Insert here your thesis' task.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Visual Analysis of Plans for Multi-Agent Path Finding with Continuous Time (MAPF-R)

Evgenii Abdalov

Katedra softwarového inženýrství
Supervisor: doc. RNDr. Pavel Surynek, Ph.D.

June 28, 2020

Acknowledgements

Děkuji všem a za vše. Nevíte-li, co sem napsat, příkaz odstraňte.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In V Praze on June 28, 2020 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Abdalov, Evgenii. *Visual Analysis of Plans for Multi-Agent Path Finding with Continuous Time (MAPF-R)*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020. Also available from: ⟨`http://site.example/thesis`⟩.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Závěrečná práce, LaTeX.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Thesis, LaTeX.

Contents

Introduction **1**

1 Understanding of MAPF-R problem and its visual analysis **3**
1.1 Multi-Agent Pathfinding problem description 3
1.2 Conflict based search algorithm 5
1.3 Visual analysis 8

2 Design and implementation of the MAPF-R visualization tool **11**

3 The MAPF-R visualization tool documentation **13**

4 MAPF-R economic assessment **15**

Conclusion **17**

A Seznam použitých zkratek **19**

B Návod k použití této šablony **21**
B.1 Výběr základu 21
B.2 Použití šablony 22
B.2.1 Typografie 22
B.2.2 Obrázky 22
B.2.2.1 Formáty grafiky 23
B.2.2.2 Získání vhodného formátu 23
B.2.2.3 Plovoucí prostředí 23
B.2.2.4 Verze obrázků 24
B.2.3 Tabulky 25
B.2.4 Literatura 25
B.2.5 Sazba URL 26

C Obsah přiloženého CD **27**

List of Figures

1.1 example of CT binary tree 8
1.2 The Visual Analytics process 8
1.3 The main components of visual graph analysis 9

B.1 Příklad obrázku 24
B.2 Gnuplot černobíle 24
B.3 Gnuplot barevně 25

List of Tables

B.1 Příklad tabulky 25

Introduction

In modern technological and economical environment the importance of data has increased dramatically. As a matter of fact, data-driven desicion making has a lead role in succesfull management techniques. Nevertheless, in order to extract genuine value from your data, the one must process data correctly. Dealing with vast amount of data has a potential to increase probability of making a miscalculation in data processing. In such situation we could implement tools that facilitate visual analysis of data, i.e. use visual interfaces to process data.

Among various algorithmic problems there is a certain class of problems, that requires proper visual analysis in order to be solved correctly - those are problems on graphs. Graph is an abstraction, that models interconnections between set of objects. It is a collection of vertices and edges, that are defined by pairs of vertices. One of the most common type of problems that are solved on graphs is space-orientated problems, where vertex is an abstraction for a location(cities,rooms, airports) and an edge is an abstraction for a route between them. The problem is commonly stated, that we need to find the shortest path from a location A to a location B. Let's make an assumption that there is only one agent, that travels on graph. In this case, in order to succesfully find a path we need to take into account only restrictions that are put by space-graph. However, if we add more agents, that are travelling on graph, we should take other agents movements as a restrictions that must be considered while finding the solution. This kind of problems is called Multi-Agent Path Finding problem. Multi-Agent Pathfinding(MAPF) problem deals with finding paths for multiple agents, so they can avoid collisions with each other and reach their target destination. The solution to MAPF problem is as complex as amount of agents involved, which leads to the necessity of using visual analysis of the solution. Thus, there is a motivation to design a visualiation tool. Visualization of MAPF problem is crucial for further analysis of the solution, finding its redundancies and detecting ways to make it more effective.

The main goal of this thesis is to design and develop a visualization tool for multi-agent pathfinding problem with continuous time that enables its visual analysis. In interest of achieving this goal, following tasks must be accomplished. Initially,we should study relevant literature on MAPF-R problem and identify aspects deserving visualization. Secondly, we will design and implement visualization tool of MAPF-R plans. After that, provide documentation of both the user part and the program so that future upgrades are possible. Finally, analyze economic potential of the visualization tool in logistic domains where MAPF-R is used as an underlying concept for navigating robotic manipulators.

Chapter 1

Understanding of MAPF-R problem and its visual analysis

In this chapter we will determine theoretical basis of MAPF-R problem, secondly we will characterize what visual analysis is, its main functions and tools, and indicate essential aspects of MAPF-R problem that have to be visualized.

1.1 Multi-Agent Pathfinding problem description

Multi-Agent Pathfinding problem deals with planning paths for multiple agents so they can avoid collisions with each other and reach their target destination. The MAPF problem has a vast range of domains where it is being applied, including robot planning, autonomous vehicles, videogames, automated warehouses.

The input of standard MAPF problem for k agents is a tuple $< G, s, t >$, where G = (V,E) is a graph, $s : [1, ..., k]$ is a set of starting vertices for k agents, $t : [1, ..., k]$ is a set of target vertices for k agents. Action is a function $a : V- > V$, where $a(v)- > v'$ suggests that if an agent is at vertex v, after performing action A it will be in vertex v'. Every agent has two types of actions: `wait` and `move`. `wait` means that agent stays at the current vertex, `Move` means that agent moves from its current vertex v to an adjacent vertex v'. Single-agent plan is a sequence of actions $\pi = (a1, ..., an)$, that leads agent i from s[i] to s[i] after being executed. The output of standard MAPF problem is a solution $\pi = (\pi 1, ..., \pi k)$, that contains k single-agent plans, where one agent has one single-agent plan.

The majority of MAPF problems is solved on grids with discrete time, where the duration of every action is one time step and each agent occupies exactly one single location in every time step. All agents are considered to be of the same shape and size and have the same constant speed. It uses space-time maps to describe agent's location at a certain moment of time - $Cell(x, y, t)$,

where (x, y) are space coordinates on the map and (t) is a timestep.

The standard MAPF problem includes such types of conflicts as:

vertex conflict occurs if agents are planned to occupy the same vertex at the same time step.

edge conflict occurs if agents are planned to traverse the same edge at the same time at the same time-step in the same direction.

following conflict occurs when one agent is planned to occupy a vertex that was occupied by another agent on the previous time step.

cycle conflict occurs if in the same time step every agent moves to a vertex that was previously occupied by another agent, forming a rotating cycle pattern.

swapping conflict occurs if agents are planned to swap locations in a single time step.

The solution has addional parameters, that evaluate a MAPF solution.

makespan is the number of time steps, that are required for all agents to reach their target position. For a MAPF solution $\pi = (\pi 1, ..., \pi k)$, the makespan is $\|\pi i\|$, where πi is a single-agent plan with the maximum amount of steps for agent i.

sum of costs is the sum of time steps, that are required for each agent to reach their target position, which is $\sum 1 \leq i \leq k \|\pi i\|$.

The MAPF problem in descrete time is less applicable in real life situations, where processes occur in real time environment. The MAPF-R is Multi-Agent Pathfinding with continuous time, where agent motion is planned for a certain time interval. This time interval is a minium time considered to be safe for planned actions, which means there would be no collisions during this time interval. Similar to the standard MAPF problem, MAPF-R has as an input a workspace and agents parameters. Workspace is a 2D space, which can be represented as a graph G = (V, E), where vertices V are location, that agents can occupy, and edges E are line trajectories, which agents move along. Agents parameters are start location and target location. An output of algorithm is a MAPF-R problem solution, which is a joint plan for agents. Joint plan consists of individual plans for each agent, where the plan is a sequence of actions for agent needed to be taken so it could reach its target position. Duration of a move is translation speed times the length of the edge. In order to evaluate the MAPF-R solution such parameters as the makespan and the sum of costs are still applicable, however the definition of a plan cost differs. Instead of the number of time steps, cost of a plan is the sum of the durations of its constituent actions.

1.2 Conflict based search algorithm

When focusing on algorithms that solve MAPF problem, the key requirement is that the solution has to be cost-optimal. Although there are several algorithms for solving MAPF-R problem, algorithm, that is considered the most efficient, is CBS, which stands for Conflict based search algorithm.

First, CBS in discrete time should be described. As an input we have workspace and agents parameters. The path is how agent i moves on space-time grid, `solution` is a set of k paths for the given set of k agents. Generally speaking, the conflict-based search algorithm finds separate plans for each agent, then it determines conflicts between those plans and solves it by re-plannig whith specific constraints, that has been put on individual plans.

There is two types of `constraints` in conflict-based search algorithm, which are vertex constraint and edge constraint, Vertex constraint $(a[i], v, t)$ is a configuration when agent i is not allowed to occupy vertex v at the time t. Similarly to vertex constraint, edge constraint $(a[i], v1, v2, t)$ is a configuration when agent i is not allowed to start moving from vertex $v1$ to vertex $v2$ at the time-step t and arriving at the time-step $t+1$. We say, that `path` for agent i is `consistent` if it satisfies all its `constraints`. The `solution` is `consistent` if all its `paths` are `consistent`.

Two types of `conflicts` are being considered in CBS: a `vertex` conflict and an `edge` conflict. A vertex conflict is a situation $(a[i], a[j], v, t)$, when agents i and j are planning to occupy vertex v at the same moment of time t. An edge conflict is a situation $(a[i], a[j], v1, v2, t)$, when agents i and j are planning to swap locations $v1$ and $v2$ between time-step t and time-step $t+1$. The solution to MAPF problem is considered to be `valid`, in case there is no conflict between any two single-agent plannings. In spite of being consistent, solution can be invalid if this solution has paths that have conflicts with each other. `Conflicts` are resolved by imposing `constraints`.

CBS consists of high-level search and low-level search. The high-level search is a constraint-tree CT, which is a binary tree. Every node N includes a set of constraints(N.constraints), a solution(N.solution) and the total cost(N.cost). The root of CT has an empty set of constraints. The child node inherits parent constraints and adds to that one new for one agent. One node consists of constraints for only one agent. The solution inside the node is a set of k paths, one path for each agent. These paths must be consistent with given constraints - path for agent i is consistent with constraints imposed to agent i. The total cost is summed over all the single-agent path costs.

As far as low-level search is concerned, any single-agent pathfinding algorithm(SIPP) can be used, for example A*. The low-level search is looking for individual paths for each agent with given set of constraints. As an input for the low-level search a set of constraints for a node N is given. As an output, low-level search returns the shortest path for agent i that is consistent with imposed constraints assosiated with agent i in node N. Afterwards,

`validation` of the node has to be processed. The `validation` is conveyed by iterating through all the time-steps and matching the locations reserved by all agents. The node N is declared to be the `goal` node in case there is no conflict, i.e. no two agents plan to be at the same time at the same location. On the contrary, the node N is declared to be the `non-goal` node if a conflict $C = (a[i], a[j], v, t)$ between two or more agents has been found as a result of `validation`.

In order to resolve a conflict $C = (a[i], a[j], v, t)$, a new constraint must be added. It is illustrated in 1.1. Only one agent i or j is allowed to occupy location v at a time t, subsequently two options is possible - impose a constraint $(a[i], v, t)$ or a constraint $(a[j], v, t)$. We need to explore both options, as we are not aware which one is more optimal then another. Therefore, a non-goal node N is split onto two children, each one obtains a new constraint, $(a[i], v, t)$ and $(a[j], v, t)$, and inherits the set of constraints from its parent - N.constraints. Note, that low-level search may be performed only for an agent which is connected with newly added constraint, as other agents paths remain the same since no new constraints have been added for them. The high-level search treats edge conflicts in a similar manner as vertex conflicts. In case plan does not have neither vertex nor edge conflicts, the solution has been found.

CCBS, conflict based search algorithm in continuous time follows CBS pattern. However, CCBS has its differences from CBS. In order to detect conflicts, CCBS uses a geometry-aware collision detection mechanism. In order to resolve conflicts, CCBS uses a geometry-aware unsafe-interval detection mechanism. Instead of location-time pairs, CCBS adds constraints over pairs of actions and time ranges. CCBS lower-level search uses a variation of SIPP, single agent pathfinding algorithm, customized to handle CCBS constraints.

In CCBS agents can have any geometric shape, different speed and acceleration and agent's actions can have any duration, a conflicts in CCBS can occur between agents traversing different edges. Conflict in CCBS is a conflict between `actions`. Formally speaking, a conflict configurtion $(a[i], t[i], a[j], t[j])$ means, that if agent i executes action $a[i]$ during the time period $t[i]$ and if agent j executes action $a[j]$ during the time period $t[j]$, then collision will happen between agent i and j.

In CCBS the high-level search is similar to its discrete version in CBS. It also uses CT to resolve conflicts by imposing constraints. The collision detection mechanism determines if node N is a goal node. In case node N is a non-goal node, it splits into two children nodes with new constraints. In order to compute new constraints, that will be added, CCBS finds for each action its `unsafe` intervals. The unsafe interval of action $a[i]$ is the maximal time interval starting from $t[i]$ during which performing $a[i]$ will cause a collision with performing performing $a[j]$ at a time $t[j]$.

In terms of low-level search, CCBS uses SIPP adopted to handle continuous time constraints. For each location $v \in V$ SIPP finds a set of `safe intervals`.

Algorithm 1 High level of CBS

```
Input: MAPF instance
Root.constraints = {null}
Root.solution = find individual paths by the low_level_search()
Root.cost = SIC(Root.solution)
insert Root to OPEN

while OPEN not empty do
  P <− best node from OPEN
  Validate the paths in P until a conflict occurs
  if has no conflict then
    return P.solution
  end if
  C <− first conflict (a[i], a[j], v, t) in P
  for agent a[i] in C do
    A <− new node
    A.constraints <− P.constraints + (a[i], v, t)
    A.solution <− P.solution
    Update A.solution by invoking low_level_search(a[i])
    A.cost = SIC(A.solution)
    if A.cost < INF then
      Insert A to OPEN
    end if
  end for
end while
```

A safe interval is considered to be a maximal time interval in which an agent is able to arrive or wait at location v without colliding with any moving obstacles. Extending a safe interval in any direction will lead to collision, thus this safe interval is `maximal`. Let's assume $C = (i, a[i], (t1, t2))$ is CCBS constraint for agent i. In this case action $a[i]$ may be `wait` action or `move` action. In case, $a[i]$ is a `wait` action, let v be a start location and v' be a target location. If the agent arrives at v in time step $t \in [t1, t2)$ then we delete an action that moves agent from v to v' during the time period t and exchange it for an action that is waiting at v until $t2$, and then agent is allowed to move to v'. In case $a[i]$ is a `wait` action, let v be a location at which the agent is waiting. Then, we forbid the agent from waiting during $t \in [t1, t2)$ by splitting safe intervals accordingly - if $[0, INF)$ is considered to be solely one safe interval for location v, then we split it in two safe intervals $[0, t1]$ and $[t2, INF)$.

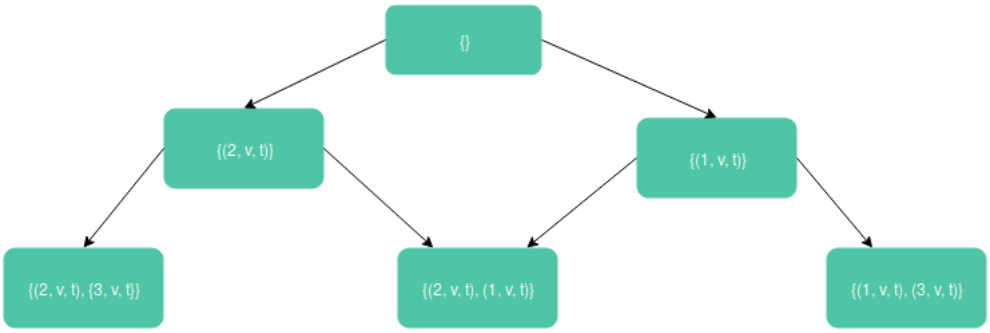

Figure 1.1: example of CT binary tree

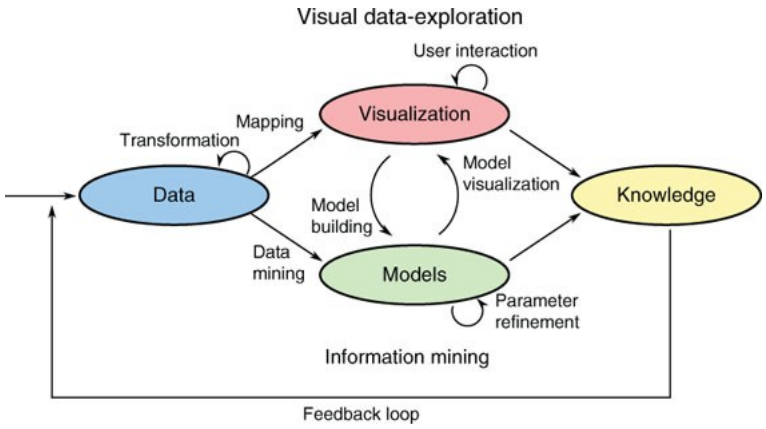

Figure 1.2: The Visual Analytics process

1.3 Visual analysis

In this section visual analysis of MAPF-R problem will be discussed. The correct MAPF-R problem visualization is necessary as long as it allows to convey a visual analysis of the solution. Visual analysis is meant to facilitate the detection of conflicts and redundant elements in terms of output solution. As a matter of fact, in case of the MAPF-R problem containing tens to hundreds of agents it is extremely hard to determine if output solution is correct without visual representation 1.2.

Visual analytics is described as the science of analytical reasoning assisted by interactive visual interfaces. Visual analysis incorporates aspects of `visual representation`, `user interaction` and `algorithmic analysis` 1.3. Those aspects are a foundation of a sufficient visual analysis tools and are closely interconnected. For instance, algorithmic analysis may function as a preprocessing step that determine specific graph layout for visual repre-

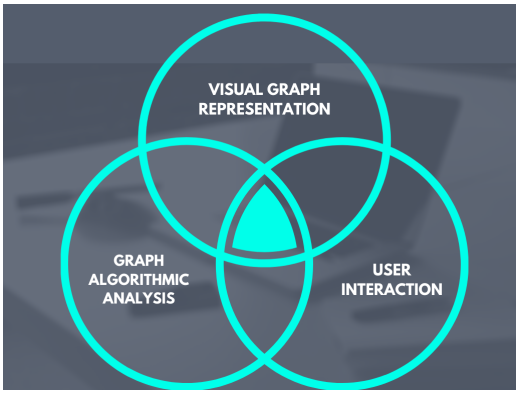

Figure 1.3: The main components of visual graph analysis

sentation. User interaction aim to discover different aspects of the data by changing visual representation and requesting different algorithmic processing of the data. User interaction may be minimal, where data is processed automatically, or, on the contrary, data processing is fully dependent on parameters inserted by user. User interaction can be classified by such criteria as `user intention`, `task`, `user action`. Those criteria are interrelated; for example, one task might be obtained by performing several actions, or several intentions might include the same task. As far as graph visualization is concerned, standard user interaction techniques might be applied, such as highlighting, brushing, linking, panning and zooming. Zooming and panning facilitate navigation in any direction and change the zoom level within the view. Highlighting is making an emphasis on interesting elements of the visual representation.

As an input data set MAPF-R visualisation tool will operate with graph charachteristics, agent parameters and its plans. The graph characteristics data set is composed of following components: a number of edges, a number of vertices and which ones are connected with edges, coordinates for each vertex within the cartesian coordinate system. The agent parameters data set includes agent shape and size parameters, its velocity and acceleration, and agent start position as well as agent target position. The agent plans data set represents agent behaviour within the timeline, i.e. agent action during specific time intervals.

Visual representation is based on data, which means that any modification in data affects the visual representation. For instance, data filtering influences which parts of the data set are going to be displayed and that could change graph modification or layout. In case of MAPF-R visualisation tool, user does not modificate or filter data set, although user could input different agent plans for the same graph, which results in different agent movement animation on the same graph layout.

There are three main techniques for displaying general graphs: node-link based, matrix-based and hybrid. The `node-link based` technique is more compatible with our needs for a visualisation tool. Node-links techniques use links between graph elements to display their relationship. The main challenge of this technique is the layout, which is the placement of the nodes, so that certain degree of graph readability is supported. The main requirements to the layout are: the nodes must not overlap, the number of edge crossing must be minimal, edge lenght should be homogeneous. Such tasks are solved by specific graph layout algorithms. There is a sub-technique, which is `graphs with geographic reference` , for example transportation graph. The geographic location dictates the precise location of the nodes and possibly of the edges. Subsequently, there is no need for a graph layout algorithm in order to place nodes on the screen, although there might be problems with long edges and crossings.

As far as visual representation is concerned, the main challenge is to establish an acceptable level of physical abstraction, i.e. determine which aspects of problem should be visualized and which could be ignored. The workspace is presented as a graph on 2D space, where nodes interconnected with edges mean that agent is physically allowed to traverse between them. The position of vertices are specified by coordinates within the cartesian coordinate system. Edge between two vertices is a line connecting two points. The exact position of edges is imperative since it has directl influence to the possibility of collision between agents. Other physical parameters of space, such as lightning, air temperature, floor level, height of ceiling, can be ignored in visualization.

As previously stated, MAPF-R problem takes into account agent physical shape and size, subsequently physical restrictions, which are implied, should be visualized properly. It will allow user to detect danger areas, where the risk of collision between agents is higher than average. Since agent's constraints are presented as a set of time intervals, agent have to move in real time according to its velocity and acceleration.

In addition to visual representation, we aim to collect statistics data as a part of data analysis. Data analysis enhances visual observation and facilitates the probability of making an analytical discovery. The statistics data indicates algorithm perfomance and effectivness. In case of MAPF-R problem visualization tool, we will collect data about overall time duration of the solution as well as time duration for individual agents to reach its target destination from starting point. Furthermore, we aim to collect data of how much time agent moves and how much time agent waits, using this data we could estimate moving/waiting ratio for each agent.

Using `visualisation analysis` in combination with `data analysis`, it is possible to convey a perfomance comparison of several MAPF-R problem solution. Based on this analysis it is feasable to find out which algorithm solves MAPF-R problem more effectively.

Chapter 2

Design and implementation of the MAPF-R visualization tool

TO_DO

Chapter 3

The MAPF-R visualization tool documentation

TO_DO

Chapter 4

MAPF-R economic assessment

TO_DO

Conclusion

Conclusion

Appendix A

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Appendix B

Návod k použití této šablony

Tento dokument slouží jako základ pro napsání závěrečné práce na Fakultě informačních technologií ČVUT v Praze.

B.1 Výběr základu

Vyberte si šablonu podle druhu práce (bakalářská, diplomová), jazyka (čeština, angličtina) a kódování (ASCII, UTF-8, ISO-8859-2 neboli latin2 a nebo Windows-1250).

V české variantě naleznete šablony v souborech pojmenovaných ve formátu práce_kódování.tex. Typ práce může být:

BP bakalářská práce,

DP diplomová (magisterská) práce.

Kódování zdrojového souboru (LaTeX), ve kterém chcete psát, může být:

UTF-8 kódování Unicode,

ISO-8859-2 latin2,

Windows-1250 znaková sada 1250 Windows.

V případě nejistoty ohledně kódování doporučujeme následující postup:

1. Otevřete šablony pro kódování UTF-8 v editoru prostého textu, který chcete pro psaní práce použít – pokud můžete texty s diakritikou normálně přečíst, použijte tuto šablonu.
2. V opačném případě postupujte dále podle toho, jaký operační systém používáte:
   - v případě Windows použijte šablonu pro kódování Windows-1250,
   - jinak zkuste použít šablonu pro kódování ISO-8859-2.

V anglické variantě jsou šablony pojmenované podle typu práce, možnosti jsou:

bachelors bakalářská práce,

masters diplomová (magisterská) práce.

B.2 Použití šablony

Šablona je určena pro zpracování systémem LaTeX 2$_\varepsilon$. (Začátečníci v LaTeXu mohou využít např. [**?**].) Text je možné psát v textovém editoru jako prostý text, lze však také využít specializovaný editor pro LaTeX, např. Kile.

Pro získání tisknutelného výstupu z takto vytvořeného souboru použijte příkaz `pdflatex`, kterému předáte cestu k souboru jako parametr. Vhodný editor pro LaTeX toto udělá za Vás. `pdfcslatex` ani `cslatex` *nebudou* s těmito šablonami fungovat.

B.2.1 Typografie

Při psaní dodržujte typografické konvence zvoleného jazyka. Česky psané „uvozovky“ zapisujte použitím příkazu `\uv`, kterému v parametru předáte text, jenž má být v uvozovkách. Anglické otevírací uvozovky se v LaTeXu zadávají jako dva zpětné apostrofy, uzavírací uvozovky jako dva apostrofy. Často chybně uváděný symbol " (palce) nemá s uvozovkami nic společného.

Dále je třeba zabránit zalomení řádky mezi některými slovy, v češtině např. za jednopísmennými předložkami a spojkami (vyjma „a“) nebo mezi číslicí a měrnou jednotkou. To docílíte vložením pružné nezalomitelné mezery – znakem `~`. V tomto případě to není třeba dělat ručně, lze použít program `vlna`.

Nezapomeňte také na rozlišení „vodorovných čárek“, které je dáno nejen typografickými zvyklostmi, ale i pravidly českého pravopisu. Pro dělení slov (na konci řádku) nebo jejich spojování nebo v rámci složenin používejte rozdělovník (v LaTeXu se zapisuje jako `-`), naopak pomlčku (v LaTeXu zapsanou jako `--`) užívejte pro význam rozmezí nebo rozsahu a nebo jako větnou pomlčku (namísto interpunkce). Zcela jiným znakem je též mínus (ve stejné výšce a stejné délky jako vodorovná čárka znaku plus), v LaTeXu se zapisuje pouze v matematickém režimu.

Více o typografii viz [**?**].

B.2.2 Obrázky

Pro umožnění vkládání obrázků je vhodné použít balíček `graphicx`, samotné vložení se provede příkazem `\includegraphics`. Takto je možné vkládat obrázky ve formátu PDF, PNG a JPEG jestliže používáte pdfLaTeX nebo

ve formátu EPS jestliže používáte LaTeX. Doporučujeme preferovat vektorové obrázky před rastrovými (vyjma fotografií).

B.2.2.1 Formáty grafiky

Z hlediska reprezentace obrazových informací existuje dělení grafických formátů na rastrové a vektorové. Ty první reprezentují obrázek pomocí barev jednotlivých bodů, ty druhé pomocí informací (souřadnice, barva) o částech obrázků (úsečka, polygon, plocha). Z toho plyne vhodnost formátů pro určitý obsah: rastrové pro fotografie, vektorové pro snadno popsatelné obrázky (zejména ty, které obsahují text, jasné tvary apod.). Mezi vektorové souborové formáty patří např. PDF, EPS, SVG, WMF; rastrové obrázky lze najít v souborech typu PNG, JPEG, GIF, TIFF.

Rastrové obrázky neumožňují, na rozdíl od vektorových, zvětšení beze ztráty vizuálně postřehnutelné kvality. Vzhledem k vlastnostem grafických formátů a nárokům na vzhled (zejména) vytištěné práce důrazně doporučujeme využít vektorovou grafiku pro všechny obrázky znázorňující typický vektorový obsah (např. diagramy) a rastrové využívat pouze pro fotografie. Důsledně se pro vektorový obsah vyvarujte vkládání grafiky využívající ztrátovou kompresi (JPEG)! Vkládáte-li už do práce rastrovou grafiku, dbejte na dostatečné rozlišení (300 dpi je naprosté minimum). Z tohoto důvodu je většina obrázků získaných z webu nevhodná.

B.2.2.2 Získání vhodného formátu

Pro získání vektorových formátů PDF nebo EPS z jiných lze použít některý z vektorových grafických editorů. Pro převod rastrového obrázku na vektorový lze použít rasterizaci, kterou mnohé editory zvládají (např. Inkscape). Pro konverze lze použít též nástroje pro dávkové zpracování běžně dodávané s LaTeXem, např. `epstopdf`. Běžný formát SVG (specifikace viz [**?**]) sice není možné vkládat přímo (zatím), konverzi však zvládne řada vektorových grafických editorů.

B.2.2.3 Plovoucí prostředí

Příkazem `\includegraphics` lze obrázky vkládat přímo, doporučujeme však použít plovoucí prostředí, konkrétně `figure`. Například obrázek B.1 byl vložen tímto způsobem. Vůbec přitom nevadí, když je obrázek umístěn jinde, než bylo původně zamýšleno – je tomu tak hlavně kvůli dodržení typografických konvencí. Namísto vynucování konkrétní pozice obrázku doporučujeme používat odkazování z textu (dvojice příkazů `\label` a `\ref`).

Figure B.1: Ukázkový obrázek v plovoucím prostředí

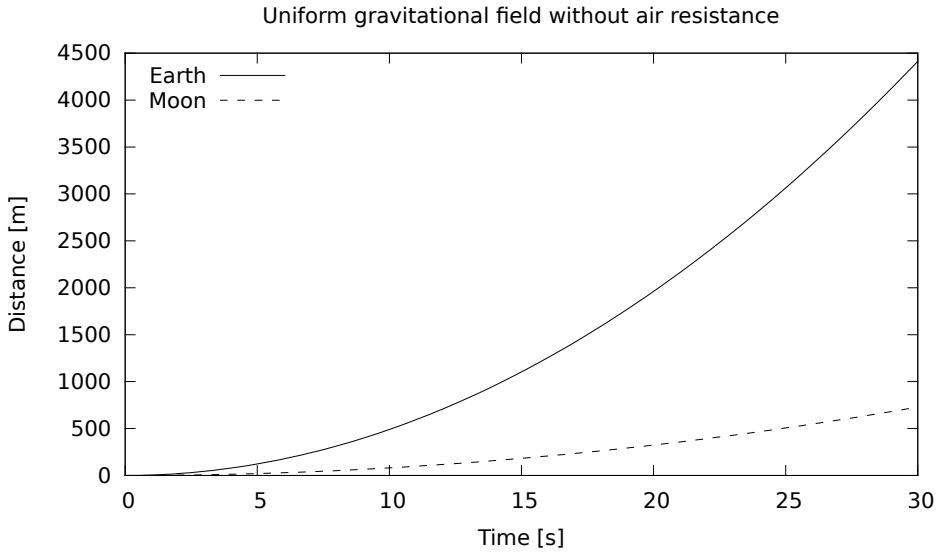

Figure B.2: Černobílá varianta obrázku generovaného programem Gnuplot

B.2.2.4 Verze obrázků

Může se hodit mít více verzí stejného obrázku, např. pro barevný či černobílý tisk a nebo pro prezentaci. S pomocí některých nástrojů na generování grafiky je to snadné.

Máte-li například graf vytvořený v programu Gnuplot, můžete jeho černobílou variantu (viz obr. B.2) vytvořit parametrem `monochrome dashed` příkazu `set term`. Barevnou variantu (viz obr. B.3) vhodnou na prezentace lze vytvořit parametrem `colour solid`.

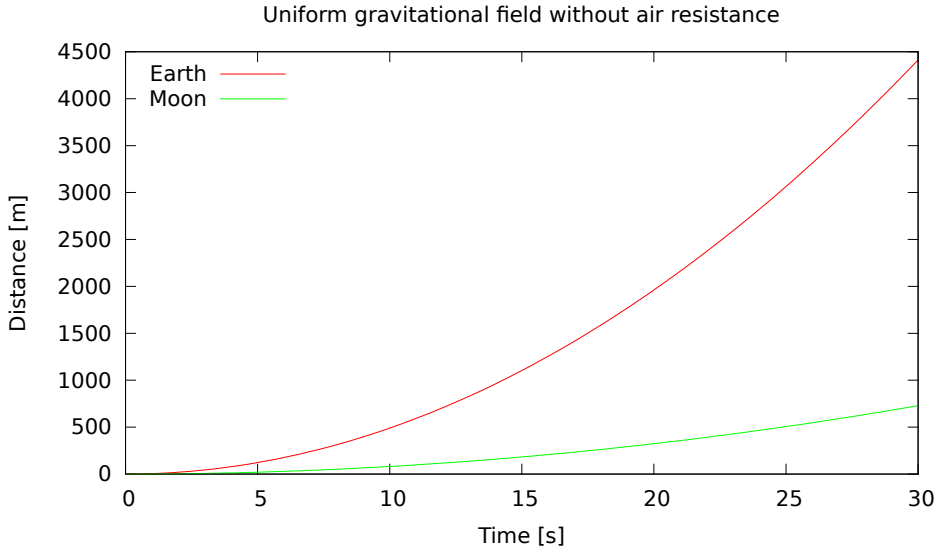

Figure B.3: Barevná varianta obrázku generovaného programem Gnuplot

Table B.1: Zadávání matematiky

Typ	Prostředí	LaTeXovská zkratka	TeXovská zkratka
Text	`math`	`\(...\)`	`$...$`
Displayed	`displaymath`	`\[...\]`	`$$...$$`

B.2.3 Tabulky

Tabulky lze zadávat různě, např. v prostředí `tabular`, avšak pro jejich vkládání platí to samé, co pro obrázky – použijte plovoucí prostředí, v tomto případě `table`. Například tabulka B.1 byla vložena tímto způsobem.

B.2.4 Literatura

Vše, čeho nejste autorem (myšlenky, nápady, text, obrázky, ...) by mělo být řádně ocitováno – pokud možno původní zdroj. Vzhledem k charakteru této práce (odborná) upřednostňujte důvěryhodné a odborné zdroje (existuje-li tištěná verze, citujte raději tu). Důrazně se tedy *vyvarujte citace z Wikipedie* (kromě odůvodněných a nejnutnějších případů).

Citování (tedy přesné specifikování použitého informačního zdroje a také odkaz na něj z textu) je vhodné provést podobně jako v tomto textu, tedy v souladu s aktuálně platnou normou ČSN ISO 690 [**?**].

B.2.5 Sazba URL

Pro vkládání URL a podobných informací doporučujeme použít příkaz `url` ze stejnojmenného balíčku. Zajistíte tím jednak odlišení adresy od ostatního textu pomocí jiného písma a také zalamování na konci řádku.

Chcete-li vkládat odkazy (funkční v PDF), použijte příkaz `href` z balíčku `hyperref`.

Appendix C

Obsah přiloženého CD

Vhodným způsobem vizualizujte obsah přiloženého média. Lze použít balíček `dirtree` a vytvořit např. následující výstup (adresáře src a text s příslušným obsahem jsou *povinné*):

```
readme.txt ..... stručný popis obsahu CD
exe ..... adresář se spustitelnou formou implementace
src
    impl ..... zdrojové kódy implementace
    thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
    thesis.pdf ..... text práce ve formátu PDF
```