15
SPSP	Single-Pair Shortest Path	15
SSSP	Single-Source Shortest Path	15

[7] bspw. mit `\glossentryname` die Bezeichnung oder mit `\glsentrydesc` die dazugehörige Beschreibung

Sollte der Platz für die Erläuterungen wie in diesem Beispiel in der mittleren Spalte mehr als ausreichend sein und kein Zeilenumbruch benötigt werden, kann auch einfach eine `tabular`-Umgebung mit einer `l`-Spalte anstelle von `X` verwendet werden.

Eigener Stil mit `tabu`

Die sehr komfortabel zu nutzende Umgebung `tabu` wird durch das Paket **tabu** bereitgestellt. Es wird später in Unterabschnitt 7.2.3 vorgestellt, wobei die dortigen Anmerkungen *unbedingt* zu beachten sind. Für die bereits verwendeten `tabularx`-Tabellen muss generell eine feste Tabellenbreite angegeben werden. Die Breite der `X`-Spalten wird anhand der angegebenen Gesamtbreite und dem für andere Spalten vom Typ `l`, `r` und `c` benötigten Platz berechnet.

Wie `tabularx` bietet auch die Umgebung `tabu` einen `X`-Spaltentyp. Für diese kann jedoch anstelle einer fest vorgegebenen Breite auch `spread 0pt` angegeben werden. Dadurch werden `X`-Spalten anfänglich in ihrer natürlichen Breite gesetzt. Sobald jedoch die Gesamtbreite der Tabelle den zur Verfügung stehenden Platz bis zum Zeilenende überschreiten würde, werden die `X`-Spalten automatisch umbrochen.

In umbrochenen Spalten gibt es beim Paket **tabu** jedoch ein kleineres Problem. In diesen setzt **tabu** zu wenig vertikalen Leerraum am unteren Ende. Um dieses Manko zu beheben, wird am Schluss jeder `X`-Spalte mit `X<{\strut}` einfach der Befehl `\strut` angehängt, der vertikalen Leerraum ober- und unterhalb der aktuellen Grundlinie einfügt. In Unterkapitel 7 wird ein Ansatz aufgezeigt, wie dies automatisiert über einen neuen Spaltenstil ausgemerzt werden kann. Der Rest des Stils ist identisch zu `acrotabularx`.

```
\newglossarystyle{acrotabu}{%
  \renewenvironment{theglossary}{%
    \begin{tabu}{@{}lX<{\strut}l@{}}% 'spread 0pt' defekt in v2.9
  }{%
    \end{tabu}\par\bigskip%
  }%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand*{\glsgroupskip}{}%
  \renewcommand*{\glossentry}[2]{%
    \glsentryitem{##1}% Entry number if required
    \glstarget{##1}{\sffamily\bfseries\glossentryname{##1}} &
    \glsentrydesc{##1} &
    ##2\tabularnewline
  }
}

\printacronyms[style=acrotabu]
```

Akronyme

APSP	All-Pairs Shortest Path	15
SPSP	Single-Pair Shortest Path	15
SSSP	Single-Source Shortest Path	15

Welche Variante gewählt wird, ist letztendlich eine Frage, was für spezifische Anforderungen an die Formatierung des Abkürzungsverzeichnis gestellt werden. Natürlich ist auch die Definition eines eigenen Stils möglich. Wenn Sie wissen wollen, wie Sie einen Stil in Tabellenform erstellen können, welcher einen Seitenumbruch zulässt, sollte Sie den nächsten Abschnitt lesen.

4.5.2 Symbolverzeichnis

Für das Erzeugen eines Symbolverzeichnisses kann ebenfalls **glossaries** verwendet werden. Allerdings muss dazu ein wenig mehr Aufwand getrieben werden, da das Paket hierfür keine dedizierte Schnittstelle bereitstellt. Wurde die Paketoption `symbols` angegeben wird jedoch zumindest das notwendige Glossar erstellt.

Als erstes sollte ein gut nutzbarer Befehl zum Definieren eines neuen Symbols erstellt werden. In Anlehnung an den Befehl für Abkürzungen `\newacronym` wird er `\newformulasymbol` genannt. Dieser hat ein optionales und vier obligatorische Argumente, wobei das optionale Argument prinzipiell alle Schlüssel-Wert-Paare enthalten kann, die durch das Paket **glossaries** akzeptiert werden. Welche davon letztlich auch Auswirkungen haben, hängt allerdings von der Gestaltung des Stils durch den Anwender ab. Der Befehl hat folgende Definition:

`\newformulasymbol[<Parameterliste>]{<Label>}{<Name>}{<Symbol>}{<Einheit>}`

Mit *<Label>* erfolgt die eindeutige Kennzeichnung des Symbols. Außerdem wird dies standardmäßig für die Sortierung verwendet, was unter Umständen etwas problematisch sein könnte. In diesem Fall ist eine manuelle Festlegung für den dazugehörigen Schlüssel durch den Anwender über das optionale Argument mit `sort=`*<Sortierung>* eventuell sinnvoll. Nach dem *<Label>* schließt der *<Name>* für das Formelzeichen an, gefolgt vom *<Symbol>* sowie der dazugehörige physikalische *<Einheit>*.

```
\newcommand*{\newformulasymbol}[5][]{%
  \newglossaryentry{#2}{%
    type=symbols,%
    name={#3},%
    description={\nopostdesc},%
    symbol={\ensuremath{#4}},%
    user1={\ensuremath{\mathrm{#5}}},%
    sort={#2},%
    #1%
  }%
}
```

Da es sich zumeist um mathematische Symbole handelt, wird für das Symbol und die Einheit mit `\ensuremath` Sorge getragen, dass diese auch im Textmodus ohne Probleme verwendet werden können. Für das aufrechte Setzen der Einheit wird für diese außerdem `\mathrm` verwendet. Als Alternative dazu könnte der Befehl `\si` aus dem Paket **siunitx** genutzt werden. Für ein kleines Beispiel werden folgend fünf Formelzeichen definiert. Damit die Darstellung der mathematischen Brüche auch für den Fließtext gut genutzt werden kann und ansehnlich ist, wird für diese der Befehl `\sfrac` aus dem Paket **xfrac** genutzt.

```
\newformulasymbol{l}{Länge}{l}{m}
\newformulasymbol{m}{Masse}{m}{kg}
\newformulasymbol{a}{Beschleunigung}{a}{\sfrac{m}{s^2}}
\newformulasymbol{t}{Zeit}{t}{s}
\newformulasymbol{f}{Frequenz}{f}{s^{-1}}
\newformulasymbol{F}{Kraft}{F}{m \cdot kg \cdot s^{-2} = \sfrac{J}{m}}
```

Die soeben definierten Symbole werden für ein kleines Beispiel mit dem Befehl `\gls{<Label>}` in einer kurzen Textpassage verwendet.

```
Die Einheiten für die \gls{f} sowie die \gls{F} werden aus den
SI"=Einheiten der Basisgrößen \gls{l}, \gls{m} und \gls{t} abgeleitet.
Und dann gibt es noch die Grundgleichung der Mechanik, welche für den
Fall einer konstanten Kraftwirkung in die Bewegungsrichtung einer
Punktmasse lautet:
\[\gls{F} = \gls{m} \cdot \gls{a}\]
```

Die Einheiten für die Frequenz sowie die Kraft werden aus den SI-Einheiten der Basisgrößen Länge, Masse und Zeit abgeleitet. Und dann gibt es noch die Grundgleichung der Mechanik, welche für den Fall einer konstanten Kraftwirkung in die Bewegungsrichtung einer Punktmasse lautet:

$$Kraft = Masse \cdot Beschleunigung$$

Das Ergebnis ist nur bedingt befriedigend. Sowohl im Fließtext als auch im Mathematikmodus werden lediglich die Bezeichnungen jedoch nicht die Symbole selbst verwendet. Damit die Formelzeichen auch für den mathematischen Satz sinnvoll nutzbar sind, sollte das Erscheinungsbild der Einträge mit `\defglsentryfmt` angepasst werden.

```
\defglsentryfmt[symbols]{%
  \ifmmode%
    \glssymbol{\glslabel}%
  \else%
    \glsgenentryfmt~\glsentrysymbol{\glslabel}%
  \fi%
}
```

Bei der Verwendung von `\gls{<Label>}` führt diese Definition dazu, dass im Mathematikmodus allein das Symbol verwendet wird. Im Fließtext wird diesem zusätzlich die Bezeichnung vorangestellt. Das nachfolgende Beispiel macht dies deutlich.

```
Die Einheiten für die \gls{f} sowie die \gls{F} werden aus den
SI"=Einheiten der Basisgrößen \gls{l}, \gls{m} und \gls{t} abgeleitet.
Und dann gibt es noch die Grundgleichung der Mechanik, welche für den
Fall einer konstanten Kraftwirkung in die Bewegungsrichtung einer
Punktmasse lautet:
\[\gls{F} = \gls{m} \cdot \gls{a}\]
```

Die Einheiten für die Frequenz f sowie die Kraft F werden aus den SI-Einheiten der Basisgrößen Länge l, Masse m und Zeit t abgeleitet. Und dann gibt es noch die Grundgleichung der Mechanik, welche für den Fall einer konstanten Kraftwirkung in die Bewegungsrichtung einer Punktmasse lautet:

$$F = m \cdot a$$

Als nächstes kümmern wir uns um die Ausgabe des Symbolverzeichnisses. Momentan erzeugt der Befehl `\printsymbols` jedenfalls kein sinnvolles Verzeichnis:

```
\printsymbols
```

Symbole

Für dieses muss erst ein Stil definiert werden, was nachfolgend ähnlich zum Stil `acrotabularx` respektive `acrotabu` geschieht. Allerdings wird hier eine Variante gezeigt, mit der die Tabelle einen Seitenumbruch zulässt.

Eigener Stil mit `tabularx` und `longtable`

Soweit mir bekannt ist, lassen sich umbruchfähige Tabellen nicht direkt mit `tabularx` setzen. Vielmehr ist für die Verwendung der Umgebung `longtable` das Paket **ltxtable** notwendig. Dieses wiederum verlangt, dass die zu setzende Tabelle in einer separaten Datei abgelegt wird. Soll diese dennoch innerhalb des Hauptdokumentes erstellt werden, kann für dieses Unterfangen die Umgebung `filecontents` aus dem gleichnamigen Paket genutzt werden, womit *bereits existierende Dateien* überschrieben werden.

Leider habe ich keine Möglichkeit gefunden, diesen Prozess ohne wahnsinnig großen Aufwand für das Erstellen eines Glossars zu portieren. Für umbruchfähige Tabellen mit automatisch berechneten Spaltenbreiten kommt meines Wissens nach momentan nur die Umgebung `longtabu` aus dem Paket **tabu** infrage.

Eigener Stil mit `longtabu`

Das Paket **tabu** definiert für umbruchfähige Tabellen die Umgebung `longtabu`, welche wiederum auf der Umgebung `longtable` basiert und nachfolgend verwendet wird. Damit diese linksbündig gesetzt wird, muss vor dem obligatorischen Argument mit den Spaltendefinitionen noch das optionale Argument `[<l>]` angegeben werden. Die im nachfolgend definierten Stil verwendeten Befehle `\toprule`, `\midrule` sowie `\cmidrule` und `\bottomrule` für unterschiedliche horizontale Linien stammen allesamt aus dem Paket **booktabs**, welches für den Satz von hochwertigen Tabellen eine große Hilfe ist.

```
\newglossarystyle{symblongtabu}{%
  \renewenvironment{theglossary}{%
    \begin{longtabu}[l]{ccX<{\strut}l}% 'spread 0pt' defekt in v2.9
  }{%
    \end{longtabu}%
  }%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand*{\glsgroupskip}{}%
  \renewcommand*{\glossaryheader}{%
    \toprule
    \bfseries Symbol & \bfseries Einheit &
    \bfseries Bezeichnung & \bfseries Seite(n)
    \tabularnewline\midrule\endhead%
    \bottomrule\endfoot%
  }%
  \renewcommand*{\glossentry}[2]{%
    \glsentryitem{##1}% Entry number if required
    \glstarget{##1}{\glossentrysymbol{##1}} &
    \glsentryuseri{##1} &
    \glossentryname{##1} &
    ##2\tabularnewline%
  }%
}
```

Innerhalb von `\newglossarystyle` wird der Befehl `\glossaryheader` für einen Tabellenkopf definiert, wie er auch für eine `longtable`-Umgebung erscheinen würde. In hier vorgestellten Fall werden der Kopf mit dem Makro `\endhead` beziehungsweise der Fuß durch `\endfoot` terminiert. Diese werden beim einem möglichen Seitenumbruch zu Beginn und am Ende auf jeder Seite gesetzt. Das Symbolverzeichnis kann sich nun durchaus sehen lassen.

```
\printsymbols[style=symblongtabu]
```

Symbole

Symbol	Einheit	Bezeichnung	Seite(n)
a	m/s^2	Beschleunigung	19, 20
F	$\mathrm{m} \cdot \mathrm{kg} \cdot \mathrm{s}^{-2} = \mathrm{J}/\mathrm{m}$	Kraft	19, 20
f	s^{-1}	Frequenz	19, 20
l	m	Länge	19, 20
m	kg	Masse	19, 20
t	s	Zeit	19, 20

Unterteilung nach griechischen und lateinischen Symbolen

Oftmals wird es für Abschlussarbeiten verlangt, das Symbolverzeichnis nach griechischen und lateinischen Formelzeichen zu untergliedern. Für die Umsetzung dieser Anforderung ist die zuvor beschriebene Lösung prinzipiell nutzbar, allerdings muss diese etwas erweitert werden.

Realisiert wird das Ganze, indem die zwei Kategorien respektive Elterneinträge `greekletters` sowie `romanletters` definiert und sämtliche Formelzeichen einem der beiden als Untereintrag zugeordnet werden.

```
\providecommand*\greeklettersname{Greek letters}
\providecommand*\romanlettersname{Roman letters}
\newglossaryentry{greekletters}{%
  type=symbols,%
  name={\greeklettersname},%
  description={\nopostdesc},%
  sort={a}%
}
\newglossaryentry{romanletters}{%
  type=symbols,%
  name={\romanlettersname},%
  description={\nopostdesc},%
  sort={b}%
}
```

Die Bezeichnungen der beiden Kategorien könnten auch als Parameter von `\newglossaryentry` mit `name=<Bezeichung>` direkt eingetragen werden. Um bei der Definition jedoch eine gewisse Flexibilität zu gewährleisten, wird dies über die zwei Befehle `\greeklettersname` sowie `\romanlettersname` realisiert. Diese werden mit KOMA-Script-Mitteln als sprachabhängige Bezeichner definiert.

```
\providecaptionname{%
  american,australian,british,canadian,english,newzealand%
}{\greeklettersname}{Greek letters}
\providecaptionname{%
  german,ngerman,austrian,naustrian,swissgerman,nswissgerman%
}{\greeklettersname}{Griechische Symbole}
\providecaptionname{%
  american,australian,british,canadian,english,newzealand%
}{\romanlettersname}{Roman letters}
\providecaptionname{%
  german,ngerman,austrian,naustrian,swissgerman,nswissgerman%
}{\romanlettersname}{Lateinische Symbole}
```

Für die Ausgabe wird der Glossarstil `symbsplitlongtabu` in ähnlicher Manier wie die Stile zuvor definiert. Dieser wird darauf ausgelegt, dass alle aufzulistenden Formelzeichen als Untereintrag von einem der beiden zuvor erstellten Elterneinträge angegeben werden. Allerdings ist bei diesem Stil aufgrund der Verwendung von `longtabu` etwas Aufwand zu betreiben, um die gewünschte Ausgabe zu erhalten. Das grundlegende Problem ist das Setzen von Überschriften und dem folgenden Tabellenkopf innerhalb der Tabelle. Die Erläuterung des Quelltextauszugs erfolgt nach dessen Ausgabe.

```
\newglossarystyle{symbsplitlongtabu}{%
  \newcommand*\symbollevel{-1}%
```

```
\renewenvironment{theglossary}{%
  \begin{longtabu}[l]{ccX<{\strut}l}% 'spread 0pt' defekt in v2.9
}{%
  \end{longtabu}%
}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glsgroupskip}{}%
\newcommand*\symbolhead{%
  \toprule
  \bfseries Symbol & \bfseries Einheit &
  \bfseries Bezeichnung & \bfseries Seite(n)
  \tabularnewline\midrule
}%
\renewcommand*{\glossaryheader}{%
  \endfirsthead%
  \symbolhead\endhead%
  \bottomrule\endfoot%
  \gdef\symbollevel{-1}%
}%
\renewcommand*{\glossentry}[2]{%
  \ifglshaschildren{##1}{%
    \ifnum\symbollevel>0\relax%
      \tabularnewline\bottomrule\tabularnewline[\smallskipamount]%
    \fi
    \gdef\symbollevel{0}%
    \tabularnewline[%
      \arraystretch\dimexpr-\ht\strutbox-\dp\strutbox\relax%
    ]%
    \multicolumn{4}{@{}l@{}}{\minisec{\glsentryname{##1}}}%
  }{%
    \GlossariesWarning{%
      There are no childrens for entry ##1.\MessageBreak
      Nothing will be printed. Maybe you should\MessageBreak
      specify ##1 as a child entry.
    }%
  }%
}%
\renewcommand*{\subglossentry}[3]{%
  \ifnum\symbollevel=0\relax%
    \tabularnewline[\medskipamount]\symbolhead%
  \else%
    \tabularnewline
  \fi%
  \gdef\symbollevel{##1}%
  \glsentryitem{##2}% Entry number if required
  \glstarget{##2}{\glossentrysymbol{##2}} &
  \glsentryuseri{##2} &
```

```
      \glossentryname{##2} &
      ##3%
    }%
  }
```

Der neu definierte Befehl `\symbollevel` wird benötigt, um bei der Ausgabe zu überprüfen, um es sich beim zuletzt gesetzten Eintrag um einen Elterneintrag (`\glossentry`) oder Untereintrag (`\subglossentry`) gehandelt hat. Im Makro `\symbolhead` wird die Definition für den zu setzenden Tabellenkopf gespeichert. Der Inhalt von `\glossaryheader` wird direkt zu Beginn der Umgebung `theglossary` ausgeführt. Dieser wird so umdefiniert, dass die Tabelle keinen Kopf vor der ersten Überschrift erhält, jedoch nach einem Seitenumbruch sehr wohl der Tabellenkopf gesetzt wird. Abgeschlossen wird die Tabelle mit einer Fußlinie.

Danach erfolgt die Formatierung der Einträge. Der Befehl `\glossentry` setzt „Griechische Symbole“ und „Lateinische Symbole“ sowie gegebenenfalls weitere Elterneinträge als Überschriften. Bevor dies passieren kann wird eine vorher begonnene Tabelle mit einer Fußlinie beendet. Um die Überschrift innerhalb von `longtabu` linksbündig zu setzen, wird mit `\multicolumn` gearbeitet. Dieser Befehl muss in der ersten Zelle zwingend als aller erstes genutzt werden. Um dies sicherzustellen, wird zuvor eine neue Tabellenspalte eingefügt. Damit jedoch kein zusätzlicher Leerraum entsteht, wird diese über das optionale Argument nach oben verschoben. Die Einträge für Formelzeichen selber erfolgen mit `\subglossentry`. Im Gegensatz zum vorher beschriebenen Stil `symblongtabu` wird zusätzlich der notwendige Tabellenkopf nach der Überschrift gesetzt, falls es sich um den *ersten* Untereintrags handelt.

Jetzt könnte man den Glossarstil `symbsplitlongtabu` bereits verwenden und jedem Formelzeichen bei der Definition mit

```
\newformulasymbol[parent=greekletters/romanletters]{...}{...}{...}{...}
```

eine der beiden Elterneinträge zuweisen. Für wenige Formelzeichen mag dies ausreichen. Komfortabler wäre es natürlich, wenn die Zuweisung zu einer der beiden Kategorien automatisch erfolgen würde. Ein Ansatz für dieses Unterfangen wird folgend vorgestellt.

Achtung!

Es kann nicht garantiert werden, dass dieser Ansatz in jedem Fall und für sämtliche Anwendungsszenarien funktioniert. Tatsächlich ist die Idee auf eine Frage aus dem TUD-LaTeX-Forum[8] heraus entstanden. Für auftretende Probleme bei der Verwendung ist dort der beste Anlaufpunkt.

Die Grundidee ist, das zu definierende Formelzeichen mit einer Liste der Makros für griechische Buchstaben abzugleichen und so zu entscheiden, in welche der beiden Kategorien das Symbol einzuordnen ist. Hierfür werden nachfolgend eine Liste `\greeksymbollist` aller griechischen Buchstaben sowie der Befehl `\ifisgreeksymbol{<Symbol>}{<Dann>}{<Sonst>}` zur Fallunterscheidung definiert. Problematisch bei der Umsetzung ist, dass ein als Argument übergebenes Formelzeichen auch mit diakritischen Zeichen und Indizes verwendet werden kann – beispielsweise $\dot{\varphi}_{\mathrm{max}}$ (`\dot{\varphi}_{\mathrm{max}}`). Deshalb werden bei der Ausführung von `\ifisgreeksymbol` temporär alle diakritische Zeichen sowie `\boldsymbol` für den Mathematikmodus „unschädlich“ gemacht und das Argument `{<Symbol>}` expandiert. Anschließend werden alle Anteile für Hoch- sowie Tiefstellung abgeschnitten und das Resultat mit der Liste `\greeksymbollist` abgeglichen.

[8] http://latex.wcms-file3.tu-dresden.de/phpBB3/viewtopic.php?f=11&t=427

```
\makeatletter
\newcommand*\greeksymbollist{}
\def\@tempa#1{\ifdefvoid{#1}{}{\listadd\greeksymbollist{#1}}}
\forcsvlist{\@tempa}{%
  \alpha,\beta,\varbeta,\gamma,\delta,\epsilon,\varepsilon,\zeta,%
  \eta,\theta,\vartheta,\iota,\kappa,\varkappa,\lambda,\mu,\nu,%
  \xi,\omicron,\pi,\varpi,\rho,\varrho,\sigma,\varsigma,\tau,%
  \upsilon,\phi,\varphi,\chi,\psi,\omega,%
  \Alpha,\Beta,\Gamma,\Delta,\Epsilon,\Zeta,\Eta,\Theta,\Iota,%
  \Kappa,\Lambda,\Mu,\Nu,\Xi,\Omicron,\Pi,\Rho,\Sigma,\Tau,\Upsilon,%
  \Phi,\Chi,\Psi,\Omega%
}
\newcommand*\ifisgreeksymbol[1]{%
  \begingroup%
    \def\@tempa##1{\let##1\@firstofone}%
    \forcsvlist{\@tempa}{%
      \acute,\bar,\breve,\check,\dot,\ddot,\dddot,\ddddot,%
      \hat,\widehat,\grave,\tilde,\widetilde,\vec,%
      \Acute,\Bar,\Breve,\Check,\Dot,\Ddot,\Hat,\Grave,\Tilde,\Vec%
    }%
    \def\boldsymbol{}%
    \protected@edef\@tempa{#1}%
    \def\@tempb##1_##2\relax{\def\@tempa{##1}}%
    \expandafter\@tempb\@tempa_\relax\relax%
    \def\@tempb##1^##2\relax{\def\@tempa{##1}}%
    \expandafter\@tempb\@tempa^\relax\relax%
    \expandafter\ifinlist\expandafter{\@tempa}{\greeksymbollist}{%
      \aftergroup\@firstoftwo%
    }{%
      \aftergroup\@secondoftwo%
    }%
  \endgroup%
}
\makeatother
```

Mit `\ifisgreeksymbol` kann jetzt auf die Art des Symbols getestet. Dies wird innerhalb von `\newformulasymbol` genutzt, um den passenden Elterneintrag auszuwählen. Über das optionale Argument kann jederzeit mit `[parent=greekletters/romanletters]` der automatische ermittelte Elterneintrag überschrieben werden.

```
\newcommand*\symbollettergroup{}
\newcommand*{\newformulasymbol}[5][]{%
  \ifisgreeksymbol{#4}{%
    \renewcommand*{\symbollettergroup}{greekletters}%
  }{%
    \renewcommand*{\symbollettergroup}{romanletters}%
  }%
  \newglossaryentry{#2}{%
```

```
      type=symbols,%
      name={#3},%
      description={\nopostdesc},%
      symbol={\ensuremath{#4}},%
      user1={\ensuremath{\mathrm{#5}}},%
      sort={#2},%
      parent={\symbollettergroup},%
      #1%
    }%
}
```

4.6 Literaturverzeichnis

Für das Erstellen eines Literaturverzeichnisses wurde in der Vergangenheit fast ausschließlich BibTeX verwendet. Leider wird auch heute immer noch darauf verwiesen, obwohl es seit einigen Jahren das Paket **biblatex** gibt, welches insbesondere für neue Dokumente den Vorzug erhalten sollte. Auch die Umstellung älterer BibTeX-Datenbanken ist mit wenigen Handgriffen realisierbar. Für **biblatex** existieren eine Menge unterschiedlicher, vordefinierter Zitierstile, welche sich im Vergleich zu BibTeX auch wesentlich leichter an die individuellen Bedürfnisse anpassen lassen.

Ein weiterer Vorteil ist die Unterstützung von Datenbanken, welche eine UTF-8-Kodierung nutzen, wenn ***biber*** zur Sortierung der Einträge verwendet wird. Zitierstil und Backend zur Sortierung lassen sich durch das optionale Argument beim Laden des Paketes festlegen. Damit die Zitierstile das optimale Ergebnis erzielen wird das Laden von **csquotes** sehr empfohlen.

```
\usepackage{csquotes}
\usepackage[backend=biber,style=alphabetic]{biblatex}
```

Die Erstellung einer Literaturdatenbank kann entweder von Hand oder mithilfe einer externen Anwendung erfolgen. Für die letztgenannte Variante sind die Programme ***Citavi*** respektive ***JabRef*** empfehlenswert. Auf eine Einführung in diese Anwendungen wird jedoch verzichtet.

Die `filecontents`-Umgebung kann verwendet werden, um innerhalb eines LaTeX-Dokumentes externe Dateien direkt beim Kompilieren zu erstellen. Damit wird nachfolgend für dieses Tutorial eine Literaturdatenbank **treatise-temp.bib** mit drei Einträgen manuell erzeugt. Die Umgebung gehört standardmäßig zu den Bordmitteln von LaTeX 2_ε. Das Paket **filecontents** erweitert die Umgebung dahingehend, dass *bereits existierende Dateien* überschrieben werden. Hier ist folglich Vorsicht geboten. Der große Vorteil ist jedoch, dass die erweiterte `filecontents`-Umgebung – im Gegensatz zur Standardversion – die Dateien in der gleichen Eingabekodierung erzeugt, wie das verwendete Dokument. Diese Funktionalität wird für dieses Tutorial benötigt, weshalb auf das Laden des Paketes **filecontents** nicht verzichtet werden kann.

```
\usepackage{filecontents}
```

Folgend wird eine Literaturdatenbank mit drei Einträge definiert. Jeder Eintrag einer **.bib**-Datei beginnt mit `@<Eintragstyp>`. Direkt danach ist für jeden Eintrag ein *eindeutiges* `{Label}` festzulegen. Anschließend können für unterschiedliche Felder die dazugehörigen Werte eingetragen werden. Die verwendbaren Eintragstypen sowie die für den jeweiligen Typ obligatorischen und optionalen Felder sind in der Dokumentation von **biblatex** zu finden.

```
\begin{filecontents}{\jobname-temp.bib}
@book{goossens94,
  author    = {Goossens, Michel and Mittelbach, Frank
               and Samarin, Alexander},
  title     = {The LaTeX Companion},
  date      = {1994},
  publisher = {Addison-Wesley},
  location  = {Reading, Massachusetts},
  language  = {english},
}
@book{knuth84,
  author    = {Knuth, Donald E.},
  title     = {The \TeX book},
  date      = {1984},
  maintitle = {Computers \& Typesetting},
  volume    = {A},
  publisher = {Addison-Wesley},
  location  = {Reading, Massachusetts},
  language  = {english},
}
@manual{hanisch14,
  author    = {Hanisch, Falk},
  title     = {Ein \LaTeX"=Bundle für Dokumente im neuen Corporate
               Design der Technischen Universität Dresden},
  date      = {2014},
  subtitle  = {Benutzerhandbuch},
  location  = {Dresden},
  language  = {german},
}
\end{filecontents}
```

Nachdem die Literaturdatenbank erstellt wurde, muss diese auch noch eingebunden werden:

```
\addbibresource{\jobname-temp.bib}
```

Im einfachsten Fall werden die gewünschten Einträge der Literaturdatenbank im Dokument mit `\cite{<macro>}` referenziert, von **biblatex** werden zusätzliche Möglichkeiten angeboten.

```
In diesem Textabschnitt werden die zwei bekannten \LaTeX-Bücher
\cite{knuth84} und \cite{goossens94} sowie das Anwenderhandbuch
\cite{hanisch14} zitiert.
```

In diesem Textabschnitt werden die zwei bekannten LaTeX-Bücher [Knu84] und [GMS94] sowie das Anwenderhandbuch [Han14] zitiert.

Das Literaturverzeichnis wird mit `\printbibliography` ausgegeben, wobei nicht alle Einträge der Literaturdatenbank sondern lediglich die tatsächlich referenzierten verwendet werden. Der Parameter `heading=bibintoc` führt zu einem Eintrag in das Inhaltsverzeichnis.

```
\printbibliography[heading=bibintoc]
```

Literatur

[GMS94] Michel Goossens, Frank Mittelbach und Alexander Samarin. *The LaTeX Companion*. Englisch. Reading, Massachusetts: Addison-Wesley, 1994.

[Han14] Falk Hanisch. *Ein LaTeX-Bundle für Dokumente im neuen Corporate Design der Technischen Universität Dresden. Benutzerhandbuch*. Dresden, 2014.

[Knu84] Donald E. Knuth. *Computers & Typesetting*. Bd. A: *The TeXbook*. Englisch. Reading, Massachusetts: Addison-Wesley, 1984.

5 Anfangszitat oder Schlauer Spruch

Es irrt der Mensch, solang er strebt.

(Johann Wolfgang von Goethe)

Oftmals möchte der Autor einer wissenschaftlichen Arbeit für das erste oder auch jedes Kapitel ein Zitat oder ähnliches voranstellen. Dies kann mit dem Befehl `\dictum[<Autor>]{<Text>}` erfolgen. Damit wird der im obligatorischen Argument angegeben Ausspruch in einer `\parbox` ausgegeben. Das optionale Argument kann für die Angabe des Autors verwendet werden. Soll das Ganze für einen Teil oder ein Kapitel erfolgen, sollte der Befehl `\dictum` innerhalb von `\setpartpreamble` beziehungsweise `\setchapterpreamble` verwendet werden. Genaueres hierzu und zu den weiteren Möglichkeiten, die Gestalt eines Zitats zu beeinflussen, ist in der Anleitung von KOMA-Script[9] zu finden. Es folgt ein Beispiel zur Verwendung.

```
\setchapterpreamble{%
  \dictum[Johann Wolfgang von Goethe]{%
    Es irrt der Mensch, solang er strebt.%
  }%
  \bigskip
}
\chapter{Einleitung}
```

6 Gleitumgebungen für Abbildungen und Tabellen

Die Positionierung von Abbildungen mit LaTeX kann zu Beginn für viele Anfänger durchaus frustrierend sein. Das liegt häufig am Missverständnis der beiden Standard-Gleitobjektumgebungen für Tabellen (`table`) und Abbildungen (`figure`). Diese sind in erster Linie zur Ergänzung des Fließtextes gedacht und sollten für das prinzipielle Verständnis des Geschriebenen nicht notwendig sein. Das oft geforderte Verhalten, ein Gleitobjekt an einer ganz bestimmten und explizit festgelegten Position im Text zu setzen, ist nicht erforderlich, insbesondere weil dadurch der Lesefluss unnötig unterbrochen wird.

Vielmehr ist es sinnvoll, Gleitobjekte entweder am Anfang oder Ende einer Seite zu platzieren, wo sie den Lesefluss deutlich weniger stören. Allerdings sollte auf jedes Gleitobjekt im Fließtext über eine Referenz – beispielsweise mit dem Befehl `\autoref` aus dem Paket **hyperref** –

[9] http://mirrors.ctan.org/macros/latex/contrib/koma-script/doc/scrguide.pdf

Abbildung 5: Beispielgrafik

Bezug genommen und gegebenenfalls eine kurze Erläuterung gegeben werden. Zusätzliche Erläuterungen zum Thema *Querverweise* sind unter Unterkapitel 8 zu finden.

Ein weitere Grund, LaTeX die Platzierung von Tabellen und Abbildungen vollständig zu überlassen, ist die Ungewissheit über den vorhandenen Platz auf der momentan erzeugten Seite. In nicht wenigen Fällen kann es passieren, dass das einzufügende Objekt zu groß für die aktuelle Seite ist, was einen wahrlich schlechten Seitenumbruch mit einer schlecht gefüllten Seite zur Folge hätte. Die Verwendung einer Gleitumgebung für eine Abbildung wird im nachfolgenden Quelltextauszug exemplarisch gezeigt, das Ergebnis ist in Abbildung 5 zu sehen. Bei Gleitobjekten sollte in jedem Fall darauf geachtet werden, dass der Befehl `\label` immer erst *nach* `\caption` verwendet wird, da der erzeugte Anker sich sonst nicht auf das Objekt bezieht.

```
\begin{figure}
\centering
\includegraphics{TUD-black}
\caption{Beispielgrafik}\label{fig:example}
\end{figure}
```

In der Konsequenz ist dies hier ein Plädoyer, bei der Platzierung von Tabellen und Abbildungen vollständig auf LaTeX zu vertrauen. Anfangs kann der Anwender dies als Kontrollverlust empfinden. Sobald Sie jedoch Änderungen am Dokument vornehmen und eventuell in einem Kapitel einen Absatz ergänzen oder gar einen ganzen Abschnitt hinzufügen werden sie dankbar sein, sich in der Folge nicht mit der Neupositionierung sämtlicher Objekte herumschlagen zu müssen. Leidgeprüfte Anwender einschlägiger Textverarbeitungsprogramme können gewiss ein Lied darüber singen.

6.1 Gleitobjektlayout

Es wurde bis jetzt das prinzipielle Vorgehen bei der Nutzung von Gleitobjekten beschrieben. Allerdings gibt es auch noch einige typografische Aspekte, welche zu beachten sind. Zum einen sollte beachtet werden, dass die Beschriftung einer Abbildung immer unterhalb dieser erfolgen sollte, bei Tabellen hingegen eine Überschrift gesetzt wird. Hierfür muss der Anwender ohne die Verwendung eines zusätzlichen Paketes selber Sorge tragen, indem er den Befehl `\caption` entweder vor oder nach dem eigentlichen Objekt in der Gleitumgebung verwendet. Der zweite Punkt ist die verwendete Schrift innerhalb der Gleitumgebungen.

Damit sich der Inhalt dieser besser vom restlichen Fließtext abhebt und vom Leser direkt als nicht dazugehörig erkannt werden kann ist es ratsam, diesen in serifenloser Schrift zu

Erste Spalte	Zweite Spalte	Dritte Spalte
Etwas Blindtext für die erste Spalte	Etwas Blindtext für die zweite Spalte	Etwas Blindtext für die dritte Spalte

Tabelle 1: Eine Tabelle in einer Gleitumgebung

Tabelle 2 Eine Tabelle in einer Gleitumgebung mit einer angepassten Formatierung

Erste Spalte	Zweite Spalte	Dritte Spalte
Etwas Blindtext für die erste Spalte	Etwas Blindtext für die zweite Spalte	Etwas Blindtext für die dritte Spalte

setzen – die Verwendung einer Serifenschrift für den Fließtext vorausgesetzt. Das Paket **floatrow** bietet die Möglichkeiten, diese beiden Punkte automatisiert umzusetzen. Außerdem wird zur Formatierung der Beschriftungen das Paket **caption** benötigt. Sollten Sie noch keine Erfahrung mit dem Setzen von Tabellen mit LaTeX haben, so wäre zuvor ein Blick in Unterkapitel 7 sehr sinnvoll.

Normalerweise wird eine Tabelle sowie deren Beschriftung innerhalb einer `table`-Gleitumgebung in der selben Schrift wie der Fließtext gesetzt, die korrekte Platzierung der Bezeichnungen bleibt dem Anwender überlassen. Zu sehen ist dies in Tabelle 1. Im Folgenden wird gezeigt, wie sich dies automatisiert für alle Gleitobjekte im Dokument ändern lässt. Die *Formatierung* der Bezeichnungen von Gleitobjekten kann mithilfe des Paketes **caption** angepasst werden.

```
\usepackage{caption}
```

Diese stellt den Befehl `\captionsetup` bereit. Mit diesem werden die Beschriftungen in Serifenlosen und das Label zur besseren Unterscheidung im fetter Schriftstärke gesetzt.

```
\captionsetup{font=sf,labelfont=bf,labelsep=space}
```

Um die *Platzierung* der Beschriftungen beeinflussen sowie den Inhalt einer Gleitumgebung formatieren zu können, wird **floatrow** geladen.

```
\usepackage{floatrow}
```

Der Inhalt von Gleitobjekten wird – entgegen des normalen Verhaltens – durch **floatrow** automatisch zentriert gesetzt. Zusätzlich soll der Inhalt noch die serifenlose Schriftfamilie verwenden. Dies lässt sich mit der Nutzung des Befehls `\floatsetup` realisieren.

```
\floatsetup{font=sf}
```

Ohne weitere Maßnahmen werden durch **floatrow** alle Beschriftungen unterhalb der Gleitobjekte ausgegeben. Da das Ziel jedoch die Verwendung von Tabellenüberschriften und Abbildungsunterschriften ist, wird dies für Tabellen angepasst.

```
\floatsetup[table]{style=plaintop}
```

Damit wären alle Einstellungen soweit erfolgt, in Tabelle 2 ist das Ergebnis zu sehen. Zumeist soll die Beschriftung für Gleitobjekte nicht zentriert sondern linksbündig gesetzt werden. Dies

lässt sich ebenfalls mit den Mitteln des Paketes **caption** und der passenden Paketoption `justification` erreichen. Damit dies auch für einzeilige Beschriftungen angewendet wird, muss zusätzlich die Einstellung `singlelinecheck` deaktiviert werden. Mit der Paketoption `format` kann außerdem beeinflusst werden, ob eine mehrzeilige Beschriftung nach einem Zeilenumbruch direkt unter dem Label oder mit Einzug fortgesetzt werden soll.

```
\captionsetup{singlelinecheck=off,format=hang,justification=raggedright}
```

In Tabelle 2 fällt die Beschriftung auf, welche die Breite der Tabelle überragt. Falls linksbündige Gleitumgebungsbezeichnungen verwendet werden, wie dies gerade eingestellt wurde, besteht das Problem quasi bei jeder Beschriftung, da diese immer am linken Seitenrand beginnen. Auch hierfür stellt das Paket **floatrow** die Befehle `\ttabbox` für Tabellen sowie `\ffigbox` für Abbildungen bereit.

Beide Makros erwarten im ersten obligatorischen Argument den Befehl `\caption` für die Beschriftung gegebenenfalls gefolgt von einem mit `\label` gesetzten Textanker. Im zweiten Argument wird das Objekt selbst – sprich Tabelle oder Abbildung – angegeben. Zu verwenden sind diese folgendermaßen:

```
\begin{table}
\ttabbox
  {<Tabellendefinition>}
  {\caption{<Beschriftung>}\label{<Label>}}
\end{table}
\begin{figure}
\ffigbox
  {<Abbildungsdefinition>}
  {\caption{<Beschriftung>}\label{<Label>}}
\end{figure}
```

Diese beiden Befehle sorgen dafür, dass die Beschriftungen in der jeweiligen Breite des Objektes gesetzt werden. Entspricht das zu setzende Objekt in seiner Breite dem Wert `\textwidth` – es ist genauso Breit wie die Laufweite des Fließtextes –, so ist deren Gebrauch nicht notwendig. Das Ergebnis der Verwendung von des Befehls `\ttabbox` ist in Tabelle 3 zu sehen.

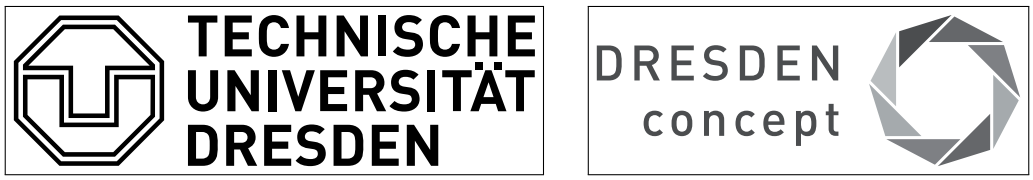
Bei der Nutzung der beiden Befehle sollte darauf geachtet werden, dass zwischen dem Ende des Objektes und der schließender Klammer des zweiten Argumentes kein ungewolltes Leerzeichen gesetzt wird. Dies lässt sich leicht vermeiden, indem direkt nach dem Ende des Objektes mit `\end{tabularx}`, `\end{tabu}`, `\end{tikz}` etc. die schließende Klammer des zweiten Argumentes oder das Zeichen für einen Kommentar `%` folgt.

6.2 Untergleitobjekte

Das Paket **floatrow** stellt allerhand Möglichkeiten zur Erstellung von nebeneinanderliegenden Gleitobjekten sowie „Untergleitobjekten“ bereit und ist insbesondere in Verbindung mit den Fähigkeiten von **caption** respektive **subcaption** sehr gut für dieses Unterfangen nutzbar. Dafür werden die Umgebungen `floatrow` sowie `subfloatrow` angeboten, wobei die letztere für ein Beispiel verwendet wird. Als erstes muss der Typ der zu erzeugenden Labels für Abbildungen

Tabelle 3 Eine in einer Gleitumgebung gesetzte Tabelle in Verbindung mit dem Befehl `\ttabbox`, der durch das Paket **floatrow** für Beschriftungen in Objektbreite zur Verfügung gestellt wird

Erste Spalte	Zweite Spalte	Dritte Spalte
Etwas Blindtext für die erste Spalte	Etwas Blindtext für die zweite Spalte	Etwas Blindtext für die dritte Spalte



a) Eine Abbildung **b)** Eine weitere Abbildung

Abbildung 6 Eine Gleitumgebung mit zwei Abbildungen

und Tabellen definiert werden. Dies geschieht mit dem Befehl `\DeclareCaptionSubType` aus dem Paket **subcaption**.

```
\DeclareCaptionSubType[alph]{figure}
\DeclareCaptionSubType[alph]{table}
```

Anschließend werden die Untergleitobjekte formatiert. Dabei wird zum einen nach dem alphanumerischen Label eine schließende Klammer gesetzt. Zum andern wird ein Eintragen der Untergleitobjekte in das Abbildungsverzeichnis unterdrückt. Lediglich die Bildunterschrift der Hauptabbildung wird in dieses eingetragen.

```
\captionsetup[subfloat]{labelformat=brace,list=off}
```

Mit diesen Einstellungen wird eine gleitende Abbildung erstellt, welche in Abbildung 6 zu sehen ist. Wurde der entsprechende Anker gesetzt, kann zusätzlich auf die untergeordnete Abbildung 6a sowie Abbildung 6b verwiesen werden.

```
\begin{figure}
\ffigbox[\FBwidth]%
  {\begin{subfloatrow}%
    \ffigbox[\FBwidth]%
      {\fbox{\includegraphics[height=2cm]{TUD-black}}}%
      {\caption{Eine Abbildung}\label{fig:tud}}%
    \ffigbox[\FBwidth]%
      {\fbox{\includegraphics[height=2cm]{DDC-21}}}%
      {\caption{Eine weitere Abbildung}\label{fig:ddc}}%
  \end{subfloatrow}}%
  {\caption{Eine Gleitumgebung mit zwei Abbildungen}\label{fig:logos}}%
\end{figure}
```

6.3 Beeinflussung des Gleitprozesses

Sobald das Dokument inhaltlich den finalen Zustand erreicht hat, haben Sie sich zu diesem Zeitpunkt schon mit Sicherheit an das standardmäßige Vorgehen von LaTeX gewöhnt und müssen respektive wollen gegebenenfalls nur noch bei wenigen Gleitobjekten – über das optionale Argument der Gleitumgebungen zur individuellen Empfehlung für die Platzierung – nachjustieren. Mögliche Werte sind:

`[h]` (here)
An der Stelle, wo es im Quelltext angegeben wurde – falls genügend Platz vorhanden ist

`[t]` (top)
Am oberen Ende der aktuellen oder der folgenden Seite

`[b]` (bottom)
Am unteren Ende der aktuellen Seite

`[p]` (page)
Auf einer separaten Seite für mindestens ein Gleitobjekt

Sie können eine, mehrere oder alle Optionen angeben, wobei die Reihenfolge keine Rolle spielt. Der Algorithmus arbeitet alle ihm zur Verfügung gestellten Optionen immer in der zuvor aufgezählten Reihenfolge ab, wobei diese nur als Empfehlung und nicht als Verpflichtung angesehen werden. Es gibt folglich keine Garantie, dass Ihr Vorschlag akzeptiert wird. Fügen sie dem optionalen Argument ein `!` an, so verhindern Sie, dass LaTeX weitere Optionen evaluiert. Das Gleitobjekt wird anhand der optionalen Parameter positioniert, selbst wenn dabei ein unschönes Seitenlayout entsteht. Kann das Gleitobjekt technisch unmöglich auf die angegebene Weise positioniert werden, wird dieses sowie alle folgenden aufgeschoben und am Ende des Abschnitts oder Kapitels angehängt. Dieser Effekt ist wohl in den seltensten Fällen gewollt.

Außerdem können die Pakete **flafter** sowie **placeins** genutzt werden. Das erstgenannte verhindert das Auftreten von Gleitobjekten im Dokument vor ihrer Definition im Quelltext. In der Konsequent bedeutet dies, dass die Option `[t]` Gleitobjekte nur am oberen Ende der nächsten Seite jedoch nicht auf der aktuellen zulässt. Mit dem zweiten Paket können Barrieren definiert werden, an welchen die Ausgabe aller noch in der Warteschlange befindlichen Gleitobjekte forciert wird – beispielsweise vor bestimmten Gliederungsüberschriften. Im Handbuch zu TUD-Script findet sich außerdem zum Thema „Platzierung von Gleitobjekten“ ein eigener Abschnitt mit weiterführenden Informationen.

6.4 Abstellen des Gleitprozesses

Trotz der zuvor genannten Vorteile von Gleitobjekten wird oftmals verlangt, den Gleitprozess vollständig abzustellen. Dies bringt jedoch in den meisten Fällen einige Probleme mit sich. Die Folge sind ein sehr unruhiges Erscheinungsbild des Satzspiegels aufgrund schlecht gefüllter Seiten, außerdem im Zweifelsfall viel Handarbeit mit hart kodierten Seitenumbrüchen sowie daraus resultierend ein gegenüber von Umbruchänderungen äußerst anfälliges Dokument. Sollte

es in *Einzelfällen* dennoch erforderlich sein, dass eine Abbildung oder Tabelle nicht gleitet, ist die einzig logische Konsequenz, auf die Verwendung einer Gleitumgebung gänzlich zu verzichten.

Hierfür eignet sich beispielsweise die Umgebung `center`, welche zum einen etwas Abstand vor und nach dem Objekt zum umgebenden Text erzeugt und zum anderen die Grafik respektive Tabelle und die dazugehörige Bezeichnung zusammenhält. KOMA-Script stellt für diesen Anwendungsfall das Makro `\captionof` für die Beschriftung zur Verfügung. Werden die Pakete **caption** und **hyperref** geladen, sollte weiterhin `\caption` ergänzt um ein zuvor aufgerufenes `\captionsetup{type=figure/table}` genutzt werden. Damit lassen sich Bildunter- sowie Tabellenüberschriften auch ohne die bekannten Gleitumgebungen `figure` respektive `table` erzeugen. Das Vorgehen wird folgend demonstriert, in Abbildung 7 ist das Ergebnis zu sehen.

```
\begin{center}
\captionsetup{type=figure}
\includegraphics{TUD-black}
\caption{Eine nichtgleitende Grafik in einer \texttt{center}"=Umgebung}
\label{fig:nonfloating-A}
\end{center}
```

Abbildung 7 Eine nichtgleitende Grafik in einer `center`-Umgebung

Alternativ zu diesem Ansatz kann das Paket **float** zum Einsatz kommen, welches für Gleitumgebungen den Platzierungsparameter `[H]` bereitstellt. Das Resultat ist identisch zu Abbildung 7.

```
\begin{figure}[H]
\includegraphics{TUD-black}
\caption{Eine nichtgleitende Grafik in einer \texttt{figure}"=Umgebung}
\label{fig:nonfloating-B}
\end{figure}
```

Mit den Mitteln des Paketes **floatrow** lässt sich eine etwas bessere Variante entwickeln, wobei für eine mehrmalige Anwendung ein eigenes Makro mit den passenden Argumenten definiert werden sollte. Zum besseren Verständnis des Quelltextauszuges ist die Dokumentation von **floatrow** zu beachten.

```
\floatbox[\captionsetup{type=figure}]{figure}[\FBwidth][%
  \dimexpr\FBheight+\abovedisplayskip\relax%
][b]{\includegraphics{TUD-black}}{%
  \caption{%
    Eine nichtgleitende Grafik mit \texttt{\textbackslash floatbox}%
  }%
  \label{fig:nonfloating-C}%
}
```

Abbildung 8 Eine nichtgleitende Grafik mit `\floatbox`

Es wird hier trotz der vorhergehenden Ausführungen abermals ausdrücklich empfohlen, Tabellen und Abbildungen in Gleitumgebungen zu setzen und die korrekte Platzierung dem LaTeX-Algorithmus zu überlassen.

7 Tabellensatz

Zum Thema „Tabellensatz mit LaTeX“ sind bereits zahlreiche Leitfäden[10] [RV12] im Internet zu finden. Deshalb werde ich meine Ausführungen zu diesem Punkt relativ kurz halten. Zwei Regeln sollten beim Satz von Tabellen in jedem Fall beachtet werden:

I. keine vertikalen Linien
II. keine doppelten Linien

Das Paket **booktabs** ist für den Satz von hochwertigen Tabellen eine große Hilfe und stellt die Befehle `\toprule`, `\midrule` sowie `\cmidrule` und `\bottomrule` für unterschiedliche horizontale Linien bereit.

```
\usepackage{booktabs}
```

Außerdem existiert das Paket **array**, welches mit dem Befehl `\newcolumntype` die Definition eigener Spaltentypen sowie die Verwendung von „Hooks“ vor (`>{...}`) und nach (`<{...}`) Einträgen innerhalb einer Spalte (`>{...}<`*`Spaltentyp`*`><{...}`) ermöglicht.

```
\usepackage{array}
```

Für alle im Folgenden vorgestellten Umgebungen zum Setzen von Tabellen gilt, dass die Inhalte zeilenweise angegeben werden, wobei die Einträge für die einzelnen Spalten mit `&` voneinander zu trennen sind. Das Beenden einer Tabellenzeile und der Wechsel zur nächsten erfolgt normalerweise mit `\\`. Da jedoch einige LaTeX-Pakete diesen Befehl innerhalb von Tabellen lokal ändern ist es zur Vermeidung unnötiger Fehler wesentlich sicherer, das Zeilenende ausschließlich mit `\tabularnewline` zu setzen.

Im Folgenden wird in einigen Tabellen bei der Definition der Spalten vor der ersten und nach der letzten das Konstrukt `@{}` verwendet, was das Einrücken der Tabellenspalten verhindert und im Handbuch genauer erläutert wird.

[10] http://userpage.fu-berlin.de/latex/Materialien/tabsatz.pdf

7.1 Die Standardumgebung `tabular`

Normalerweise gibt es für `tabular` vier unterschiedliche Spaltentypen. Die drei Spaltentypen `l`, `c` und `r` stehen für linksbündige, zentrierte und rechtsbündige Spalten zu Verfügung, welche allerdings keinen Zeilenumbruch erlauben. Der Inhalt wird quasi wie in einer `\mbox` gesetzt, wodurch die Spalten sehr breit werden und über den Seitenrand hinausragen können.

Der Spaltentyp `p{<Breite>}` hingegen legt die Spaltenbreite fest und setzt die Zellinhalte in einer `\parbox`, wobei diese mit ihrer obersten Zeile an der Grundlinie ausgerichtet wird. Das Paket **array** stellt außerdem die Spaltentypen `m{<Breite>}` und `b{<Breite>}` bereit, welche ebenfalls mit `\parbox` gesetzt werden, die Ausrichtung an der Grundlinie jedoch zentriert respektive an der unteren Zeile erfolgt. Es folgt ein Beispiel zur Nutzung der `tabular`-Umgebung.

```
\begin{tabular}{lcrp{33mm}}
\toprule
\textbf{Linksbündig} & \textbf{Zentriert} &
\textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline\midrule
a   & b   & c   & Dieser Text wird im Blocksatz gesetzt\tabularnewline
aa  & bb  & cc  & Auch Zeilenumbrüche sind vorhanden\tabularnewline
aaa & bbb & ccc & Worttrennungsmusterkontrolle\tabularnewline
\bottomrule
\end{tabular}
```

Linksbündig	**Zentriert**	**Rechtsbündig**	**Blocksatz**
a	b	c	Dieser Text wird im Blocksatz gesetzt
aa	bb	cc	Auch Zeilenumbrüche sind vorhanden
aaa	bbb	ccc	Worttrennungsmusterkontrolle

Die Tabellenbreite ergibt sich aus der Breite der einzelnen Spalten. Bei dieser Umgebung liegt es allein beim Anwender, auf die korrekte Breite der Tabelle zu achten, damit diese nicht über die Seitenränder hinausragt. Das kann auf Dauer recht aufwändig werden. Das Festlegen der Gesamtbreite einer Tabelle und das automatische Berechnen einiger oder aller Spaltenbreiten ist sicher die angenehmere Variante. Wie sich dies sehr komfortabel bewerkstelligen lässt, wird – ebenso wie die Lösung des Problems eines nicht umbrochenen Eintrags wie in der letzten Spalte der dritten Zeile der obigen Tabelle – im nachfolgenden Abschnitt demonstriert.

7.2 Tabellen mit variabler Spaltenbreite

Die Pakete **tabularx**, **tabulary** oder auch **tabu** stehen für das Setzen von Tabellen mit dynamisch berechneter Spaltenbreite bei fest vorgegebener Gesamtbreite zur Verfügung, welche nun kurz vorgestellt werden sollen. Die beiden erstgenannte Pakete **tabularx** sowie **tabulary** stammen vom gleichen Paketautor und haben sich als sehr stabil erwiesen. Das Paket **tabu** ist relativ neu und versucht, viele Funktionalitäten ganz unterschiedlicher Pakete für den Tabellensatz in sich zu vereinen und glänzt insbesondere durch seine Vielseitigkeit.

```
\usepackage{tabularx}
\usepackage{tabulary}
\usepackage{tabu}
\usepackage{longtable}
```

Jedes der Pakete stellt einen oder mehrere neue Spaltentypen zur Verfügung. Dies bedeutet jedoch nicht, dass in den dazugehörigen Tabellenumgebungen lediglich diese genutzt werden können. Vielmehr werden diese zusätzlich angeboten, was bedeutet, dass auch in den neuen Tabellenumgebungen weiterhin die gewohnten Spaltentypen `l`, `c`, `r` und `p` verfügbar sind.

7.2.1 Die Tabellenumgebung `tabularx`

Das Paket **tabularx** stellt den Spaltentyp `X` bereit, welcher prinzipiell dem Spaltentyp `p` entspricht. Auch für diesen wird eine `\parbox` verwendet, allerdings wird deren Breite *automatisch* berechnet. Die Umgebung `tabularx` erwartet vor der Spaltentypenangabe als obligatorisches Argument die gewünschte Breite der Tabelle. Spalten vom Typ `l`, `c` und `r` werden jedoch weiterhin *ohne* Zeilenumbruch gesetzt werden. Nur für Spalten vom Typ `X` und deren Derivate wird aus dem verbliebenen Platz die Breite berechnet. Für die `tabularx`-Umgebung wird das vorherige Beispiel wiederholt.

```
\begin{tabularx}{11.7cm}{lcrX}
\toprule
\textbf{Linksbündig} & \textbf{Zentriert} &
\textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline\midrule
a    & b    & c    & Dieser Text wird im Blocksatz gesetzt\tabularnewline
aa   & bb   & cc   & Auch Zeilenumbrüche sind vorhanden\tabularnewline
aaa  & bbb  & ccc  & Worttrennungsmusterkontrolle\tabularnewline
aaaa & bbbb & cccc & \hspace{0pt}Worttrennungsmusterkontrolle
\tabularnewline\bottomrule
\end{tabularx}
```

Linksbündig	**Zentriert**	**Rechtsbündig**	**Blocksatz**
a	b	c	Dieser Text wird im Blocksatz gesetzt
aa	bb	cc	Auch Zeilenumbrüche sind vorhanden
aaa	bbb	ccc	Worttrennungsmusterkontrolle
aaaa	bbbb	cccc	Worttrennungsmusterkontrolle

Die Breite der letzten Spalte wurde dabei aus der Angabe der Gesamtbreite mit `11.7cm` berechnet. Des Weiteren ist zu sehen, wie das Problem des Zeilenumbruchs behandelt werden kann. Normalerweise wird das erste Wort in einem Absatz von LaTeX *nie* umbrochen. Dies kann durch das Einfügen eines breitenlosen Leerzeichens mit `\hspace{0pt}` umgangen werden. Die gefundene Lösung ist allerdings alles andere als elegant.

Tabelle 4 Eine tabularx-Tabelle

Linksbündig	Zentriert	Rechtsbündig	Blocksatz
Ein linksbündiger Blindtext zur Demonstration einer S-Spalte	Ein zentrierter Blindtext zur Demonstration einer T-Spalte	Ein rechtsbündiger Blindtext zur Demonstration einer U-Spalte	Ein längerer Blindtext im Blocksatz zur besseren Demonstration einer Y-Spalte

Mit den Möglichkeiten des Paketes **array** ist das Problem relativ schnell gelöst. Es wird mit `\newcolumntype` ein neuer Spaltentyp definiert. Das erste Argument legt den Namen des Spaltentyps fest. Mit `>{<Definitionen>}{<Typ>}` wird im zweiten Argument das Ausführen von *<Definitionen>* vor dem Setzen des eigentlichen Inhaltes in einer `{<Typ>}`-Spalte definiert. Es wird ein neuer, auf der `X`-Spalte basierender Typ `Y` definiert, welcher zu Beginn der Spalte den Phantomabstand automatisch einfügt. Darauf basierend werden drei Spaltentypen für den links- und rechtsbündigen sowie zentrierten Textsatz mit einem möglichen Zeilenumbruch erstellt.

```
\newcolumntype{Y}{>{\hspace{0pt}}X}
\newcolumntype{D}{>{\raggedright}Y}
\newcolumntype{E}{>{\centering}Y}
\newcolumntype{F}{>{\raggedleft}Y}
```

Nachfolgend wird die Tabelle gesetzt mit den neuen Spaltentypen gesetzt.

```
\begin{table}
\begin{tabularx}{\textwidth}{@{}DEFY@{}}
\toprule
\textbf{Linksbündig} & \textbf{Zentriert} &
\textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline
\midrule
Ein linksbündiger Blindtext zur Demonstration einer S"~Spalte &
Ein zentrierter Blindtext zur Demonstration einer T"~Spalte &
Ein rechtsbündiger Blindtext zur Demonstration einer U"~Spalte &
Ein längerer Blindtext im Blocksatz zur besseren Demonstration
einer Y"~Spalte\tabularnewline
\bottomrule
\end{tabularx}
\caption{Eine \texttt{tabularx}"=Tabelle}\label{tab:tabularx}
\end{table}
```

Die Breite der einzelnen `X`-Spalten und deren Derivaten ist dabei identisch. Ein Gewichten der Spaltenbreiten untereinander ist prinzipiell möglich, allerdings mit einigen Einschränkungen verbunden. Genaueres hierzu ist der Dokumentation des Paketes **tabularx** zu entnehmen.

Tabelle 5 Eine `tabulary`-Tabelle

Linksbündig	Zentriert	Rechtsbündig	Blocksatz
Ein linksbündiger Blindtext zur Demonstration einer L-Spalte	Ein zentrierter Blindtext zur Demonstration einer C-Spalte	Ein rechtsbündiger Blindtext zur Demonstration einer R-Spalte	Ein wesentlich längerer und absolut inhaltsleerer Blindtext im Blocksatz für eine um einiges bessere Demonstration einer J-Spalte

7.2.2 Die Tabellenumgebung `tabulary`

Das Paket **tabulary** verfolgt im Vergleich zum zuvor beschrieben Paket **tabularx** einen etwas anderen Ansatz. Anstatt die verfügbare Breite über alle Spalten gleichmäßig zu verteilen, werden die Spaltenbreite anhand des jeweils darin befindlichen Inhaltes zueinander gewichtet. Dafür werden die vier Spaltentypen `L`, `C`, `R` sowie `J` zur Verfügung gestellt. Diese entsprechen prinzipiell dem Spaltentyp `p`, allerdings wird deren Breite *automatisch* berechnet, wobei Spalten mit mehr Inhalt breiter gesetzt werden. Für die automatische Berechnung der Spaltenbreiten muss verständlicherweise die gewünschte Gesamtbreite der Tabelle angegeben werden, was bei der `tabulary`-Umgebung mit dem ersten obligatorischen Argument vor der Angabe der Spaltendefinition erfolgt. Werden eine oder mehrere Spalten vom Typ `l`, `c` und `r` angegeben, so werden diese weiterhin in ihrer natürlichen Breite und ohne Zeilenumbruch gesetzt.

Die Gewichtung der Spalten vom Typ `LCRJ` und deren Derivate wird aus dem verbliebenen Platz die Breite berechnet. Die Berechnung der Spaltenbreite kann durch den Anwender außerdem über zwei Längen beeinflusst werden. Mit `\tymin` wird die Mindestgröße für Spalten festgelegt. Sollten Sie wissen, dass eine Spalte sehr schmal wird, können Sie allerdings auch einfach eine der Standardspalten `lcrp` nutzen. Mit der Länge `\tymax` kann die maximale Ausdehnung einer der Spalten vom Typ `LCRJ` festgelegt werden.

Für die `tabulary`-Umgebung wird das vorherige Beispiel wiederholt. Zu beachten ist, dass die Spaltentypen `LCRJ` mit dem Makro `\tyformat` bereits ein breitenloses Leerzeichen einsetzen, womit sich das manuelle Einfügen von selbigem durch den Anwender erübrigt.

```
\begin{table}
\begin{tabulary}{\textwidth}{@{}LCRJ@{}}
\toprule
\textbf{Linksbündig} & \textbf{Zentriert} &
\textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline\midrule
Ein linksbündiger Blindtext zur Demonstration einer L"~Spalte &
Ein zentrierter Blindtext zur Demonstration einer C"~Spalte &
Ein rechtsbündiger Blindtext zur Demonstration einer R"~Spalte &
Ein wesentlich längerer und absolut inhaltsleerer Blindtext im
Blocksatz für eine um einiges bessere Demonstration einer J"~Spalte
\tabularnewline\bottomrule
\end{tabulary}
\caption{Eine \texttt{tabulary}"=Tabelle}\label{tab:tabulary}
\end{table}
```

7.2.3 Die Tabellenumgebung `tabu`

Das Paket **tabu** bietet eine mächtige und komfortable Alternative zu **tabularx**. Es kam in diesem Tutorial bereits für die Verzeichnisse von Abkürzungen und Symbolen in Abschnitt 4.5 zum Einsatz. Leider ist das Paket in er aktuellen Version v2.8 mit Vorsicht zu genießen. Zum einen wären für das Paket in der aktuellen Version seit geraumer Zeit ein paar kleinere Bugfixes notwendig, mehr dazu sehen Sie später. Außerdem wird sich die Benutzerschnittstelle in einer zukünftigen Version[11] sehr stark ändern. Sie sollten sich bewusst sein, dass mit der Version v2.8 gesetzte Dokumente gegebenenfalls später angepasst werden müssen.

Nichtsdestotrotz soll hier folgend die individuelle Verwendung einer `tabu`-Umgebung gezeigt werden, insbesondere weil für das Setzen umbruchfähiger Tabellen mit automatisch berechneten Spaltenbreiten momentan (fast) keine Alternative zu der Umgebung `longtabu` existiert.

Dieses Paket definiert ebenso einen Spaltentyp `X`, welchem allerdings zusätzlich ein optionales Argument angehangen werden kann. Mit diesem lässt sich sowohl die Gewichtung der automatisch berechneten Spalten untereinander als auch die Positionierung zur Grundlinie sowie die Ausrichtung des Inhaltes ändern.

```
\begin{tabu} to 11.7cm {lcrX}
\toprule
\textbf{Linksbündig} & \textbf{Zentriert} &
\textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline\midrule
a    & b    & c    & Dieser Text wird im Blocksatz gesetzt\tabularnewline
aa   & bb   & cc   & Auch Zeilenumbrüche sind vorhanden\tabularnewline
aaaa & bbbb & cccc & Worttrennungsmus\-terkontrolle\tabularnewline
aaa  & bbb  & ccc  & Worttrennungsmusterkontrolle
\tabularnewline\bottomrule
\end{tabu}
```

Linksbündig	**Zentriert**	**Rechtsbündig**	**Blocksatz**
a	b	c	Dieser Text wird im Blocksatz gesetzt
aa	bb	cc	Auch Zeilenumbrüche sind vorhanden
aaaa	bbbb	cccc	Worttrennungsmusterkontrolle
aaa	bbb	ccc	Worttrennungsmusterkontrolle

Das Beispiel zeigt auch gleich das momentan meiner Meinung nach größte Problem des Paketes: Bei mehrzeiligen Zellen fehlt vertikale Zwischenraum zur nächsten Tabellenzeile. Nun besteht entweder die Möglichkeit, bei jeder Angabe einer Spalte im Tabellenkopf, welche eine `\parbox` verwendet, mit dem Ausdruck `>{\strut}` die fehlenden Unterlänge auszugleichen oder es wird abermals ein entsprechender Spaltentyp definiert. Das ist für die `X`-Spalten der Umgebung `tabu` etwas schwieriger, da `\newcolumntype` normalerweise die Definition eines optionalen Argumentes nicht vorgesehen ist. Mit ein wenig Trickserei ist das dennoch möglich:

[11] https://groups.google.com/d/topic/comp.text.tex/xRGJTC74uCI

Tabelle 6 Eine tabu-Tabelle in Verbindung mit dem Befehl \ttabbox, welcher vom Paket floatrow für Beschriftungen in Objektbreite bereitgestellt wird

Linksbündig	Zentriert	Rechtsbündig	Blocksatz
Ein linksbündiger Blindtext zur Demonstration einer Z[l]-Spalte	Ein zentrierter Blindtext zur Demonstration einer Z[c]-Spalte	Ein rechtsbündiger Blindtext zur Demonstration einer Z[r]-Spalte	Ein Blindtext im Blocksatz innerhalb einer Z-Spalte

```
\makeatletter
\newcolumntype{Z}{}
\renewcommand*{\NC@rewrite@Z}[1][]{%
  \NC@find>{\hspace{0pt}}X[#1]<{\@finalstrut\@arstrutbox}%
}
\makeatother
```

Anschließend kann der Z-Spaltentyp äquivalent zu X genutzt werden. Zu beachten ist dabei insbesondere die Möglichkeit, die Spalten in ihrer resultierenden Breite sehr einfach zueinander gewichten zu können – im Gegensatz zur Umgebung tabularx, wo dieses Unterfangen etwas schwieriger ist. Das Ergebnis ist in Tabelle 6 zu sehen. Zu beachten ist die Verwendung von \ttabbox aus dem Paket **floatrow**, um die Beschriftung an die Breite der Tabelle anzupassen. Genaueres dazu in Abschnitt 6.1.

```
\begin{table}
\ttabbox{%
  \begin{tabu} to .8\textwidth {@{}Z[3,l]Z[3,c]Z[3,r]Z[2,j]@{}}
    \toprule
    \textbf{Linksbündig} & \textbf{Zentriert} &
    \textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline\midrule
    Ein linksbündiger Blindtext zur Demonstration einer Z[l]"~Spalte &
    Ein zentrierter Blindtext zur Demonstration einer Z[c]"~Spalte &
    Ein rechtsbündiger Blindtext zur Demonstration einer Z[r]"~Spalte &
    Ein Blindtext im Blocksatz innerhalb einer Z"~Spalte
    \tabularnewline\bottomrule
  \end{tabu}%
}{%
  \caption[Eine \texttt{tabu}"=Tabelle]{%
    Eine \texttt{tabu}"=Tabelle in Verbindung mit dem Befehl
    \texttt{\textbackslash ttabbox}, welcher vom Paket \texttt{floatrow}
    für Beschriftungen in Objektbreite bereitgestellt wird%
  }%
  \label{tab:tabu}%
}
\end{table}
```

8 Querverweise

Damit alle möglichen Querverweise in einem PDF-Dokument automatisch verlinkt werden, sollte das Paket **hyperref** geladen werden. Um die erzeugten Links verträglich aussehen zu lassen, werden die beiden Paketoptionen `colorlinks` sowie `linkcolor=blue` verwendet. Da **hyperref** allerhand Veränderungen an vielen Standardbefehlen vornimmt, sollte dieses als letztes in der Präambel eingebunden werden. Nur Pakete, bei denen in der Dokumentation explizit darauf hingewiesen wird, dass diese nach **hyperref** zu laden sind, sollten auch danach folgen. Eines dieser wenigen Pakete ist das in diesem Tutorial verwendete **glossaries**.

```
\usepackage[colorlinks,linkcolor=blue]{hyperref}
```

Um auf ein bestimmtes Objekt wie beispielsweise eine Tabelle, eine Abbildung oder einen bestimmten Anschnitt im Dokument referenzieren zu können, muss mit `\label{<label>}` ein Anker für selbiges an der entsprechenden Stelle gesetzt werden. Bei Überschriften sollte dies direkt nach der Nutzung des Gliederungsbefehls[12] erfolgen. Für Abbildungen, Tabellen oder andere Objekte in Gleitumgebungen muss der Anker unmittelbar nach der Verwendung von `\caption` respektive `\captionof` erstellt werden. Um nun auf diesen Anker zu referenzieren, sollte nicht `\ref` sondern `\autoref` im Fließtext genutzt werden. Letztgenannter Befehl erzeugt nicht nur einen Link mit der dazugehörigen Nummerierung sondern stellt dieser die Bezeichnung des referenzierten Objektes[13] voran.

Mit dem Befehl `\pageref` lässt sich auf die Seite eines Ankers verweisen. Das Paket **varioref** kann dabei unterscheiden, ob der Anker sich auf der aktuellen Seite, auf einer Seite direkt davor beziehungsweise danach oder auf einer entfernteren Seite befindet und dementsprechend einen Hinweis darauf ausgeben. Hierfür werden die Befehle `\vref` sowie `\vpageref` zur Verfügung gestellt. Das Paket **cleveref** vereint die Vorzüge von `\autoref` in Verbindung mit `\vref` durch den Befehl `\cref`.

9 Zitate

Für das wörtliche Zitieren kann das Paket **csquotes** verwendet werden. Dieses stellt den Befehl `\enquote` bereit, der die kennzeichnenden Anführungszeichen automatisch in Abhängigkeit der verwendeten Dokumentsprache setzt.

```
\usepackage{csquotes}
```

Da für die unterschiedlichen Fachdisziplinen teils sehr unterschiedliche Zitierrichtlinien existieren, kann hier keine allgemeingültige Empfehlung zu Stilfragen gegeben werden. Zu beachten ist auf jeden Fall, dass zu wörtlichen Zitaten eine Quellenangabe erfolgen *muss*. Dabei sollte darauf geachtet werden, dass nicht nur die Quelle selbst sondern zumindest auch die Seitenzahl des Zitats angegeben wird.

```
\enquote{Dies ist ein zugegebenermaßen nicht sehr sinnvolles Zitat.}
\cite[58]{hanisch14}
```

[12] `\chapter`, `\section` etc.
[13] „Abbildung“, „Tabelle“, ...

„Dies ist ein zugegebenermaßen nicht sehr sinnvolles Zitat.“ [Han14, S. 58]

Längere Zitate sollten abgesetzt vom Fließtext ausgegeben werden. Hierfür bieten sowohl die Standard- als auch die KOMA-Script-Klassen standardmäßig die Umgebungen `quote` und `quotation` an. Allerdings verwenden beide fest eingestellte Absatzauszeichnungen. Sollen jedoch die mit der Option `parskip` vorgegebenen Einstellungen für Absätze beachtet werden, so bietet sich das Paket **quoting** an, welches die gleichnamige Umgebung `quoting` definiert und die Absätze wie im restlichen Dokument auszeichnet.

```
\usepackage{quoting}

\begin{quoting}
\enquote{%
  Dies ist ein noch sinnloseres Zitat. Allerdings wird zumindest die
  Wirkung der Umgebung \texttt{quoting} bei der Absatzauszeichnung
  deutlich.

  Wie zu sehen ist, wird der zweite Absatz~--- wie jeder weitere~---
  aufgrund der Option \texttt{parskip=false} eingezogen.
}
\cite[sinngemäß nach][\pno{} 12, zweiter Absatz]{hanisch14}
\end{quoting}
```

„Dies ist ein noch sinnloseres Zitat. Allerdings wird zumindest die Wirkung der Umgebung `quoting` bei der Absatzauszeichnung deutlich.

Wie zu sehen ist, wird der zweite Absatz – wie jeder weitere – aufgrund der Option `parskip=false` eingezogen. “ [sinngemäß nach Han14, S. 12, zweiter Absatz]

Literatur

[Han14] Falk Hanisch. *Ein LaTeX-Bundle für Dokumente im neuen Corporate Design der Technischen Universität Dresden. Benutzerhandbuch.* Dresden, 2014.

10 Worttrennungen

Bei der Verwendung von LaTeX 2ε gibt es ab und an das Problem, dass bestimmte Wörter am Zeilenende falsch getrennt werden. Dies hat historische Gründe. Als TeX in den 1980er-Jahren entwickelt wurde, war es aufgrund der zu dieser Zeit zur Verfügung stehenden Speichergrößen schier unmöglich, die Worttrennungen mit einem Wörterbuchansatz umzusetzen. Stattdessen wurde – um den benötigten Speicherbedarf möglichst gering zu halten – ein Algorithmus entworfen, der mögliche Trennstellen in einem Wort mit einer Mustererkennung ermittelt. Die verwendeten Muster basieren jedoch auf der englischen Sprache, weshalb in nicht-englischsprachigen Texten auch heute noch teilweise falsche Trennstellen gefunden werden.

In Unterkapitel 1 wurde bereits auf das Paket **hyphsubst** hingewiesen, welches dieses Manko – aufgrund der nicht mehr so stark gegebenen Restriktionen bezüglich des Speicherbedarfes – mit einem Wörterbuch behebt. Dieses Paket muss für **pdfLaTeX** vom Anwender eingebunden

werden, LuaLaTeX und XeLaTeX verwenden die Trennmuster – ein geladenes Sprachpaket wie **polyglossia** oder **babel** vorausgesetzt – automatisch. Damit auch Wörter mit Umlauten richtig getrennt werden, ist zudem das Paket **fontenc** mit der `T1`-Schriftkodierung respektive **fontspec** notwendig. Der Beginn einer Dokumentpräambel könnte folgendermaßen aussehen:

```
\documentclass[ngerman,<Klassenoptionen>]{<Dokumentklasse>}
\ifpdftex{
  \usepackage[T1]{fontenc}
  \usepackage[ngerman=ngerman-x-latest]{hyphsubst}
}{
  \usepackage{fontspec}
}
\usepackage{babel}
```

Die Option `ngerman` lädt dabei die Trennmuster für die neue deutsche Rechtschreibung, wobei der Wert `ngerman-x-latest` die neuesten aller verfügbaren Trennmuster lädt. Für die alte Orthografie ist stattdessen die Option `german` mit dem Wert `german-x-lastest` zu verwenden.

10.1 Einmalige und globale Worttrennungskorrektur

Sollte es dennoch passieren, dass ein Wort nicht korrekt getrennt wird, lässt sich dies manuell korrigieren. Dafür stellt das Paket **babel** mehrere Kurzbefehle zur Verfügung. Mit dem Makro `\hyphenation` lassen sich für ein bestimmtes Wort außerdem alle potenziellen Trennstellen angeben. Zum Thema Worttrennungen in deutschsprachigen Texten ist im TUD-Script-Handbuch unter dem Kapitel Tipps & Tricks mehr zu finden. Dort wird auch erläutert, wie das Problem der alleinigen Trennung von Wörtern mit Bindestrich an selbigem umgangen werden kann.

10.2 Worttrennung im Flattersatz

Normalerweise ist bei LaTeX 2ε die Worttrennung für den Flattersatz[14] deaktiviert. Mit dem Paket **ragged2e** kann diese aktiviert werden. Nach dem Laden stehen die Befehle `\Centering`, `\RaggedLeft` sowie `\RaggedRight` für den Anwender zur Verfügung, um Worttrennungen im Flattersatz zu verwenden. Die Option `newcommands` überschreibt die originalen Befehle `\centering`, `\raggedleft` und `\raggedright` mit den korrelierten Befehlen aus dem Paket. Genaueres dazu ist der Dokumentation des Paketes zu entnehmen.

11 Mikrotypografische Feinheiten

Unter Mikrotypografie oder Detailtypografie wird die Gestaltung folgender Feinheiten des Schriftsatzes verstanden:

- korrekte Anwendung der Satzzeichen
- Schriftart, Kapitälchen und Ligaturen
- Wortabstände sowie Laufweite (Buchstaben- und Zeichenabstände)
- optischer Randausgleich

[14] `\raggedright`, `\raggedleft`, `\centering`

Die letzten beiden Punkte werden sensationell gut durch das Paket **microtype** bearbeitet. Dieses muss lediglich in der Präambel geladen werden, wobei es nicht mit der klassischen TEX-Engine, wohl jedoch mit **pdfTEX** als auch **LuaTEX** sowie **XETEX** funktioniert.

```
\usepackage[babel]{microtype}
```

Die Option `babel` ist nur notwendig, wenn einzelne Textpassagen oder das gesamte Dokument in Englisch oder Französisch verfasst werden und hierfür die genutzte Sprache mit den entsprechenden Möglichkeiten von **babel** eingestellt wird.

Das korrekte Anwenden von Satzzeichen wird für deutschsprachige Texte bereits in mehreren Artikeln [Neu97a; Neu97b; Str07; Bie09] sehr gut erläutert, weshalb ein Hinweis auf diese Arbeiten für weitergehende Informationen zu dieser Thematik absolut ausreichend ist. Nachfolgend werden Tipps gegeben sowie einige LATEX 2_ε-Pakete vorgestellt, die bei der Umsetzung der in den genannten Artikeln erläuterten Empfehlungen helfen.

11.1 Abkürzungen

In [Str07] wird gleich zu Beginn darauf hingewiesen, wie häufig verwendete Abkürzungen sehr einfach im Dokument verwendet werden können und dabei typografisch richtig gesetzt werden. Beispielsweise gelingt dies für die Abkürzung „zum Beispiel (z. B.)“ mit dieser Definition:

```
\newcommand{\zB}{\mbox{z.\,B.}\xspace}
```

Wenn ohnehin das Paket **glossaries** für Akronyme verwendet wird, kann alternativ dazu auch dessen Funktionalität genutzt werden. Nach dem Einbinden muss in der Präambel lediglich Folgendes eingefügt werden:

```
\newignoredglossary{abbreviation}
\newcommand*{\newabbreviation}[4][]{%
  \newacronym[type=abbreviation,#1]{#2}{\mbox{#3\xspace}}{#4}%
}%
```

Mit `\newignoredglossary` wird dabei ein Glossar erstellt, welche bei der Ausgabe aller Glossare mit `\printglossaries` ignoriert wird. Im Dokument kann die gewünschte Abkürzung mit

```
\newabbreviation{zB}{z.\,B.}{zum Beispiel}
```

definiert und über die gewohnten Befehlen aus **glossaries** verwendet werden.

```
Die Ausgabe der soeben definierten Abkürzung \enquote{\acrshort{zB}}
erfolgt mit dem Befehl \texttt{\textbackslash acrshort}. Wurde das
Paket \texttt{glossaries} mit der Option \texttt{shortcut} geladen,
kann auch das Makro \texttt{\textbackslash acs} genutzt werden.
```

Die Ausgabe der soeben definierten Abkürzung „z. B.“ erfolgt mit dem Befehl `\acrshort`. Wurde das Paket `glossaries` mit der Option `shortcut` geladen, kann auch das Makro `\acs` genutzt werden.

11.2 Listen

Für Auflistungen aller Art bietet LaTeX die drei Standardumgebungen `itemize`, `enumerate` und `description`. Das häufigste Anliegen bei der Verwendung dieser Umgebungen ist das Reduzieren der Abstände zwischen den einzelnen, mit `\item` gesetzten Punkten, welche sehr häufig als zu groß empfunden werden. Eine Aufzählung erscheint ohne Anpassungen wie folgt:

```
\begin{itemize}
\item erster Punkt
\item zweiter Punkt
\item dritter Punkt
\end{itemize}
```

Die resultierende Ausgabe: Aufzählung mit vertikalen Standardabständen

- erster Punkt
- zweiter Punkt
- dritter Punkt

Dabei wirken die Abstände zwischen den Punkten tatsächlich etwas überdimensioniert. Mit **enumitem** können alle Listenumgebungen an die individuellen Bedürfnisse angepasst werden.

```
\usepackage{enumitem}
```

Um diese ungewollten Abstände zwischen den Listenpunkten zu entfernen, kann der Schlüssel `noitemsep` verwendet werden. Mit dem Befehl `\setlist` werden – ohne zusätzliche Angaben im optionalen Argument – alle Umgebungen für Aufzählungen global geändert. Hier soll dies lediglich für die Umgebung `itemize` geschehen:

```
\setlist[itemize]{noitemsep}

\begin{itemize}
\item erster Punkt
\item zweiter Punkt
\item dritter Punkt
\end{itemize}
```

Die resultierende Ausgabe: Aufzählung ohne vertikale Abstände

- erster Punkt
- zweiter Punkt
- dritter Punkt

Alternativ zur globalen Änderung einer speziellen oder aller Aufzählungslisten kann der Schlüssel `noitemsep` auch nur gezielt an einzelne Umgebung über das optionale Argument weitergereicht werden (`\begin{itemize}[noitemsep]`).

Natürlich erlaubt das Paket noch weitergehende Einstellungen für die Umgebungen `itemize`, `enumerate` und `description`. Zusätzlich können diese für eigene Aufzählungsumgebungen geklont und den eigenen Bedürfnissen angepasst werden. Genaueres hierzu ist der Dokumentation des Paketes **enumitem** entnehmen.

11.3 Ligaturen

Die meisten LaTeX-Schriftfamilien enthalten Ligaturen, die für einen typografisch sauberen Satz bei zusammengesetzten Wörtern, wie sie gerade im Deutschen häufig vorkommen, aufgetrennt werden müssten. Mit der Anwendung ***DeLig*** kann dies automatisiert geschehen. Wird **LuaLaTeX** als Textsatzsystem genutzt, kann alternativ das Paket **selnolig** verwendet werden.

11.4 Auslassungspunkte

Das typografisch richtige Setzen von Auslassungspunkten mit den Befehlen `\dots` respektive `\textellipsis` wird im Handbuch unter Das Setzen von Auslassungspunkten genauer beschrieben, weshalb hier nur kurz darauf eingegangen wird. Es wird das Paket **ellipsis** benötigt.

```
\usepackage{ellipsis}
```

Für deutschsprachige Texte sollten außerdem noch folgendes ausgeführt werden:

```
\let\ellipsispunctuation\relax
```

11.5 Mathematiksatz

Von LaTeX 2$_\varepsilon$ werden einige Umgebungen für das Setzen mathematischer Formeln und Ausdrücke bereitgestellt, welche meist jedoch nicht *alle* Anforderungen eines Nutzers an den Mathematiksatz erfüllen können. Das Paket **amsmath** bietet deshalb eine Reihe weiterführender Umgebungen für das Setzen von einzeiligen und mehrzeiligen sowie ausgerichteter Formeln zur Verfügung. Auf eine Einführung in das Paket wird – aufgrund der bereits in großer Zahl vorhandenen Einführungen – in diesem Tutorial verzichtet. Außerdem stehen mit **mathtype.pdf** und **mathswap.pdf** zwei weitere Tutorials zur Verfügung, die sich explizit mit der Thematik eines guten mathematischen Satz beschäftigen.

Das Paket **amsmath** stellt für LaTeX 2$_\varepsilon$ den De-facto-Standard für den mathematischen Satz dar. Allerdings wird es – insbesondere aus diesem Grund – sehr „vorsichtig“ weiterentwickelt. Für das Setzen neuer Dokumente sollte deshalb das Paket **mathtools** verwendet werden, das auf **amsmath** basiert und dieses um zusätzliche Funktionalitäten ergänzt sowie Bugfixes bereithält.

11.6 Darstellung von mathematischen Brüchen

Für ansehnliche und insbesondere im Fließtext gut lesbare mathematische Brüche stellt das Paket **xfrac** den Befehl `\sfrac` zur Verfügung.

```
\usepackage{xfrac}
```

11.7 Einheiten

Mit dem Paket **siunitx** lassen sich sowohl physikalische Einheiten als auch Zahlen typografisch korrekt setzen. Sollen die Funktionalitäten in einem deutschsprachigen Dokument problemlos nutzbar sein, so sind die Hinweise zum Setzen von Einheiten mit **siunitx** im TUD-Script-Handbuch zu beachten. Das Paket **units** ist nicht ganz so mächtig wie **siunitx**, kann allerdings als pragmatische Alternative gesehen und genutzt werden.

11.8 Datumsformatierung

Das Paket **isodate** stellt `\printdate` bereit. Dieser Befehl kann für die Formatierung von Datumsangaben verwendet werden. Wird das Paket mit einer der Dokumentklassen aus dem TUD-Script-Bundle geladen, wird der genannte Befehl automatisch für alle bereitgestellten Datumsfelder verwendet. Alternativ dazu kann das Paket **datetime2** genutzt werden, dessen Befehl `\DTMDate` wird ebenfalls unterstützt.

```
\usepackage{isodate}
```

12 Erstellen von Abbildungen

Die einfachste Möglichkeit, in einer wissenschaftlichen Abhandlung Grafiken zu verwenden, ist sicherlich das Einbinden von externen Abbildungen mit dem Befehl `\includegraphics` aus dem Paket **graphicx**. Allerdings ist für eine wissenschaftliche Arbeit in höchster Qualität meiner Meinung nach das bloße Einfügen einer gescannten oder heruntergeladenen Grafik unzureichend. Vielmehr sollte diese zum einen als skalierbare Vektorgrafik vorliegen, um sowohl die höchste Druckqualität als auch die Wiederverwendbarkeit sicherzustellen, und zum anderen in ihrer Gestaltung dem verwendeten Layout des verfassten Dokumentes entsprechen. Dies betrifft sowohl die eingesetzten Farben und Schriften als auch die Stärke der verwendeten Linien.

Zum Zeichnen von Grafiken sind die freien Programme ***LaTeXDraw*** und besonders ***Inkscape*** empfehlenswert, da diese die Verwendung der Dokumentschriften innerhalb der erstellten Grafiken ermöglichen. Im Handbuch zu TUD-Script wird dies für ***Inkscape*** genauer erläutert. Eine andere Variante ist das „Programmieren“ von Grafiken. Die zwei bekanntesten LaTeX-Pakete sind **tikz** und **pstricks**. Mit beiden wird folgend eine Abbildung exemplarisch erstellt, ohne dabei ins Detail zu gehen, da es – neben den umfangreichen Dokumentationen – zahlreiche Beispiele und Erläuterungen im Internet gibt.

12.1 Das Paket tikz

Zur Verwendung von **tikz** ist nicht viel zu sagen. Einfach in der Dokumentpräambel einbinden und es kann losgehen. Für das nachfolgende Beispiel sind außerdem noch zwei weitere Programmbibliotheken für **tikz** notwendig.

```
\usepackage{tikz}
\usetikzlibrary{chains}
\usetikzlibrary{decorations.markings}
\tikzset{on grid}
```

Ein kurze Anmerkungen zur Skalierung der Grafiken möchte ich mir erlauben. Normalerweise verwendet **tikz** Koordinaten im Zentimeter-Raster. Ich persönlich bevorzuge der relativierte Skalierung auf die Textbreite des Dokumentes, wobei eine Einheit genau 1 % dieser entspricht. Das kann mit folgendem Quelltext erreicht werden:

```
\newlength{\tikzunit}
\setlength{\tikzunit}{.01\textwidth}
\tikzset{x=\tikzunit,y=\tikzunit}
```

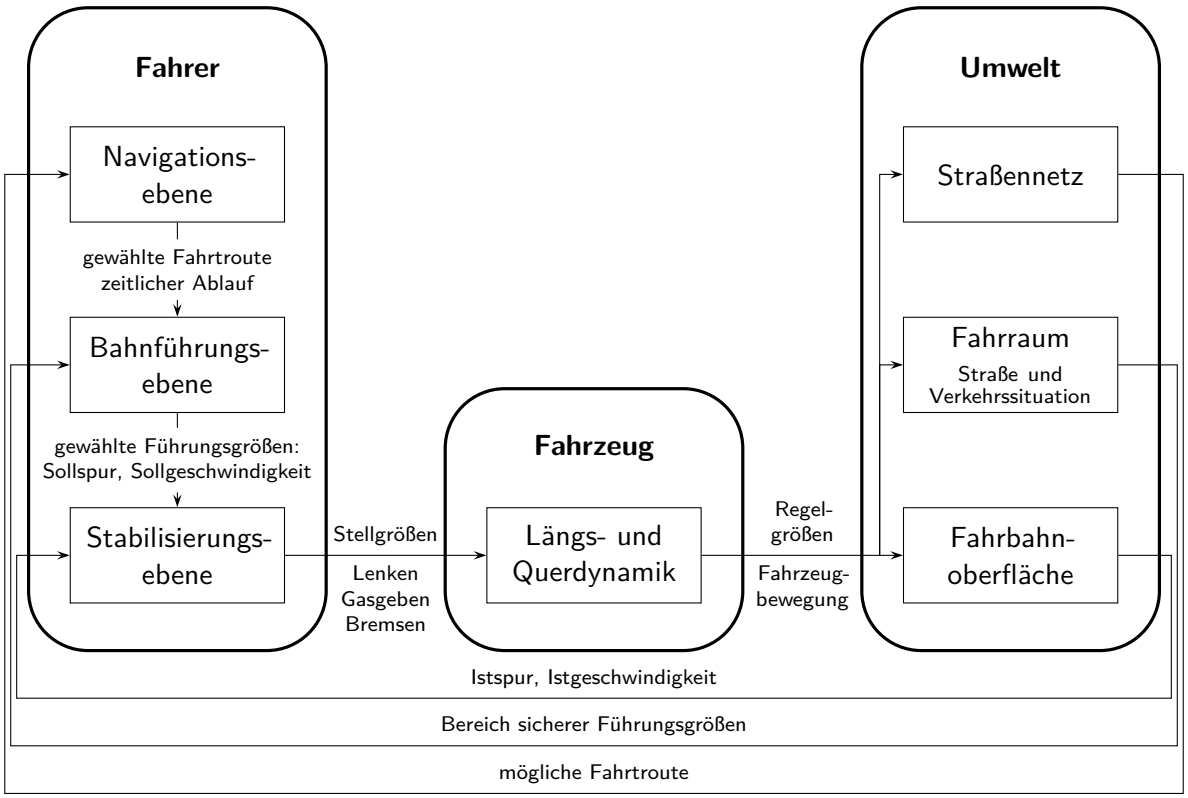


Abbildung 9 Eine mit TikZ erstellte Grafik

Nachfolgend wird eine Grafik innerhalb einer Gleitumgebung exemplarisch erstellt. Das Ergebnis ist in Abbildung 9 zu sehen.

```
\begin{figure}
\begin{tikzpicture}
  \tikzstyle{inner box}=[%
    text width=17\tikzunit,
    align=center,
    rectangle,
    inner sep=.5\tikzunit,
    minimum height=8\tikzunit,
    font=\hspace{0pt},
    draw
  ]
  \tikzstyle{inner label}=[align=center, font=\scriptsize]
  \tikzstyle{inner box chain}=[every node/.style={on chain}]
  \tikzstyle{inner box chain below}=[%
    inner box chain, node distance=8\tikzunit,continue chain=going below
  ]
  \tikzstyle{inner box chain right}=[%
    inner box chain,node distance=35\tikzunit,continue chain=going right
  ]
```

```
\tikzstyle{inner box chain above}=[%
  inner box chain,node distance=16\tikzunit,continue chain=going above
]
\tikzstyle{pstarrow->}=[%
  decoration={markings,
    mark=at position 1 with {\arrow[xscale=1.5]{stealth}};
  },
  postaction={decorate},
  shorten >=0.7pt
]
\newcommand{\tikzparbox}[2][9]{%
  \parbox{#1\tikzunit}{\centering\hspace{0pt}#2}%
}
\begin{scope}[start chain]
  \begin{scope}[inner box chain below]
    \node(NE)[inner box]{Navigations\-ebene};
    \node(NB)[inner label]{gewählte Fahrtroute\\ zeitlicher Ablauf};
    \node(BE)[inner box]{{Bahnführungs\-ebene}};
    \node(BS)[inner label]{%
      gewählte Führungsgrößen:\\ Sollspur, Sollgeschwindigkeit%
    };
    \node(SE)[inner box]{Stabilisierungs\-ebene};
  \end{scope}
  \begin{scope}[inner box chain right]
    \node(LQ)[inner box]{Längs- und Querdynamik};
    \node(FO)[inner box]{Fahrbahn\-oberfläche};
  \end{scope}
  \begin{scope}[inner box chain above]
    \node(FR)[inner box]{Fahrraum\\ \smallskip{\scriptsize Straße
    und\\ \vspace{-1.5ex}Verkehrssituation}};
    \node(SN)[inner box]{Straßennetz};
  \end{scope}
\end{scope}
\begin{scope}[inner label,minimum size=0pt]
  \draw [pstarrow->] (FO) -| ++(13.5,-12) to
    node [above]{Istspur, Istgeschwindigkeit} ++(-97,0) |- (SE);
  \draw [pstarrow->] (FR) -| ++(14  ,-32) to
    node [above]{Bereich sicherer Führungsgrößen} ++(-98,0) |- (BE);
  \draw [pstarrow->] (SN) -| ++(14.5  ,-52) to
    node [above]{mögliche Fahrtroute} ++(-99,0) |- (NE);
\end{scope}
\begin{scope}[inner label]
  \draw               (NE) to (NB);
  \draw [pstarrow->] (NB) to (BE);
  \draw               (BE) to (BS);
  \draw [pstarrow->] (BS) to (SE);
  \draw [pstarrow->] (SE) to
```

```
      node[above] {\tikzparbox{Stell\-größen}}
      node[below] {\tikzparbox{Lenken Gasgeben Bremsen}}
    (LQ);
    \draw [pstarrow->] (LQ) to
      node[above]{\tikzparbox{Regel\-größen}}
      node[below]{\tikzparbox{Fahrzeugbewegung}}
    (FO);
    \draw [pstarrow->] (LQ)+(24,0) |- (FR);
    \draw [pstarrow->] (LQ)+(24,0) |- (SN);
  \end{scope}
  \begin{scope}[very thick,rounded corners=5\tikzunit]
    \draw (-12.5,-40) rectangle (12.5,14);
    \draw ( 22.5,-40) rectangle (47.5,-18);
    \draw ( 57.5,-40) rectangle (82.5,14);
  \end{scope}
  \begin{scope}[font=\bfseries]
    \node at (0,9) {Fahrer};
    \node at (35,-23) {Fahrzeug};
    \node at (70,9) {Umwelt};
  \end{scope}
\end{tikzpicture}
\caption{Eine mit TikZ erstellte Grafik}\label{fig:tikz}
\end{figure}
```

12.2 Das Paket pstricks

Die zuvor mit **tikz** erstellte Grafik wird nachfolgend mit dem Paket **pstricks** erstellt. Mit diesem lassen sich PostScript-Grafiken erstellen. Ähnlich wie bei **tikz** gibt es ein Kernpaket und zusätzliche Ergänzungen. Diese werden jedoch nicht als Programmbibliotheken sondern ganz normal als LaTeX-Pakete eingebunden.

```
\usepackage{pstricks,pst-node}
```

Die Verwendung von PostScript hat zahlreiche Vorteile, macht die direkte Nutzung von **pdfLaTeX** etwas schwierig. Das Paket **auto-pst-pdf** schafft hier jedoch Abhilfe. Wird das Textsatzsystem **pdfLaTeX** über die zusätzliche Option `--shell-escape` beziehungsweise `--enable-write18` mit erweiterten Schreibrechten aufgerufen, so werden die Abbildungen automatisch im PDF-Format erzeugt und eingebunden.

```
\usepackage{auto-pst-pdf}
```

Bei einigen Pakete kommt es in der Kombination mit **auto-pst-pdf** zu kleineren Problemen, die sich jedoch meist relativ leicht beheben lassen. Im Handbuch wird die gleichzeitige Verwendung von **auto-pst-pdf** in Kombination mit den Paketen **tikz**, **todonotes** sowie **floatrow** – bei der Nutzung des Befehls `\ffigbox` – erklärt. Diese Hinweise sollten Sie gegebenenfalls beachten.

Ähnlich wie zuvor bei der **tikz**-Grafik wird die nachfolgende **pstricks**-Abbildung auf die Textbreite des Dokumentes skaliert, wobei eine Einheit genau 1 % dieser entspricht. Außerdem werden noch ein paar weitere Einstellungen gemacht. Dies geschieht mit:

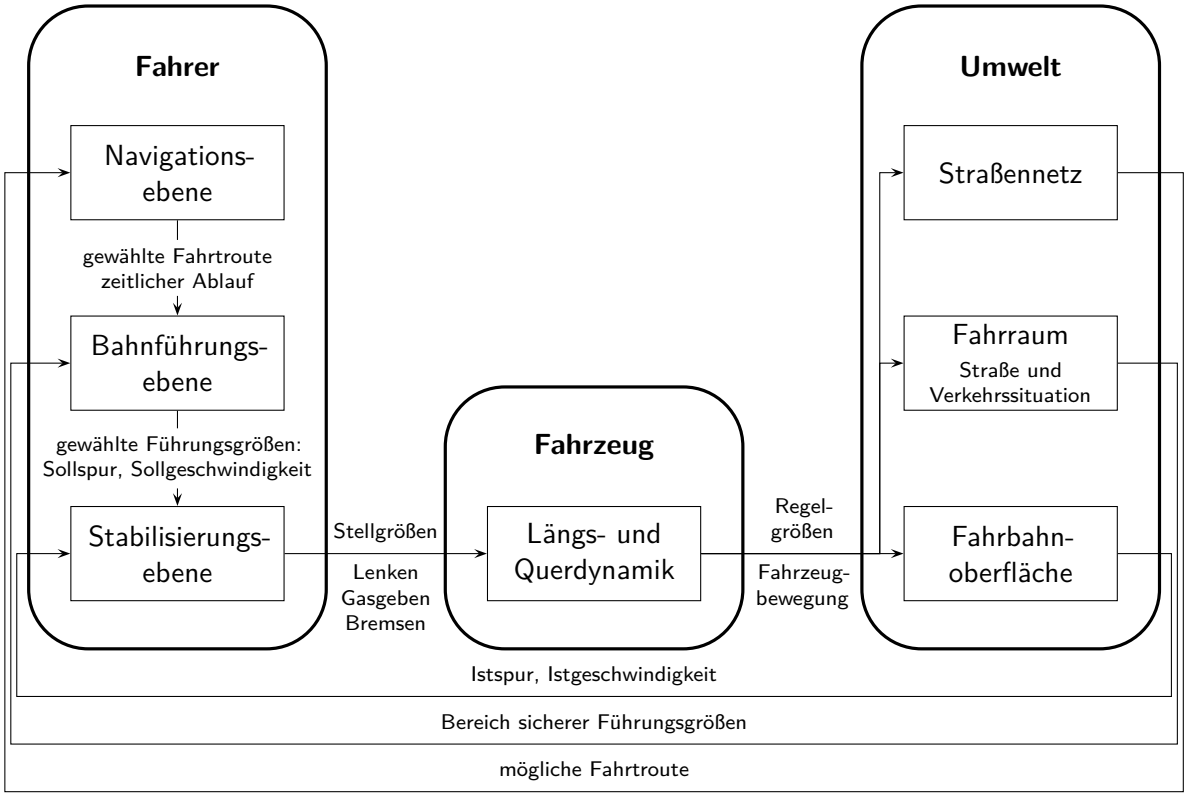


Abbildung 10 Eine mit pstricks erstellte Grafik

```
\psset{%
  unit=.01\textwidth,%
  cornersize=absolute,%
  labelsep=.8ex,%
  linewidth=.4pt,%
  arrowscale=1.5,%
}
```

Die innerhalb einer Gleitumgebung erzeugte **pstricks**-Grafik ist in Abbildung 10 dargestellt.

```
\begin{figure}
\begin{pspicture}(0,-2)(100,64)
\newcommand{\fnodetext}{}
\def\fnodetext(#1)#2#3{%
  \fnode[framesize=18 8](#1){#2}%
  \rput(#1){\parbox{17\psunit}{\centering\hspace{0pt}#3}}%
}
\newcommand{\scriptbox}[2][24]{%
  \parbox{#1\psunit}{\scriptsize\centering\hspace{0pt}#2}%
}
\rput(15,10){%
  \rput(0,49){\textbf{Fahrer}}
  \fnodetext(0,40){NE}{Navigations\-ebene}
```

```
  \fnodetext(0,24){BE}{Bahnführungs\-ebene}
  \fnodetext(0,08){SE}{Stabilisierungs\-ebene}
  \ncline{->}{NE}{BE}
  \ncput*{\scriptbox{gewählte Fahrtroute\\ zeitlicher Ablauf}}
  \ncline{->}{BE}{SE}
  \ncput*{%
    \scriptbox{gewählte Führungsgrößen:\\ Sollspur,~Sollgeschwindigkeit}%
  }
  \psframe[dimen=middle,linewidth=1.2pt,linearc=5](-12.5,0)(12.5,54)
}
\rput(50,10){%
  \rput(0,17){\textbf{Fahrzeug}}
  \fnodetext(0,8){FZ}{Längs- und\\ Querdynamik}
  \psframe[dimen=middle,linewidth=1.2pt,linearc=5](-12.5,0)(12.5,22)
}
\rput(85,10){%
  \rput(0,49){\textbf{Umwelt}}
  \fnodetext(0,40){SN}{Straßennetz}
  \fnodetext(0,24){FR}{%
    Fahrraum\\ \smallskip\scriptsize{Straße und Verkehrssituation}%
  }
  \fnodetext(0,08){FO}{Fahrbahn\-oberfläche}
  \psframe[dimen=middle,linewidth=1.2pt,linearc=5](-12.5,0)(12.5,54)
}

\ncline{->}{SE}{FZ}
\naput{\scriptbox[9]{Stell\-größen}}
\nbput{\scriptbox[9]{Lenken Gasgeben Bremsen}}
\ncline{->}{FZ}{FO}
\naput{\scriptbox[9]{Regel\-größen}}
\nbput{\scriptbox[9]{Fahrzeugbewegung}}

\psset{armA=15,armB=0,angleA=0,angleB=180}
\ncangles{->}{FZ}{FR}
\ncangles{->}{FZ}{SN}

\psset{angleA=180,angleB=0}
\ncloop[loopsize=12,arm=4.5]{<-}{SE}{FO}
\naput{\scriptbox{Istspur, Istgeschwindigkeit}}
\ncloop[loopsize=32,arm=5]{<-}{BE}{FR}
\naput{\scriptbox[30]{Bereich sicherer Führungsgrößen}}
\ncloop[loopsize=52,arm=5.5]{<-}{NE}{SN}
\naput{\scriptbox{mögliche Fahrtroute}}
\end{pspicture}
\caption{Eine mit pstricks erstellte Grafik}\label{fig:pstricks}
\end{figure}
```

12.3 Auslagern von Grafiken in separate Dateien

Damit „programmierte“ Grafiken zum einen das Dokument respektive dessen Struktur nicht unnötig unübersichtlich machen und zum anderen nur nach einer Änderung abermals kompiliert werden, um die Kompilierungsdauer möglichst gering zu halten, können diese in separate Dateien ausgelagert werden. Hierfür bieten sich die Pakete **subfiles** und **standalone** an.

13 Dokumentation von Quelltexten

Sollen in einer Dokument Quelltextauszüge beschrieben werden oder ganze Quelltextdateien eingebunden werden, so empfiehlt sich für dieses Vorhaben das Paket **listings**. Neben der bloßen Einbindung der Quelltexte bietet es zusätzlich die Möglichkeit, die Syntax in Abhängigkeit der verwendeten Programmiersprache hervorzuheben. Zusätzlich lässt sich ein Verzeichnis mit allen eingebundenen sowie direkt im Dokument angegebenen Quelltextauszügen erstellen.

```
\usepackage{listings}
```

Wird das Paket in einem Dokument mit UTF-8-Kodierung verwendet, muss dies dem Paket in der Präambel mitgeteilt werden:

```
\lstset{%
  inputencoding=utf8,extendedchars=true,
  literate=%
    {ä}{{\"a}}1 {ö}{{\"o}}1 {ü}{{\"u}}1
    {Ä}{{\"A}}1 {Ö}{{\"O}}1 {Ü}{{\"U}}1
    {~}{{\textasciitilde}}1 {ß}{{\ss}}1
}
```

Nachdem nun die Grundeinstellungen erfolgt sind, kann ein eigener Stil für eine gewünschte Programmiersprache definiert werden. Im folgenden Beispiel wird dies für ein MATLAB-Skript demonstriert. Als erstes wird mit dem Befehl `\lstdefinestyle` ein eigener **listings**-Stil erstellt:

```
\lstdefinestyle{matlab}{%
  language=Matlab,%
  basicstyle=\ttfamily,%
  commentstyle={\color{green!50!black}},
  keywordstyle={\color{blue!75!black}},%
  columns=flexible,keepspaces,%
}
```

Dabei wird die verwendete Programmiersprache definiert sowie die Formatierung für die Quelltextauszüge festgelegt. Für weitere Einstellungsmöglichkeiten sollte die Dokumentation zum Paket zu Rate gezogen werden. Der so definierte Stil lässt sich anschließend folgendermaßen verwenden.

```
\begin{lstlisting}[style=matlab]
...
\end{lstlisting}
```

Soll innerhalb eines Quelltextauszuges ein Abschnitt nicht ausgegeben sondern als LaTeX-Code interpretiert und ausgeführt werden, lässt sich mit `\lstset{escapechar=<Zeichen>}` ein spezielles Escape-Zeichen definieren. Alle innerhalb einer **listings**-Umgebung zwischen zwei Escape-Zeichen eingefügten Angaben in der Form *<Zeichen>...<Zeichen>* werden als LaTeX-Code gelesen und ausgeführt. Außerdem ist es mit dem Befehl `\lstnewenvironment` möglich, eine individuelle **listings**-Umgebung zu definieren.

Das Einbinden und Darstellen von externen Quelldateien in ein Dokument ist über den Befehl `\lstinputlisting[<Parameterliste>]{<Dateiname>}` möglich, wobei dabei im optionalen Argument ebenfalls der definierte Stil angegeben werden sollte. Es folgt die Ausgabe eines kurzen Beispiels mit dem zuvor definierten Stil `matlab`.

```
function perfect = isperfect(test_value)
  % This function checks to see if the given number is "perfect";
  %   a "perfect" number is an integer whose factors
  %   (excluding itself) add up to it.
  %
  % Example:  6 is a perfect number: 1 + 2 + 3          =  6 ==  6
  %           8 is not:              1 + 2 + 4          =  7 ~=  8
  %          28 is a perfect number: 1 + 2 + 4 + 7 + 14 = 28 == 28
  %
  % Arguments:
  %       test_value        (input)     check to see if this value
  %                                     is perfect or not
  %       perfect           (output)    if yes, set this to 1
  %                                     if no, set this to 0
  %
  % this will keep track of the sum of all integer divisors
  tempsum = 0;
  % we check all possible divisors
  for (divisor = 1 : test_value - 1)
    div_result = test_value / divisor;
    if (div_result == floor(div_result))
      % this is an integer divisor, so add it to the sum
      tempsum = tempsum + divisor;
    end
  end
  % now, does the sum equal the test_value?
  if (tempsum == test_value)
    perfect = 1;
  else
    perfect = 0;
  end
end
```

14 Und ganz zum Schluss ...

Mit Sicherheit werden Sie im Laufe der Zeit auf die eine oder andere Frage zu LaTeX 2_ε stoßen, die in diesem Tutorial nicht behandelt wurde. Sollten Sie an diesen Punkt gelangen, so empfehle zunächst einen Blick in die Kapitel Benötigte, unterstützte und empfehlenswerte Pakete sowie Praktische Tipps & Tricks im Handbuch zu TUD-Script. Sollte Ihre Frage trotzdem unbeantwortet bleiben, so wäre der Besuch eines LaTeX-Forums der nächste Schritt. Für allgemeine Anfragen kann ich unter anderem folgende Foren empfehlen:

- mrunix[15]
- golatex[16]
- LaTeX Stack Exchange[17]
- LaTeX Community[18]

Insbesondere alle TUD-Script betreffende Fragen sollten natürlich direkt im dazugehörigen TUD-LaTeX-Forum[19] gestellt werden.

Literatur

[Bie09] Christoph Bier. *typokurz – Einige wichtige typografische Regeln*. 21. Mai 2009. URL: http://zvisionwelt.wordpress.com/downloads/#typokurz (Am: 01. 12. 2014).

[Neu97a] Marion Neubauer. „Feinheiten bei wissenschaftlichen Publikationen – Mikrotypographie-Regeln, Teil I“. In: *Die TEXnische Komödie* 4/96 (Feb. 1997), S. 23–40. URL: http://www.dante.de/tex/Dokumente/dtk-neubauer.pdf (Am: 01. 12. 2014).

[Neu97b] Marion Neubauer. „Feinheiten bei wissenschaftlichen Publikationen – Mikrotypographie-Regeln, Teil II“. In: *Die TEXnische Komödie* 1/97 (Mai 1997), S. 25–44. URL: http://www.dante.de/tex/Dokumente/dtk-neubauer.pdf (Am: 01. 12. 2014).

[RV12] Axel Reichert und Herbert Voß. *LaTeX – Satz von Tabellen*. Freie Universität Berlin. 12. Jan. 2012. URL: http://userpage.fu-berlin.de/latex/Materialien/tabsatz.pdf (Am: 01. 12. 2014).

[Str07] Werner Struckmann. *Einige typographische Grundregeln und ihre Umsetzung in LaTeX*. 3. Sep. 2007. URL: http://www2.informatik.hu-berlin.de/sv/lehre/typographie.pdf (Am: 01. 12. 2014).

[Tal12] Nicola L. C. Talbot. *LaTeX for Complete Novices*. Bd. 1. Dickimaw LaTeX Series. Norfolk, UK: Dickimaw Books, 2012. ISBN: 978-1-909440-00-5. URL: http://www.dickimaw-books.com/latex/novices/ (Am: 01. 12. 2014).

[Tal13] Nicola L. C. Talbot. *Using LaTeX to Write a PhD Thesis*. Bd. 2. Dickimaw LaTeX Series. Norfolk, UK: Dickimaw Books, 2013. ISBN: 978-1-909440-02-9. URL: http://www.dickimaw-books.com/latex/thesis/ (Am: 01. 12. 2014).

[15] http://mrunix.de/forums/forumdisplay.php?38-LaTeX-Forum
[16] http://golatex.de/
[17] http://tex.stackexchange.com/
[18] http://www.latex-community.org/forum/
[19] http://latex.wcms-file3.tu-dresden.de/phpBB3/

Index

Da das Tutorial etwas umfangreicher ist, wird für alle erläuterten Optionen, Umgebungen und Befehle ein Index erstellt. Bei Elementen, die weder TUD-Script noch KOMA-Script zuzuordnen sind, wird das dazugehörige Paket mit angeführt.

Allgemeiner Index

Klassen- und Paketoptionen

A

B

C

D

E

F

G

J

N

P

S

T

X

Befehle und Umgebungen mit zugehörigen Parametern

Copy & Paste

Zum Ende des Dokumentes wird das vorgestellte Tutorial als übersetzbarer Quelltext ausgegeben, um dieses via Copy & Paste verwenden und alle Punkte nachvollziehen zu können. Bitte beachten Sie, dass – abhängig vom genutzten PDF-Betrachter – beim Kopieren die dargestellten Einzüge und Absätze möglicherweise verloren gehen können. Dies kann insbesondere aufgrund fehlender Leerzeilen zu einem unvorteilhaften Ausgabeergebnis führen. Alternativ finden Sie den folgenden LaTeX-Quelltext auch im Pfad `<texmf>/source/latex/tudscr/doc/examples/` als Datei **treatise-example.tex**.

Um das im kopierten Beispiel erstellte Literaturverzeichnis in das Dokument einbinden zu können, bedarf es dem einmaligen Aufruf von ***biber*** frühestens nach dem ersten Durchlauf von **pdfLaTeX**. Dies erfolgt mit dem Aufruf `biber <Dateiname>`. Für das Erzeugen von Abkürzungs- und Symbolverzeichnis ist das Ausführen des Perl-Skriptes `makeglossaries <Dateiname>` ausreichend. Danach ist ein weiteres mal die Verwendung von `pdflatex <Dateiname>` notwendig, um die Verzeichnisse in das Dokument einzubinden. Gegebenenfalls ist für korrekte Seitenangaben in Abkürzungs- und Symbolverzeichnis ein abermaliges Ausführen des Skriptes notwendig. *Nutzer von* **MiKTeX** *müssen für die Verwendung des Perl-Skriptes* **makeglossaries** *unter Umständen einen separaten Perl-Interpreter installieren. In Abschnitt 4.5 ist dazu mehr zu finden.*

Achtung!

```
\documentclass[english,ngerman]{tudscrreprt}
\usepackage{babel}
\ifpdftex{
  \usepackage[T1]{fontenc}
  \usepackage[ngerman=ngerman-x-latest]{hyphsubst}
}{
  \usepackage{fontspec}
}
\usepackage{scrhack}
\usepackage{tudscrsupervisor}

\AfterPackage*{hyperref}{%
\usepackage[%
  acronym,% Abkürzungen
  symbols,% Formelzeichen
  nomain,% kein Glossar
  nogroupskip,%
  toc,%
  section=chapter,%
  nostyles,%
  translate=babel,%
% mit Tex Live einfach verwendbar
  xindy={language=german-din},
]{glossaries}
\makeglossaries
}% Ende von AfterPackage
```

```
\AfterPackage*{glossaries}{%
\newglossarystyle{acrotabu}{%
  \renewenvironment{theglossary}{%
    \begin{tabu}{@{}lX<{\strut}l@{}}% 'spread 0pt' defekt in v2.9
  }{%
    \end{tabu}\par\bigskip%
  }%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand*{\glsgroupskip}{}%
  \renewcommand*{\glossentry}[2]{%
    \glsentryitem{##1}% Entry number if required
    \glstarget{##1}{\sffamily\bfseries\glossentryname{##1}} &
    \glsentrydesc{##1} &
    ##2\tabularnewline
  }
}

\newcommand*{\newformulasymbol}[5][]{%
  \newglossaryentry{#2}{%
    type=symbols,%
    name={#3},%
    description={\nopostdesc},%
    symbol={\ensuremath{#4}},%
    user1={\ensuremath{\mathrm{#5}}},%
    sort={#2},%
    #1%
  }%
}

\defglsentryfmt[symbols]{%
  \ifmmode%
    \glssymbol{\glslabel}%
  \else%
    \glsgenentryfmt~\glsentrysymbol{\glslabel}%
  \fi%
}
\newglossarystyle{symblongtabu}{%
  \renewenvironment{theglossary}{%
    \begin{longtabu}[l]{ccX<{\strut}l}% 'spread 0pt' defekt in v2.9
  }{%
    \end{longtabu}%
  }%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand*{\glsgroupskip}{}%
  \renewcommand*{\glossaryheader}{%
```

```
    \toprule
    \bfseries Symbol & \bfseries Einheit &
    \bfseries Bezeichnung & \bfseries Seite(n)
    \tabularnewline\midrule\endhead%
    \bottomrule\endfoot%
  }%
  \renewcommand*{\glossentry}[2]{%
    \glsentryitem{##1}% Entry number if required
    \glstarget{##1}{\glossentrysymbol{##1}} &
    \glsentryuseri{##1} &
    \glossentryname{##1} &
    ##2\tabularnewline%
  }%
}
}% Ende von AfterPackage

\usepackage{csquotes}
\usepackage[backend=biber,style=alphabetic]{biblatex}

\usepackage{filecontents}
\begin{filecontents}{\jobname-temp.bib}
@book{goossens94,
  author    = {Goossens, Michel and Mittelbach, Frank
               and Samarin, Alexander},
  title     = {The LaTeX Companion},
  date      = {1994},
  publisher = {Addison-Wesley},
  location  = {Reading, Massachusetts},
  language  = {english},
}
@book{knuth84,
  author    = {Knuth, Donald E.},
  title     = {The \TeX book},
  date      = {1984},
  maintitle = {Computers \& Typesetting},
  volume    = {A},
  publisher = {Addison-Wesley},
  location  = {Reading, Massachusetts},
  language  = {english},
}
@manual{hanisch14,
  author    = {Hanisch, Falk},
  title     = {Ein \LaTeX"=Bundle für Dokumente im neuen Corporate
               Design der Technischen Universität Dresden},
  date      = {2014},
  subtitle  = {Benutzerhandbuch},
  location  = {Dresden},
```

```
  language  = {german},
}
\end{filecontents}
\addbibresource{\jobname-temp.bib}

\usepackage{caption}
\captionsetup{font=sf,labelfont=bf,labelsep=space}
\usepackage{floatrow}
\floatsetup{font=sf}
\floatsetup[table]{style=plaintop}
\captionsetup{singlelinecheck=off,format=hang,justification=raggedright}
\DeclareCaptionSubType[alph]{figure}
\DeclareCaptionSubType[alph]{table}
\captionsetup[subfloat]{labelformat=brace,list=off}

\usepackage{booktabs}
\usepackage{array}
\usepackage{tabularx}
\usepackage{tabulary}
\usepackage{tabu}
\usepackage{longtable}

\usepackage{quoting}

\usepackage[babel]{microtype}

\usepackage{enumitem}
\setlist[itemize]{noitemsep}

\usepackage{ellipsis}
\let\ellipsispunctuation\relax

\usepackage{xfrac}

\usepackage{isodate}

\usepackage[colorlinks,linkcolor=blue]{hyperref}

\begin{document}

\faculty{Juristische Fakultät}
\department{Fachrichtung Strafrecht}
\institute{Institut für Kriminologie}
\chair{Lehrstuhl für Kriminalprognose}
\title{%
  Entwicklung eines optimalen Verfahrens zur Eroberung des
  Geldspeichers in Entenhausen
```

```
}
\thesis{master}
\graduation[M.Sc.]{Master of Science}
\author{%
  Mickey Mouse%
  \matriculationnumber{12345678}%
  \dateofbirth{2.1.1990}%
  \placeofbirth{Dresden}%
  \course{Klinische Prognostik}%
  \discipline{Individualprognose}%
\and%
  Donald Duck%
  \matriculationnumber{87654321}%
  \dateofbirth{1.2.1990}%
  \placeofbirth{Berlin}%
  \course{Statistische Prognostik}%
  \discipline{Makrosoziologische Prognosen}%
}
\matriculationyear{2010}
\supervisor{Dagobert Duck \and Mac Moneysac}
\professor{Prof. Dr. Kater Karlo}
\date{10.09.2014}
\makecover
\maketitle

\newcommand{\taskcontent}{%
  Momentan ist das besagte Thema in aller Munde. Insbesondere wird es
  gerade in vielen~--- wenn nicht sogar in allen~--- Medien diskutiert.
  Es ist momentan noch nicht abzusehen, ob und wann sich diese Situation
  ändert. Eine kurzfristige Verlagerung aus dem Fokus der Öffentlichkeit
  wird nicht erwartet.

  Als Ziel dieser Arbeit soll identifiziert werden, warum das Thema
  gerade so omnipräsent ist und wie dieser Effekt abgeschwächt werden
  könnte. Zusätzlich sind Methoden zu entwickeln, mit denen sich ein
  ähnlicher Vorgang zukünftig vermeiden lässt.
}
\taskform[pagestyle=empty]{\taskcontent}{%
  \item Recherche
  \item Analyse
  \item Entwicklung eines Konzeptes
  \item Anwendung der entwickelten Methodik
  \item Dokumentation und grafische Aufbereitung der Ergebnisse
}

\TUDoption{abstract}{multiple,section}
\begin{abstract}
```

```
  Dies ist der deutschsprachige Teil der Zusammenfassung, in dem die
  Motivation sowie der Inhalt der nachfolgenden wissenschaftlichen
  Abhandlung kurz dargestellt werden.
\nextabstract[english]
  This is the english part of the summary, in which the motivation and
  the content of the following academic treatise are briefly presented.
\end{abstract}

\declaration[company=FIRMA]

\tableofcontents
\listoffigures
\listoftables

\printacronyms[style=acrotabu]
\printsymbols[style=symblongtabu]

\setchapterpreamble{%
  \renewcommand*{\dictumwidth}{.4\textwidth}%
  \dictum[Johann Wolfgang von Goethe]{%
    Es irrt der Mensch, solang er strebt.%
  }%
  \bigskip
}
\chapter{Einleitung}
Nachdem nun der Vorspann und~--- bis auf das Literaturverzeichnis am
Ende des Dokumentes auf Seite~\pageref{sec:bibliography}~--- alle
Verzeichnisse erfolgreich ausgegeben wurden, wird nun die Verwendung
der weiteren Umgebungen und Befehle demonstriert, welche im Tutorial
\texturn{treatise.pdf} vorgestellt wurden.

\section{Die Verwendung von Akronymen und Symbolen}
\newacronym{apsp}{APSP}{All-Pairs Shortest Path}
\newacronym{spsp}{SPSP}{Single-Pair Shortest Path}
\newacronym{sssp}{SSSP}{Single-Source Shortest Path}

In der Graphentheorie wird häufig die Lösung des Problems des kürzesten
Pfades zwischen zwei Knoten gesucht. Dieses Problem wird häufig auch
mit \gls{spsp} bezeichnet. Es lässt sich auf die Variationen \gls{sssp}
und \gls{apsp} erweitern. Für die Lösung von \gls{spsp}, \gls{sssp}
oder \gls{apsp} kommen unterschiedliche Algorithmen zum Einsatz.

\newformulasymbol{l}{Länge}{l}{m}
\newformulasymbol{m}{Masse}{m}{kg}
\newformulasymbol{a}{Beschleunigung}{a}{\sfrac{m}{s^2}}
\newformulasymbol{t}{Zeit}{t}{s}
\newformulasymbol{f}{Frequenz}{f}{s^{-1}}
```

```
\newformulasymbol{F}{Kraft}{F}{m \cdot kg \cdot s^{-2} = \sfrac{J}{m}}

Die Einheiten für die \gls{f} sowie die \gls{F} werden aus den
SI"=Einheiten der Basisgrößen \gls{l}, \gls{m} und \gls{t} abgeleitet.
Und dann gibt es noch die Grundgleichung der Mechanik, welche für den
Fall einer konstanten Kraftwirkung in die Bewegungsrichtung einer
Punktmasse lautet:
\[\gls{F} = \gls{m} \cdot \gls{a}\]

\section{Referenzen und das Literaturverzeichnis}
Das Literaturverzeichnis wird auf Basis der nachfolgend verwendeten
Zitate erstellt und ist auf Seite~\pageref{sec:bibliography} zu finden.
In diesem Textabschnitt werden die zwei bekannten \LaTeX-Bücher
\cite{knuth84} und \cite{goossens94} sowie das Anwenderhandbuch
\cite{hanisch14} zitiert.

\section{Grafiken und Tabellen in Gleitumgebungen}
Es folgt die Demonstration von Gleitumgebungen, welche sowohl für
Grafiken als auch Tabellen verwendet werden sollten. Im vorliegenden
Beispiel kann unter Umständen der Eindruck entstehen, dass diese Seite
etwas zu überladen mit Gleitobjekten ist. Dies liegt nicht an der
Verwendung der Gleitobjekte sondern vielmehr am zu geringen Textvolumen
und den eingeschränkten Möglichkeiten von \LaTeX{}, diese an geeigneten
Stellen zu platzieren.

\subsection{Abbildungen als Gleitobjekte und das Einbinden von Grafiken}
In \autoref{fig:example} wird dargestellt, wie eine Grafik im PDF"~Format
in ein Dokument eingebunden und auf diese verwiesen werden kann. Ein
Querverweis auf ein Gleitobjekt sollte im Fließtext am besten mit Befehl
\texttt{\textbackslash autoref\{\emph{<Label>}\}} erstellt werden.
Hierfür ist ein entsprechender Anker am zu referenzierenden Objekt nötig,
welcher mit dem Makro \texttt{\textbackslash label} erzeugt wird. Dabei
ist entscheidend, dass dieser Anker erst \emph{nach} der Beschriftung des
Objektes, welche mit \texttt{\textbackslash caption} zu erstellen ist,
definiert werden sollte.

\begin{figure}
\centering
\includegraphics{TUD-black}
\caption{Beispielgrafik}\label{fig:example}
\end{figure}

\subsection{Untergleitobjekte}
Nachdem nun schon eine gleitende Abbildung und zwei gleitende Tabellen
erstellt wurden, folgt jetzt noch eine gleitende Abbildung mit zwei
Unterabbildungen. Durch die drei gesetzten Anker kann im Fließtext
sowohl auf \autoref{fig:logos} als auch auf \autoref{fig:tud} sowie
```

```
\autoref{fig:ddc} verwiesen werden.

\begin{figure}
\ffigbox[\FBwidth]%
  {\begin{subfloatrow}%
    \ffigbox[\FBwidth]%
      {\fbox{\includegraphics[height=2cm]{TUD-black}}}%
      {\caption{Eine Abbildung}\label{fig:tud}}%
    \ffigbox[\FBwidth]%
      {\fbox{\includegraphics[height=2cm]{DDC-21}}}%
      {\caption{Eine weitere Abbildung}\label{fig:ddc}}%
  \end{subfloatrow}}%
  {\caption{Eine Gleitumgebung mit zwei Abbildungen}\label{fig:logos}}%
\end{figure}

\subsection{Tabellen als Gleitobjekte}
Tabellen sollten in der \texttt{table}"=Gleitumgebung gesetzt werden.
Welche Umgebung für die Tabelle selbst dabei genutzt wird ist dabei
nicht relevant. Es können sowohl die normale \texttt{tabular}"=Umgebung
als auch die Umgebungen \texttt{tabularx}, \texttt{tabulary} sowie
\texttt{tabu} für variable Spaltenbreiten bei einer fest vorgegebenen
Tabellenbreite oder jede andere Tabellenumgebung genutzt werden.
Nachfolgend wird dies an mehreren Beispielen demonstriert.

\subsubsection{Eine gleitende tabularx-Tabelle}
Es wird eine Tabelle mithilfe der \texttt{tabularx}"=Umgebung erstellt.
Zu sehen ist diese in \autoref{tab:tabularx}. Für diese werden zuvor
neue Spaltentypen definiert.

\newcolumntype{Y}{>{\hspace{0pt}}X}
\newcolumntype{D}{>{\raggedright}Y}
\newcolumntype{E}{>{\centering}Y}
\newcolumntype{F}{>{\raggedleft}Y}

\begin{table}
\begin{tabularx}{\textwidth}{@{}DEFY@{}}
\toprule
\textbf{Linksbündig} & \textbf{Zentriert} &
\textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline
\midrule
Ein linksbündiger Blindtext zur Demonstration einer S"~Spalte &
Ein zentrierter Blindtext zur Demonstration einer T"~Spalte &
Ein rechtsbündiger Blindtext zur Demonstration einer U"~Spalte &
Ein längerer Blindtext im Blocksatz zur besseren Demonstration
einer Y"~Spalte\tabularnewline
\bottomrule
\end{tabularx}
```

```
\caption{Eine \texttt{tabularx}"=Tabelle}\label{tab:tabularx}
\end{table}

\subsubsection{Eine gleitende tabulary-Tabelle}
Es wird eine Tabelle mithilfe der \texttt{tabulary}"=Umgebung erstellt.
Zu sehen ist diese in \autoref{tab:tabulary}.

\begin{table}
\begin{tabulary}{\textwidth}{@{}LCRJ@{}}
\toprule
\textbf{Linksbündig} & \textbf{Zentriert} &
\textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline\midrule
Ein linksbündiger Blindtext zur Demonstration einer L"~Spalte &
Ein zentrierter Blindtext zur Demonstration einer C"~Spalte &
Ein rechtsbündiger Blindtext zur Demonstration einer R"~Spalte &
Ein wesentlich längerer und absolut inhaltsleerer Blindtext im
Blocksatz für eine um einiges bessere Demonstration einer J"~Spalte
\tabularnewline\bottomrule
\end{tabulary}
\caption{Eine \texttt{tabulary}"=Tabelle}\label{tab:tabulary}
\end{table}

\subsubsection{Eine gleitende tabu-Tabelle}
In \autoref{tab:tabu} ist eine weitere Tabelle mit variabler Breite der
Spalten und festgelegter Gesamtbreite zu sehen, welche in der Umgebung
\texttt{tabu} gesetzt wurde. Auch für diese wird zuerst ein neuer
Spaltentyp definiert, der die Unzulänglichkeiten der Umgebung reduziert.
Mit \texttt{\textbackslash ttabbox} aus dem Paket \texttt{floatrow} wird
die Beschriftung auf die Breite der Tabelle begrenzt.

\makeatletter
\newcolumntype{Z}{}
\renewcommand*{\NC@rewrite@Z}[1][]{%
  \NC@find>{\hspace{0pt}}X[#1]<{\@finalstrut\@arstrutbox}%
}
\makeatother

\begin{table}
\ttabbox{%
  \begin{tabu} to .8\textwidth {@{}Z[3,l]Z[3,c]Z[3,r]Z[2,j]@{}}
    \toprule
    \textbf{Linksbündig} & \textbf{Zentriert} &
    \textbf{Rechtsbündig} & \textbf{Blocksatz} \tabularnewline\midrule
    Ein linksbündiger Blindtext zur Demonstration einer Z[l]"~Spalte &
    Ein zentrierter Blindtext zur Demonstration einer Z[c]"~Spalte &
    Ein rechtsbündiger Blindtext zur Demonstration einer Z[r]"~Spalte &
    Ein Blindtext im Blocksatz innerhalb einer Z"~Spalte
```

```
    \tabularnewline\bottomrule
  \end{tabu}%
}{%
  \caption[Eine \texttt{tabu}"=Tabelle]{%
    Eine \texttt{tabu}"=Tabelle in Verbindung mit dem Befehl
    \texttt{\textbackslash ttabbox}, welcher vom Paket \texttt{floatrow}
    für Beschriftungen in Objektbreite bereitgestellt wird%
  }%
  \label{tab:tabu}%
}
\end{table}

\section{Zitate}
Bei der Verwendung von wörtlichen Zitaten sollten diese als solche
gekennzeichnet werden.
\enquote{Dies ist ein zugegebenermaßen nicht sehr sinnvolles Zitat.}
\cite[58]{hanisch14}
Für eine möglichst gut nachvollziehbare Referenz sollte nicht nur
das Werk selber sondern zumindest die Seitenzahl und gegebenenfalls
der Absatz der originalen Textstelle angegeben werden.
\begin{quoting}
\enquote{%
  Dies ist ein noch sinnloseres Zitat. Allerdings wird zumindest die
  Wirkung der Umgebung \texttt{quoting} bei der Absatzauszeichnung
  deutlich.

  Wie zu sehen ist, wird der zweite Absatz~--- wie jeder weitere~---
  aufgrund der Option \texttt{parskip=false} eingezogen.
}
\cite[sinngemäß nach][\pno{} 12, zweiter Absatz]{hanisch14}
\end{quoting}
Ebenfalls sollten sinngemäße Zitate mit einer möglichst genauen Referenz
angegeben werden. Dies kann im Laufe der Arbeit auch für einen selbst von
Vorteil sein, wenn beispielsweise die originale Textpassage noch einmal
analysiert werden soll.

\printbibliography[heading=bibintoc]\label{sec:bibliography}%

\end{document}
```