Assignment Problems for Optimizing Text Input

Anna Maria Feit
Assignment Problems for Optimizing Text Input

Anna Maria Feit

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall T2 of the school on 11 June 2018 at 12.

Aalto University
School of Electrical Engineering
Communications and Networking
User Interfaces
Supervising professor
Professor Antti Oulasvirta, Aalto University, Finland

Preliminary examiners
Professor Michel Beaudouin-Lafon, Université Paris-Sud, France
Associate Professor Geehyuk Lee, KAIST, Republic of Korea

Opponent
Dr. Shumin Zhai, Google Inc., USA
Abstract

Text input methods are an integral part of our daily interaction with digital devices. However, their design poses a complex problem: for any method, we must decide which input action (a button press, a hand gesture, etc.) produces which symbol (e.g., a character or word). With only 26 symbols and input actions, there are already more than $10^{26}$ distinct solutions, making it impossible to find the best one through manual design. Prior work has shown that we can use optimization methods to search such large design spaces efficiently and automatically find the best solution for a given task and objective. However, work in this domain has been limited mostly to the performance optimization of keyboards.

The Ph.D. thesis advances the field of text-entry optimization by enlarging the space of optimizable text-input methods and proposing new criteria for assessing their optimality. Firstly, the design problem is formulated as an assignment problem for integer programming. This enables the use of standard mathematical solvers and algorithms for efficiently finding good solutions. Then, objective functions are developed, for assessing their optimality with respect to motor performance, ergonomics, and learnability. The corresponding models extend beyond interaction with soft keyboards, to consider multi-finger input, novel sensors, and alternative form factors. In addition, the thesis illustrates how to formulate models from prior work in terms of an assignment problem, providing a coherent theoretical basis for text-entry optimization. The proposed objectives are applied in the optimization of three assignment problems: text input with multi-finger gestures in mid-air, text input on a long piano keyboard, and -- for a contribution to the official French keyboard standard -- input of special characters via a physical keyboard. Combining the proposed models offers a multi-objective optimization approach able to capture the complex cognitive and motor processes during typing. Finally, the dissertation discusses future work that is needed to solve the long-standing problem of finding the optimal layout for physical keyboards, in light of empirical evidence that prior models are insufficient to respond to the diverse typing strategies people employ with modern keyboards.

The thesis advances the state of the art in text-entry optimization by proposing novel objective functions that quantify the performance, ergonomics and learnability of a text input method. The objectives presented are formulated as assignment problems, which can be solved with integer programming via standard mathematical solvers or heuristic algorithms. While the work focused on text input, the assignment problem can be used to model other design problems in HCI (e.g., how best to assign commands to UI controls or distribute UI elements across several devices), for which the same problem formulations, optimization techniques, and even models could be applied.

Keywords  Text entry, combinatorial optimization, computational interaction, human-computer interaction
Preface

It takes a village to raise an academic. This is what I realized thinking back on these last years. I was surprised to see how many people I met that shaped my thinking and inspired my work. Before I present my thesis project, I want to take the time to acknowledge their contribution to this dissertation and my growth as a researcher.

First and foremost I want to thank Antti – I could not have wished for a better supervisor to guide me through these past years. The high standards you set for your students but also your own work, motivated me every day to give my best. You gave me the feeling that I worked together with you and not for you while teaching me so many things along the way. Whatever the problem, you showed me how to turn it into our advantage; and no matter the topic, you had the right book at hand. I admire your ambition, your genius, and your dedication to your students. They set an example, which I will strive to live up to also in the coming years. Thank you for giving me all the support and opportunities I needed to become the researcher I am today.

I also want to thank my pre-examiners Professor Geehyuk Lee and Professor Michel Beaudouin-Lafon; your thorough comments and valuable suggestions contributed to improving this dissertation. I am honored to have Dr. Shumin Zhai as an opponent; you are a great inspiration to my research and I admire your work.

I am proud that I was the first member of the User Interfaces research group founded by Antti when he joined Aalto University. Throughout the years, he brought many great people to our group and created a climate of inspiring discussions and active collaborations. I particularly enjoyed having had Daryl as my colleague. You are the most social nerd I know and the perfect office mate. KumariPaba and Janin, thank you for our many interesting conversations, which were sometimes even related to our
work. I also want to thank Jussi for answering all my annoying statistics questions; and Mathieu – I admire you for your patience with your fellow Frenchmen.

I am deeply grateful to Perttu, Olli, and Samuli for all the hours you spend putting tiny markers on hands, processing eye-tracking videos and motion capture data, and searching for bugs in study software. You contributed a great deal to my dissertation work.

Thank you to all my colleagues who were a part of our research group over the years and who created a stimulating and inspiring environment for me to grow as a researcher; Byungjoo, Crista, Eve, Joanna, Kashyap, Kseniia, Markku, Morteza, Niraj, and Sunjun; and all the talented students and interns I got to know and work with over the years, particularly Chen, Ece, Gabriela, Hendrik, Johanna, Mehmet, and Vivek. Thank you also to all the members of the HelsinCHI Community.

Before I joined Aalto University, I made my first steps as a researcher during my Master studies as part of Antti’s HCI group at MPII and MMCI at Saarland University. I greatly enjoyed my time there and sincerely thank you, Gilles, Myroslav, and Srinath, for your involvement and friendship. I am particularly grateful to Mirella for all the hours you spent on the piano, for believing in my crazy ideas and joining me in all my undertakings. Right from the beginning, I was given the opportunity to visit other research groups and universities. Looking back, I am astonished and proud to see that I became part of a large network of colleagues from all over the world. I particularly enjoyed teaching at the summer school on computational interaction in Glasgow and Zurich, together with brilliant people, including John, Rod, Otmar, Per Ola, and Andrew.

During my internship at Microsoft Research, I made many valuable experiences that let me grow as an independent researcher. I am grateful to Merrie for this opportunity; after 8 years in Computer Science and HCI, you were the first female supervisor and it was inspiring to see you excel as a researcher and mother alike. I also want to thank all the members of the Enable group, in particular, my co-authors Anne, Arturo, Harish, Shaun, and Shane; and the PALS I got to know during my time there.

Besides the people already mentioned, I want to thank all my collaborators, in particular, Andreas and Maximilian from MPII; Marco and Pascal from DFKI; Daniel from LMU Munich; Valentin and Pascal from University of Stuttgart / LMU Munich; Christoph and Seonwook from ETH Zurich; and Hannu and Petteri from TypingMaster.
During my studies at Aalto University, I was part of the Department for Communications and Networking. I thank all the technical and administrative staff who made my life easier throughout the years. I am grateful to the ELEC Doctoral school for funding my work, and to the ACM, Nokia and the HPY research foundation for their financial support. Furthermore, my research was funded by the Academy of Finland project COMPUTED and the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant agreement No 637991).

I am proud to finish this chapter and am curious where life will lead me in the coming years, but no matter where I know that I can count on these people to be there along the way. Most important of all I know that my family will stand behind me the same as they always have. I am deeply grateful to my partner Luc; I am amazed how you took on this adventure as your own without so much as a second thought. You were a piece of home for me in a strange country; you kept me sane when everything went crazy and pulled me back to my life outside the university. Thank you for sharing your life with me. To my sister Katharina, thank you for being my best friend despite the distance between us; I know you will always be there for me the same as I am for you. My life would be boring without you. To my parents, Markus and Sabine; I always knew that whatever I wanted to do in life, you would be there to support me and catch me if it goes wrong. This knowledge still gives me the courage to take on any new challenge with a smile and live without fear of the future. It is my sincere hope that I will be able to give my daughter the same security and support that you gave me so that she can pursue whatever dream she will have in the future.

Helsinki, May 14, 2018,

Anna Maria Feit
Contents

Preface .................................................. 1
List of Publications .................................... 7
Author's Contribution .................................... 9
List of Figures .......................................... 11
List of Tables ........................................... 12
Symbols .................................................. 15

1. Introduction .......................................... 17
   1.1 The Design Problem .............................. 19
   1.2 Design Using Combinatorial Optimization ...... 21
       1.2.1 Formulation as an assignment problem .... 21
       1.2.2 Modeling of evaluative knowledge ......... 22
       1.2.3 Instantiation of optimization problems .... 23
   1.3 The Research Goal and Methods ................. 23
       1.3.1 Experimental methods ...................... 24
       1.3.2 Mathematical Modeling ..................... 26
       1.3.3 Optimization methods ...................... 27
   1.4 Contributions of This Work ...................... 28
   1.5 The Structure of the Thesis ...................... 31

2. Background: Combinatorial Optimization for Designing
   Text Input Methods .................................. 33
   2.1 The Letter-Assignment Problem .................. 35
       2.1.1 The decision variables and design space .... 36
       2.1.2 Constraints .................................. 36
       2.1.3 The objective function ..................... 37
   2.2 Optimization Methods and Complexity .......... 38
       2.2.1 Exact optimization methods ............... 39
       2.2.2 Heuristic optimization methods .......... 40
       2.2.3 Multi-objective optimization ............. 41
   2.3 Modeling Limitations with the Assignment Problem . 43
   2.4 Summary .......................................... 44
3. Related Work: The History and State of the Art in Text Entry Optimization

3.1 Manual Design of Physical Keyboards  46
3.2 Model-based Optimization of Physical Keyboards  47
3.3 Optimization of Soft Keyboards  49
3.4 Summary  51
  3.4.1 The space of optimizable text entry methods  51

4. Objective Functions for the Letter-Assignment Problem  55

4.1 Motor Performance  58
  4.1.1 One-finger input  59
  4.1.2 Chord input  60
  4.1.3 Multi-finger input  63
4.2 Ergonomics  67
  4.2.1 Strain  69
  4.2.2 Muscle fatigue  70
  4.2.3 Finger individuation  72
4.3 Learnability  73
  4.3.1 Motor learning and complexity  75
  4.3.2 Transfer of motor skills  76
  4.3.3 Memorability  77
  4.3.4 Intuitiveness and familiarity  79
4.4 Input Recognition  81
  4.4.1 Noise  83
  4.4.2 Disambiguation  85
4.5 Summary  86

5. Application of Objectives to the Optimization of Novel Text Input Methods  89

5.1 Transfer of Motor Skills  91
  5.1.1 The design space and constraints  92
  5.1.2 Objectives and the optimization model  93
  5.1.3 The task instance  94
  5.1.4 Optimization technique  95
  5.1.5 Outcome  96
5.2 Mid-air Text Input with Chording Gestures  100
  5.2.1 The design space and constraints  102
  5.2.2 Objectives and the optimization model  103
  5.2.3 The task instance  106
This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


Author’s Contribution

Publication I: “PianoText: Redesigning the Piano Keyboard for Text Entry”

The general idea of using the piano keyboard as a text-entry device was provided by Antti Oulasvirta. Under his supervision, I independently developed and implemented the optimization approach for assigning characters to piano keys, using musical structures for making use of the pianist's expertise. I implemented and conducted both of the user studies, and I analyzed the associated data. The paper was written in collaboration with Antti Oulasvirta.

Publication II: “Investigating the Dexterity of Multi-Finger Input for Mid-Air Text Entry”

The idea and theoretical approach for using multi-finger chord gestures for mid-air input was jointly developed by all four authors. I developed the models for performance and finger individuation and generalized them to multi-finger gestures, designed the experimental studies, and analyzed the associated data. The gesture recognizer was developed by Christian Theobalt and Srinath Sridhar. It was implemented by Srinath Sridhar, who also developed the experimental software, and the studies were then conducted by both of us. Antti Oulasvirta formulated the objective function and implemented the optimization algorithm, using the input data from the experiment. The paper was written in close collaboration by all the authors.
Publication III: “How We Type: Movement Strategies and Performance in Everyday Typing”

I identified the need to capture and study the hand and finger movements of modern typists. I developed, implemented, and conducted the experimental study; analyzed the related data; and identified the most important findings, with feedback from Antti Oulasvirta and Daryl Weir. Our joint idea of clustering participants by their finger-to-key mapping pattern was implemented by Daryl Weir, and he and I identified and interpreted the final clusters, with feedback from Antti Oulasvirta. The paper was written collaboratively by all three authors.

Publication IV: “Observations on Typing from 136 Million Keystrokes”

With feedback from Per Ola Kristensson and Antti Oulasvirta, I developed the methodology for the online typing test. It was implemented and maintained by research assistant Samuli De Pascale. I identified both the research questions and the variables for analysis. Then, I supervised Vivek Dhakal’s performance of the data analysis. Together we developed the approach for clustering the data, which he then implemented. We interpreted the results jointly with Antti Oulasvirta. I wrote the bulk of the publication, with contributions from Vivek Dhakal and Antti Oulasvirta and with feedback from Per Ola Kristensson.
List of Figures

4.1 Variables for determining the angular speed of a single finger ........................................... 61

5.1 Frequency distribution of bigrams in English and music .................................................... 95
5.2 Optimized assignment of letters to piano keys ................................................................. 96
5.3 Frequency distribution of note pairs in music and PianoText ............................................. 97
5.4 Examples of mid-air chording gestures .................................................................................. 100
5.5 The degrees of freedom of the hand studied and their discretization .................................... 102
5.6 Fitts’ Law models for each joint and an example of relative coactivation ............................ 107
5.7 The Azerty layout .................................................................................................................. 112
5.8 Example stimulus for capturing special-character performance ......................................... 119
5.9 Movement times for entering special characters .................................................................. 121
5.10 Comparison of an optimized layout with Azerty and a manually created layout .................. 124

6.1 Performance of participants in the 136M-Keystrokes dataset ................................................ 139
6.2 Average inter-key intervals for letters in different keyboard rows ...................................... 140
6.3 Average inter-key interval for bigrams typed with hand alternation, finger alternation, the same finger, or letter repetition ................................................................. 142
6.4 Typing strategies for the left hand .......................................................................................... 144
6.5 Typing strategies for the right hand ....................................................................................... 145
List of Tables

1.1 Overview of the publications .................................. 32
3.1 Overview of prior work in text entry optimization ........ 53
4.1 Objective categories for text-entry optimization .......... 57
4.2 Heuristic criteria for optimizing a keyboard layout ...... 64
4.3 Objective functions for optimizing motor performance ... 68
4.4 Objective functions for optimizing ergonomics .......... 74
4.5 Objective functions for optimizing learnability .......... 82
4.6 Objective functions for optimizing input recognition ... 87
5.1 Frequency of special characters in French text .......... 118
5.2 Criteria for rating the similarity between two symbols .. 120
5.3 Overview of the optimization cases ......................... 127
6.1 Phenomena of touch typing ascertained via prior work .. 132
List of Abbreviations

CMC (joint) . Carpometacarpal (joint)
DOF .......... Degree of freedom
DSK .......... Dvorak Simplified Keyboard
HCI .......... Human–computer interaction
ID .......... Index of difficulty
IKI .......... Inter-key interval
IP (joint) .... Interphalangeal (joint)
LAP .......... Linear assignment problem
MCP (joint) .. Metacarpophalangeal (joint)
QAP .......... Quadratic assignment problem
UI .......... User interface
WPM .......... Words per minute
List of Abbreviations
Symbols

$A_k$ . . . . . . . . Set of target angles
C . . . . . . . . . . . Recognition confusion matrix
c_{ik} . . . . . . . . Interaction cost
$C_{b\phi}$ . . . . . . . Relative coactivation
$D_{kl}$ . . . . . . . . Distance
$F(x)$ . . . . . . . . Objective function
$f_{kl}$ . . . . . . . . Frequency of notes in music
$i, j$ . . . . . . . . Symbols
$I$ . . . . . . . . . . . Individuation index
$k, l$ . . . . . . . . Input actions
$L(i)$ . . . . . . . . Character position
$M$ . . . . . . . . . . Number of input actions
$\mathcal{M}$ . . . . . . . Mnemonic sets
$N$ . . . . . . . . . . Number of symbols
$N_{kl}$ . . . . . . . . Set of neighboring keys
$p_i$ . . . . . . . . . Symbol frequency
$p_{ij}$ . . . . . . . . . Bigram frequency
$s_{ij}$ . . . . . . . . Similarity
$t_{kl}$ . . . . . . . . Movement time
$T_{kl}$ . . . . . . . . Movement time obtained from lookup table
Symbols

$U_k$ . . . . . . . . Sensing uncertainty

$W_l$ . . . . . . . . Width

$X$ . . . . . . . . Design space

$x$ . . . . . . . . Design vector

$x_{ik}, y_m$ . . . . . Binary decision variables

$\alpha_{k,\theta}$ . . . . . Joint angle

$\Theta_k$ . . . . . . . Hand configuration

$\theta, \phi$ . . . . . . Joints

$\omega$ . . . . . . . Objective weight
1. Introduction

The input of text is an integral part of our interaction with digital systems, from traditional computers to mobile devices and many other systems. The amount of text entered has tremendously increased over the last few decades, as much of the verbal communication in the workplace and with friends and family has been replaced with text (email, social media, online chat, etc.). At the same time, the form factor of computing devices has changed dramatically, and efficient and low-effort text input is needed not only in desk-type environments but also for mobile devices and wearable technology and with large or distant devices (e.g., TVs and cars) – in public, private, and professional settings alike, such as computing systems in factories and hospitals. Across all these scenarios, the fundamental principle of the text input is largely the same: selection of symbols from a grid principally arranged in the so-called Qwerty layout, named after the first six characters on the top row. However, the selection technique ranges from direct selection with one or multiple fingers, as performed on the standard physical or soft keyboard, to distant or indirect pointing with the hand [99] or a control device (e.g., a mouse, touchpad, or infrared controller [148]) and even to relative control of a selection cursor by such means as a rotary control [176] or joystick [169].

The suitability of these techniques for selecting letters in the Qwerty layout varies greatly. For example, expert typists can reach 100 or more words per minute (WPM) on physical keyboards when using multiple fingers [133, 166, 167] and 60 WPM on mini-Qwerty keyboards using two thumbs [20]. In contrast, typing rates on soft keyboards reach only 30–40 WPM with mobile phones [3] and 10–20 WPM [2, 78] with smartwatches. Character selection on large screens via remote pointing can be done at 13–19 WPM [99, 148]. Less is known of how the user experience or ergonomics of these methods compare to those of physical keyboards.
The adoption of new text input methods is often impeded by long learning times and lower initial levels of performance. Only a few alternatives actually get used in practice. Continuous stroking across a keyboard from one letter to the next has been proposed as an alternative to selection by pointing [73, 74]. This is available for many commercial soft keyboards on mobile phones. The approach allows higher input for expert users, thanks to open-loop control that does not require visual attention for online correction of movements. Also, novice users can easily transition from traditional pointing to gesture input without their performance suffering. In addition, other methods based on continuous gestures have been proposed [45, 98, 117, 164, 168, 171], but users have not yet adopted any of these in large numbers. One reason might be that significant learning time is required before performance comparable to that with the traditional selection method is achieved. The origins of this family of techniques lie in attempts to improve the recognition of handwritten text (reviewed elsewhere [119]), a feature available for many computing devices. Recognition of different writing styles remains difficult, and performance with handwriting input is limited.

Chording keyboards [32, 89] reduce the number of buttons to be controlled, by using combinations of key presses to enter characters or letter sequences. However, they are difficult to learn, since no visual guidance helps users memorize the key combinations. The only chording keyboard in widespread use is the stenotype, which is mainly used by professionals, such as court reporters, to record the spoken word. Typists undergo intensive training over several years, at the end of which they can enter text at rates of 180–225 WPM [23].

The hands and fingers are generally the preferred method for interaction with computers and specifically for text input. An alternative is speech input, which grows in popularity as recognition improves. It is used predominantly for “natural” interaction with personal assistant systems or when hands-free input is desirable (e.g., in cars or other immersive environments). However, there are many persistent challenges, such as error correction or interference with cognitive processes of text generation [147], which preclude the ubiquitous use of speech for computer interaction. Moreover, performance approaching the rate of natural speech cannot be reached in practice [92].

Other alternatives to manual input are used primarily by people with mobility impairments – for instance, text input via gaze or head track-
ing [42, 95, 178]. While successful in allowing these people to interact with computers, they are often slow and cumbersome to use when compared to manual input methods.

In summary, many text input methods exist, but most of the ones in real-world use are based on the same principle, wherein a user performs a series of manual input actions (i.e., movements of hands or fingers that can be recognized by the device as a discrete event) to select characters. These actions may differ fundamentally on the basis of the interaction technique. Nevertheless, the design of the text input methods is still largely the same: characters are spatially arranged into the Qwerty layout. This universal design is not suitable for every device and interaction technique, and it influences efficiency, ergonomics, and learnability.

In this thesis, I present and elaborate on techniques of model-based optimization for designing text input systems by taking into account the performance, ergonomics, learnability, and other aspects of the input actions performed by the user to operate the system. I place the focus on hand and finger movements, as used for most text input methods; alternative techniques, such as speech or gaze input, are beyond the scope of this thesis. However, approaches similar to those presented here may be applicable in those scenarios too.

1.1 The Design Problem

As discussed by MacKenzie and Soukoreff [92, 152], entering text can be modeled in terms of Shannon's noisy channel [146], as the process of communicating through a channel subject to various sources of noise. On one end is a user (the sender) who encodes the to-be-entered message into input actions and transmits them through the channel. On the other end, there is a sensor (the receiver) that decodes the message back into the corresponding symbols (e.g., characters, words, or other units of language). The recognition of the input actions can be an analogue operation, as in the case of typewriters, or done by digital sensors, as with multitouch systems, cameras, accelerometers, gyroscopes, etc. The channel through which the message is transmitted is noisy. This means that the message could get corrupted at various points: during creation (e.g., using the wrong gesture for a character), transmission (e.g., moving too rapidly and pressing the wrong key on a keyboard), or reception (e.g., being uncertain in recognition of a hand gesture). The throughput of the channel, the rate of useful
information being received, is influenced by the input actions, the sensing, and the noise [152]. Following this model, for the thesis I define a text input method thus:

A text input method takes a series of input actions (movements performed by the user's hands and fingers) and recognizes them as symbols (characters, words, or other units of language).

This definition covers a wide range of manual text input methods and demarcates the scope of the thesis. The corresponding design space is very large. It includes decisions on the interface of the text input method for performance of the input actions (e.g., touchscreen or physical buttons, button shape and size, or even absence of a visible interface), the interaction technique that defines the input actions (continuous gestures vs. discrete button presses, how many hands are required for input, etc.), the use of intelligent methods for recognizing and supplementing the user's input (e.g., a computer vision algorithm for recognizing hand gestures but also word prediction or autocorrection methods), and many more. With this definition I do not attempt to delineate the entire design space of text input methods; rather, I want to highlight the wide range of methods to which the findings presented in this thesis are applicable. Independent of the decisions on which input actions to offer and how to recognize them, one fundamental problem is central to the design of all these text input methods, which is the focus of the thesis:

What is the best assignment of symbols to input actions, minimizing the cost of text input?

In terms of the above model, this corresponds to the question of what the best system is for encoding/decoding symbols to/from input actions if one is to maximize the throughput of the channel. In practice, the cost of text input can depend on many factors, among them the user's motor performance, aspects of ergonomics and/or learnability, and also the cost of recognizing the input actions. This design problem typically has a very large solution space. If we consider even just the input of 26 symbols via 26 input actions, there are already \(26! > 10^{26} = 100,000,000,000,000,000,000,000,000,000,000\) individual ways to assign all the symbols to the input actions (for example, to assign the English letters to the button presses on a keyboard).
1.2 Design Using Combinatorial Optimization

Traditional design approaches commonly rely on an iterative process to generate and refine designs manually on the basis of heuristics, prior experience, subjective feedback, etc. [129]. Such a process is unsuitable for exploring millions of designs and makes it difficult to assess the impact of design decisions on various evaluation criteria or find the best tradeoff between decisions. In contrast, given a machine understandable definition of the design problem, a computer could easily generate and evaluate millions of solutions within few seconds.

Therefore, the core approach applied for this thesis is the use of combinatorial optimization to solve the design problem expressed above. Combinatorial optimization refers to the process of finding the best combination of design decisions under the given objective criteria [122]. It relies on algorithmic and mathematical methods to automatically generate and evaluate solutions. This allows the systematic exploration of very large design spaces and can provide guarantees as to the goodness of a solution and its optimality.

Traditional design can be seen as a “conscious decision-making process by which information [...] is transformed into an outcome” [161, p. 17]. Combinatorial optimization automates creation of this outcome by systematically generating and evaluating solutions on the basis of the given information. However, this imposes three challenges to the designer:

1. Explicitly formulating the design decisions and variables of the design problem.

2. Capturing the relevant information – that is, the evaluative knowledge used to assess the goodness of a design – in the form of mathematical functions.

3. Obtaining task-specific parameters for instantiating the objective functions.

The goal for the thesis is to advance the field of text entry optimization with respect to these challenges. This is examined in the following sections.

1.2.1 Formulation as an assignment problem

The problem of finding the best keyboard layout has long been recognized [18, 31, 43, 66, 120] as an instance of the assignment problem, a
well-known class of optimization problems concerned with the question of how to assign items of one group to items of another [17]. It can be used to model many real-life problems, such as the assignment of jobs to machines, positions for satellites, and the wiring of computer backboards [17]. Mathematically formulating the problem of designing a text input method as an assignment problem exposes the structure of the problem and the relations between the decisions. Moreover, it allows us to build on a large body of research on mathematical and algorithmic approaches to finding the optimal solution for such a problem (see Burkard et al. [16, 17] provides an overview). However, prior work on text entry optimization has been limited to keyboard layouts, with many scholars not explicitly formulating the underlying design problem ([80, 115, 127, 150]). This has limited the optimization capability. In this thesis, I formulate the more general problem of finding the best assignment of input actions to symbols as a linear or quadratic assignment problem. This allows me to extend the space of optimizable text input methods, from what has already been proposed to novel interaction techniques and symbol sets, such as mid-air gestures and special characters.

1.2.2 Modeling of evaluative knowledge

The optimization process relies on the formulation of an objective function (also called a cost or gain function) that captures the design knowledge in an executable form used to evaluate the goodness of a solution. In contrast to engineering, where design knowledge comes in the form of evaluable models from the natural sciences and economics, design of user interfaces (UIs) is based on information from disciplines such as psychology, behavioral science, and sociology. In those fields, knowledge about the user and the system is often tacit or provided in terms of theories, heuristics, or empirical observations. A key challenge for optimization of text input methods is to formulate this knowledge as mathematical functions that can be maximized or minimized. Prior work has been concerned mainly with the optimization of motor performance when text is being entered with one finger or in touch typing. Researchers have typically used a mathematical model for pointing performance (Fitts’ Law [36]) or heuristics of touch-typing performance established many decades ago [30, 134, 131]. My work advances the field of text entry optimization in that I present the mathematical formulation of novel objective functions. Empirical knowledge of typing performance with physical keyboards comes primarily from studies
of professional typists operating typewriters. Little work has been done on modern typing, where the keyboard is different in its physical properties and the typing tasks too are different from those of the 1930s–1980s. I have conducted extensive empirical studies on modern typing that inform the development of new objective functions for optimizing the design of physical keyboards and other text input methods. However, my work goes beyond the performance of physical keyboard. I propose new objective functions for capturing motor performance with novel interaction techniques, such as mid-air gestures, but also aspects of ergonomics and learnability – among them fatigue and strain, finger individuation, memorability, and skill transfer – never considered in prior work.

1.2.3 Instantiation of optimization problems

Designing a specific text input method by means of combinatorial optimization requires instantiating the optimization problem, that is parameterizing the objective functions with task-specific parameters. These are typically obtained from empirical studies or the literature. The publications written in the thesis project contribute new empirical observations surrounding typing on physical keyboards, special-character entry, and the performance of mid-air gestures. I have used these results, together with observations from earlier literature, to formulate and optimize three novel text input methods: typing on a piano keyboard to exploit a pianist’s expertise, using mid-air gestures for entering text, and inputting a large set of special characters on a physical keyboard.

1.3 The Research Goal and Methods

The ultimate goal behind this thesis is to design better text input methods that allow entering text efficiently and with little effort. Following prior work on text entry, I apply methods of combinatorial optimization that allow efficient exploration of the large design spaces using mathematical and algorithmic methods. However, as characterized above, applying optimization methods comes with three major challenges. My objective is to advance text entry research with respect to these challenges in the following ways:

1. Establishing the assignment problem as a general model for designing any text input method.
2. Formulating objective functions for evaluating the goodness of text input methods beyond keyboard layouts.

3. Advancing the empirical understanding of motor behavior during text input in order to formulate and instantiate objective functions.

To this end, I used several quantitative research methods, described below.

### 1.3.1 Experimental methods

To advance our understanding of how users interact with a text input method, I have conducted a series of experimental studies, including long-term studies, laboratory experiments, and online studies. When choosing an empirical method, we trade off between criteria, which all express desirable elements but interfere with each other [103]: (1) *generalizability*, or validity of the results across the population of users; (2) *precision* of the measures studied, obtained by controlling extraneous factors that are not subject to study; and (3) *realism* of the studied task relative to the context in which we want the evidence to be applicable. I have set my focus in accordance with the goal of the research at hand, emphasizing the most relevant factor(s) and choosing the empirical method accordingly.

_Laboratory studies_ are common in text entry research, where the focus often is on collecting precise measurements of performance rather than studying realistic situations. Text input involves complex cognitive, perceptual, and motoric processes [134]. Therefore, it is important to maximize precision and thus minimize interference caused by factors that are not relevant with regard to the research questions. The goal in my experimental studies was to understand or evaluate the motor behavior during text input, including factors that influence motor performance, ergonomics, and movement strategies. As is common in text entry research [92], for all publications I used a transcription-based typing task, wherein the goal for participants was to transcribe randomly presented phrases as quickly and accurately as possible. While such a task is seldom encountered in real-world contexts, it affords a focus on the motor processes involved in typing, excluding creative and other cognitive processes involved in more realistic tasks, such as text generation. Hence, the behavior of participants is rendered more comparable.

Evaluation of a new text input method is challenging. Estimating the performance or usability of a method for text input requires long-term training, which is seldom feasible in a laboratory setting, on account
of the expenses and time involved. Accordingly, studies often have to trade off between long-term training and a large number of participants. One approach to assessing the performance achievable with a system is an “accelerated learning” experiment [73]. Here, participants repeatedly practice the input of words until their performance plateaus. While such a protocol is unrealistic and may limit the external validity of the findings, it is suitable if the purpose is to gain an understanding of the upper bounds of performance with the given system. It has the benefit that long learning times are avoided and experiment costs hence are lower. This method has been used to evaluate the optimized design presented in Publication II and to model the special-character input performance in the optimization case described in Section 5.3.

Web-based experiments are studies conducted over the internet. These have gained popularity on account of improvements to technology and readily available crowdsourcing platforms (e.g., LabintheWild [123] and Amazon Mechanical Turk.¹) Some platforms pay their participants, while others rely on subjects’ intrinsic motivation (e.g., curiosity or seeking fun). Although online studies do not allow as rigorous control as laboratory studies do, they can be conducted more quickly and with a larger, more diverse sample, while still yielding similar results [40, 123]. Moreover, a larger sample increases the statistical power of the findings and yields better estimates and shapes of distributions [124].

A common problem with web-based experiments is low data quality arising from technical problems or from participants not taking the study seriously. Self-selection of participants and high dropout rates are other possible threats to their validity [108, 124]. The design of the study task and software can help to minimize the impact of such problems [123, 125]. Text entry experiments commonly implement detailed logging, to the granularity of individual key presses. This allows a higher level of control, wherein “cheaters” can be easily identified even while the study is still in progress and outliers due to technical problems can be removed post hoc.

I conducted two online studies, wherein the goal was to understand high-performance motor behavior in typing on physical keyboards. For Publication IV, I collected observations from more than 160,000 participants, using an online typing test hosted on the website of TypingMaster Oy.²

¹ See https://www.mturk.com/mturk/.
² At https://www.typingtest.com/. The company’s main site is http://www.typingmaster.com/.
The company offers typing tests and courses. The goal was to gain a better understanding of modern typing behavior and the factors influencing performance. This large dataset, rare in text entry research, allowed detailed statistical analysis, aiding in linking keystroking patterns to behavioral phenomena and typing performance.

In an unpublished study described in Section 5.3, the goal was to collect data on the speed of typing letters on the physical keyboard in combination with those keys used for special-character entry. Including the use of modifier keys (Alt and Shift), this required observations for more than 7,500 key pairs. To gather enough data, I conducted an online experiment for which we recruited more than 630 paid participants via the crowdsourcing platform CrowdFlower\(^3\) and about 270 unpaid participants through advertising on the typing test webpage from the earlier study.

Long-term experiments are important in text entry research, although, for reasons of their high cost, they are seldom conducted. Text input is a complex motor skill that typically requires many hours of practice. Therefore, long-term experiments offer greater realism than, for example, the accelerated learning method described above.

In Publication I, I have reported on intensive training of a single participant to enter text with the optimized input method. After 140 hours of training, performance was still improving. This long-term experiment used a mix of controlled performance testing and self-paced learning, wherein the participant was free to choose from a set of various transcription tasks to use in practicing input for a given number of hours per week. While the small number of participants does not allow generalization from the findings, such long-term training provides a higher degree of realism than a shorter, more controlled laboratory study does, and allows obtaining an estimate of learnability and the performance achievable with the device.

1.3.2 Mathematical Modeling

A mathematical model is a simplification and abstraction of the real world, formulated using mathematical constructs [10]. They are approximations of the real world, aggregating information on those factors that are of interest with respect to the research question or goal. In my work, I use quantitative mathematical models to capture the empirical knowledge about the users’ behavior. These models are then formulated as objective

\(^3\) See https://www.crowdflower.com/.
functions for evaluating the goodness of a text input method on the basis of the model’s output. This is essential in the process of optimization.

Exact optimization methods, such as the integer programming described in this thesis (see Chapter 2), rely on mathematical properties of the objective function so require the use of models that employ closed mathematical expressions to describe the relationship between the design dimensions and the criteria for optimization (e.g., algebraic models, functional equation models, and lookup tables). The use of algorithmic models that are defined in terms of a simulation procedure in combination with exact optimization methods is not straightforward, because the process they model must be simulated on a computer before one can obtain a prediction. In Chapter 4, I elaborate on objective functions that are based on such mathematical models, derived from the component papers of this thesis or from prior work. In some cases, this work necessitated development of models based on others’ empirical observations and heuristics, models that can then be used in evaluating a text input method.

1.3.3 Optimization methods

The overarching goal for the thesis project is the design of better text input methods. My main approach toward this end is to use optimization methods. The traditional (user-centered) design process relies on creativity, heuristics, prior experience, subjective feedback, etc. [129] for manual creation and refinement of solutions to a design problem. However, the design spaces in user-interface design are commonly very large, as exemplified right at the start of the dissertation, and such a process only allows exploring a small part of this space or yields a few point observations in particular parts of the space. In contrast, optimization offers a systematic way of automatically creating and evaluating millions of solutions from the design space in just a short time. The aim is to find the optimal solution with respect to a given objective that captures relevant aspects of how the users behave when interacting with a design. However, optimization only supplements the manual design process. It relies on designers or researchers defining the design problem (mathematically), including the decisions that span the design space, and setting the objectives for evaluating the goodness of solutions. This definition work for the design of text input methods is the principal goal behind the thesis.

There are several approaches to optimization, with the choice among them depending mainly on the characteristics of the problem and one’s
Introduction

objectives. As are many other problems in user-interface design, the design

task I look at in this thesis is a combinatorial optimization problem wherein

the goal is to find the best design from a finite set of solutions, where each
design is defined by a combination of several discrete decisions [114, 122].

For many combinatorial optimization problems, there are various mathe-
matical formulations, highlighting certain characteristics of the problem
and allowing the use of various solution approaches [16]. I formulate the
assignment problem as a quadratic or linear integer program (depend-
ing on the characteristics of the objective function). This allows me to
use exact methods to solve the problem. In contrast to heuristic methods,
these guarantee searching the full design space for the best solution and
provide bounds for the goodness of intermediate solutions. The problem
formulation and choice of method are further described in Chapter 2.

1.4 Contributions of This Work

This thesis contributes to research into text entry and computational
interaction. In line with the goals stated above, the main outcome of this
work can be summarized with the following contribution statements:

C1 This thesis shows that the assignment problem can be generalized
beyond keyboard layouts to model the problem of assigning symbols to
(manual) input actions for text entry. This yields a coherent theoretical
basis for optimizing text input methods.

C2 This dissertation contributes a set of novel objective functions that
mathematically capture the evaluative knowledge for assessing the
goodness of a text input method with respect to aspects of performance,
learnability, ergonomics, and input recognition.

C3 This work expands the space of optimizable text input methods, with
optimization of three new design cases: typing on the piano, performing
mid-air input, and entering special characters.

C4 This thesis advances the understanding of modern typing behavior
and informs revision of the optimization principles used in prior work
to optimize physical keyboard layouts.

C1: Prior work has shown that the problem of finding the best spatial
arrangement of letters on a keyboard can be modeled via the quadratic
assignment problem [18, 120]. In this thesis, I show this approach to be
applicable to a range of text input methods that extends beyond keyboards. Hence, I broaden the definition of text input to consider the generation of symbols by any manual input actions. This allows formulating the more general design problem of finding the best assignment of symbols to any input actions, which can be modeled in terms of the linear or quadratic assignment problem, as presented in Chapter 2. The assignment problem captures the sequential nature of text input by taking into account the relationship between successive input events. This formulation enables me to draw on a large body of established optimization techniques for solving the (quadratic) assignment problem. Using exact or heuristic methods, I can efficiently search the large design spaces, trade off to balance among criteria, and obtain guarantees of the goodness of solutions.

C2: The challenge in using model-based interface optimization for designing text input methods is to articulate vague design goals such as “enable high performance and low fatigue” more formally, as quantifiable metrics that can be used during optimization. Text input involves complex perceptual, motor, and cognitive processes [134] that need to be accounted for when one wishes to design usable text input methods. Prior work focused mainly on optimization of movement performance instead, relying on heuristics of touch typing or using Fitts’ Law to quantify pointing time for soft keyboards. While task performance is an important factor, I argue that, if we wish to design usable text input methods, we must take into account more than the time it takes to move from one key to another. Therefore, in Chapter 4, I propose a set of novel objective functions that quantify important aspects of the typing process and criteria fundamental to UI design in general [38]. These cover the performance of the spectrum of input methods but also aspects of ergonomics and learnability – stress and fatigue, finger individuation, memorization potential, skill transfer, intuitiveness, and others. The objective functions are based in part on my empirical findings and partially on theories and observations presented in prior work on motor control, visual attention, and human memory. They also address criteria used in prior research on keyboard optimization, which I reformulate as assignment problems by using integer programming. This yields a coherent set of objective functions, which can be combined and solved by means of commercially available solvers and techniques for multi-objective optimization.

C3: In Chapter 3, I review prior work in the field of text entry optimization. I conclude that the state of the art is restricted to finding the best
spatial organization of letters in a grid. The optimized text input methods encompass only touch typing on physical keyboards, soft keyboards operated with one or two end-effectors, and ambiguous grouping keyboards. Objective criteria are largely limited to assessing motor performance in touch typing or pointing. Few have considered the similarity to an alphabetical arrangement or the Qwerty layout additionally or aimed to facilitate input recognition (e.g., for autocorrection). The present work expands the space of optimizable text input methods. In addition to performance, the objective functions proposed capture aspects of ergonomics and learnability also applicable to input methods other than the keyboard, such as mid-air gestures. With Chapter 5, I demonstrate this by presenting the optimization of three novel text input methods. They all combine several objective functions in a multi-objective optimization process to better capture several aspects of the interaction. This affords more realistic predictions of the usability of a system and, thereby, optimization for better text input methods. The first case entails optimizing the assignment of letters to the keys of a piano keyboard so as to allow pianists to exploit their existing motor skills and thus reduce the learning time. The transfer of skills from another domain to text input has not been considered before, and the piano keyboard has a form factor very different from the standard keyboard’s. With the second case, I consider the use of multi-finger gestures for text input in mid-air. I present objective functions that allow quantifying the movement speed, anatomical comfort, complexity, and memorability of such gestures, for automatically deriving an optimal set for text input. The optimization of mid-air gestures had never been considered in the literature. The third is a real-world case and is behind the new French keyboard standard. I optimize the assignment of a large set of special characters to the key slots of the physical keyboard, taking into account performance and ergonomic criteria but also the intuitiveness of the layout. This is different from assigning normal letters in that the frequency distributions vary greatly, depending on the input task, and the model needs to take into account the relations between special characters and normal letters. Prior research considered neither such large cases nor the input of special characters.

C4: Our current understanding of typing behavior is based largely on empirical studies of professionally trained touch typists that were conducted more than 30 years ago. Since then, the input device and typing behavior have changed dramatically: the physical form factor of modern
Introduction

keyboards is different from that of typewriters, the keyboard is used for considerably more tasks than entry of formal text, and many computer users have never received any formal training (whereas typing used to be done mostly by trained secretaries). Nevertheless, even recent work has optimized keyboard layouts with respect to models and heuristics based on these studies. In Chapter 6, I present significant findings from two empirical studies investigating the typing behavior of modern computer users. The first offers detailed analysis of hand and finger movements, using motion capture data, while the second study conveys a broader understanding of typing performance through analysis of a large dataset, from more than 168,000 computer users. The findings reveal that untrained typists develop various movement strategies, using anywhere from two to 10 fingers, that allow high-performance input comparable to that of people trained in the touch-typing system. Factors such as consistent finger usage, parallelization of movements, and global hand movement affect typing performance more than the number of fingers used for typing does. These findings advance our understanding of modern typing behavior and have allowed me to revise the optimization principles used in prior work to optimize keyboard layouts. Moreover, they represent an important step toward a general model of multi-finger typing.

1.5 The Structure of the Thesis

The work presented in this thesis is partially based on four publications, referred to in the text by the Roman numerals indicated at the beginning of the dissertation, along with one unpublished piece. As summarized in Table 1.1, each covers one or more of the following research themes: (1) gathering of empirical knowledge to enhance understanding of the user’s interaction with an input method, (2) quantitative modeling of important aspects of the interaction, and (3) design of a text input method by means of multi-objective optimization.

However, the thesis as a whole constitutes more than a summary of these publications, and the following chapters form a contribution in their own right. In the next chapter, I will give an introduction to using combinatorial optimization for the design of text input methods and also specify the general letter-assignment problem used throughout this thesis. Then, Chapter 3 provides a historical account of the most important work that has led to the current state of the art in text entry optimization. There, I
will also review the space of text input methods and objectives that optimization in prior work has addressed. The heart of the thesis is Chapter 4, in which I formulate the empirical design knowledge from both prior work and my studies as objective functions for optimizing text input methods. They cover aspects of performance, ergonomics, and learnability. Application of these functions is presented in Chapter 5. Proceeding from the work presented in publications I and II, along with the unpublished work mentioned above, I demonstrate the optimization of three text input methods that go beyond state-of-the-art text entry optimization. In Chapter 6, I then discuss the contributions of findings on typing behavior of modern computer users from two empirical studies, presented in publications III and IV. These have allowed me to revise the optimization principles used in the past for optimizing physical keyboard layouts and are the first step toward a general model of multi-finger typing. Finally, Chapter 7 provides a summary and discussion of the implications of my work, and it highlights future directions for research into text entry optimization.

<table>
<thead>
<tr>
<th>Publ.</th>
<th>Text input method</th>
<th>Research type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Multi-finger typing on an 88-key piano</td>
<td>Optimization</td>
</tr>
<tr>
<td>II</td>
<td>Mid-air chording gestures</td>
<td>Empirical study, modeling, optimization</td>
</tr>
<tr>
<td>III</td>
<td>Multi-finger typing on a physical keyboard</td>
<td>Empirical study</td>
</tr>
<tr>
<td>IV</td>
<td>Multi-finger typing on a physical keyboard</td>
<td>Empirical study</td>
</tr>
<tr>
<td>Unpubl.</td>
<td>Special-character input on a physical keyboard</td>
<td>Empirical study, modeling, optimization</td>
</tr>
</tbody>
</table>

Table 1.1. An overview of the published and unpublished work on which this thesis is based in part, the text input method studied, and the research type (empirical study, quantitative modeling of the user’s input behavior, or optimization of a text input method).
This chapter gives an introduction to the use of combinatorial optimization for the design of text input methods. It shows how we can generally model the problem of how to best assign symbols to input actions, presents a brief overview of the optimization techniques for solving it, and discusses its computational complexity.

Any design can be seen as a result of a set of decisions [122]. The goal for each decision in a user-centered design process is to maximize utility for the user and minimize the effort or cost of using the design. Traditionally, a designer makes and refines each decision in an iterative process based on design heuristics, creativity, prior experience, subjective feedback, and other factors [129]. However, this process typically permits the designer only to explore a few options and gives no guarantees as to the goodness of a solution, let alone of finding the best one.

Combinatorial optimization has been proposed as a rigorous and efficient supplement to this process [113, 114]. It uses computational methods for systematically exploring the design space to search for the best combination of decisions with respect to a given design objective. In other words, the optimization process algorithmically improves the design in light of the given knowledge about what makes a design good for a user. Combinatorial optimization requires explicitly defining the design problem, the design space, and the objective. It thereby offers an actionable formalism that exposes each of the assumptions and their tradeoffs. This, in turn, enables the usage of well-known algorithms for efficiently solving the problem.

The benefits of combinatorial optimization extend to a wide range of design problems in the human–computer interaction (HCI) field [114]. However, in this thesis, I focus on the optimization of text input methods, specifically the problem of finding the best assignment of symbols to input actions. In this application, we typically use so-called one-shot optimiza-
tion, wherein we define the design problem and then let the computer solve it. Advances in algorithms and hardware have made it possible to integrate optimization into the design process as an interactive tool or to automatically adapt interfaces [114]. Whatever the case, there are always three main challenges involved in using combinatorial optimization to solve a design problem [114]:

1. Mathematical definition of the design problem in terms of decision variables, objectives, and constraints.

2. Formulation of evaluative knowledge (models, heuristics, etc.) as objective functions.

3. Obtaining task-specific input data that parameterize the objective function.

While for many design problems in HCI it remains unclear how they can be modeled mathematically, finding the best keyboard layout was identified already in the 1970s, by operations researchers, as an instance of the assignment problem [18, 120]. However, this formulation has remained limited to the optimization of keyboard layouts [8, 60, 85, 66, 181]. In the previous chapter, I characterized the problem of designing a text input method more generally as that of finding the best assignment of symbols to input actions. I devote the remainder of this chapter to discussing the application of the assignment problem to this general formulation.

The formulation of evaluative knowledge in an objective function is essential to the success of the optimization approach and forms the core of the thesis project. The objective function encapsulates the design knowledge that predicts how well a user can interact with a given text input method and thus determines the goodness of the final solution. In traditional areas of application for optimization and operations research, objective functions often have their roots in the natural sciences and economics. In contrast, for applications in HCI, we need to predict the goodness of an interface from an end-user perspective. Among other elements, the objective function must capture psychological, behavioral, and/or social factors that influence the interaction with a system. Their mathematical formulation as part of an optimization problem is a key challenge in the domain of optimization of text input methods. In Chapter 4, I will formulate objective functions for various criteria and multiple text input methods within the scope of the assignment problem.
Designing a specific text input method hence requires collecting the input data necessary to instantiate the corresponding optimization problem with task-specific parameters. For the problems considered in this thesis, said data come from empirical studies and observations found in the literature. When instantiated, we can then use standard algorithmic or mathematical approaches to computationally arrive at the best solution. A brief overview of optimization methods is provided at the end of this chapter, and concrete optimization for three such instances is presented in Chapter 5.

In the work presented below, I use integer programming to define the assignment problem for text input design, as proposed by Oulasvirta and Karrenbauer [66, 114]. Integer programming offers a rigorous way to formulate the structure of a design task in terms of decision variables that take only integer or binary values. It enables tackling the optimization problem by means of powerful solvers that use exact methods. These apply a structured non-random search approach that guarantees finding the optimal solution in finite time and assures designers of the goodness of intermediary solutions with respect to the global optimum [66]. Moreover, integer programming offers a universal framework within which to define and compare design problems in HCI [114].

2.1 The Letter-Assignment Problem

The assignment problem, a fundamental problem class in combinatorial optimization, has been used to model and solve many real-world problems, such as the scheduling of work shifts, the placement of factories, and the layout of computer-internal wiring [17]. Operations researchers demonstrated already in the 1970s that the problem of finding the keyboard layout for a standard typewriter that allows the highest typing speed can be modeled as a (quadratic) assignment problem [18, 120]. Working from this observation, many researchers have proposed new keyboard layouts as a result of solving various instances of this assignment problem, using a range of exact or heuristic methods [8, 60, 66, 85, 181].

Introduced in the previous chapter, a similar problem is at the core of the design of many text input methods, not just the keyboard. Accordingly, I define the letter-assignment problem more generally, as follows:

*Given a set of $N$ symbols and $M$ input actions, what is the assignment of symbols to actions that minimizes the cost of text input?*
This formulation enables us to go beyond the optimization of typing speed on physical or soft keyboards to consider also other objective criteria and manual text input methods as defined in the beginning. Depending on the characteristics of the objectives, the letter-assignment problem can be modeled by means of a linear or a quadratic function, as I explain below. In general, to formulate a design problem mathematically, we need to define three aspects of it:

1. The decision variables that constitute the design or solution space.
2. The constraints that determine which combination of decisions forms a valid solution.
3. The objective function that determines how to evaluate the goodness of a solution in the design space.

These are specified in the description presented below.

2.1.1 The decision variables and design space

Any assignment of symbols to input actions can be defined by a combination of decision variables $x_{ik}$ that denote whether a symbol $i$ is assigned to an input action $k$. Formally, an assignment can be expressed as the design vector $x = (x_{11}, x_{12}, \ldots, x_{NM}) \in X$, where

$$x_{ik} \in \{0, 1\} \quad \forall i \in \{1, \ldots, N\}, j \in \{1, \ldots, M\}, \text{such that}$$

$$x_{ik} = \begin{cases} 1 & \text{if symbol } i \text{ is assigned to input action } k \\ 0 & \text{otherwise} \end{cases}$$

The set $X$ of all design vectors (that is, all combinations of decisions) denotes the design space.

2.1.2 Constraints

The design space, as defined above, contains many solutions that are not feasible, such as the trivial design wherein all decision variables get the value 0: no symbol is mapped to any input action. Constraints specify which designs are feasible and which cannot be accepted as solutions to the letter-assignment problem. Note that these constraints may vary with the concrete design problem – for example, they may change if there are more or fewer input actions than symbols. In any case, the letter-assignment problem requires two types of constraints:
Assign each symbol at least or exactly once: One constraint common to all letter-assignment problems is that each symbol must be assigned to at least one input action. This means that at least one of the decision variables $x_1, x_2, \ldots, x_M$ denoting the assignment of a symbol $i$ to any input action must be 1. Mathematically this can be formulated by requiring that the sum of these decision variables be greater than or equal to 1:

$$\sum_{k=1}^{M} x_{ik} \geq 1 \quad \forall i \in \{1, \ldots, N\}$$ (2.1)

If formulated in this manner, the constraint allows solutions with redundancy, wherein one symbol can be assigned to multiple input actions. If this is not desired, we must replace the $\geq$ with $=$ to enforce each symbol being assigned to exactly one input action.

Assign each input action once at most, or multiple times: While we want to ensure that each symbol is assigned an input action and can thus be entered, not all actions might need to be used for text input. Different constraints may apply, depending on the number of input actions in comparison to the number of symbols. If there are more input actions than symbols, we might want to ensure that an input action is mapped to no more than one symbol, so that no two symbols get mapped to the same input action. This can be formulated as follows:

$$\sum_{i=1}^{N} x_{ik} \leq 1 \quad \forall k \in \{1, \ldots, M\}$$ (2.2)

If the number of input actions and that of the symbols are the same, we can replace the $\leq$ with $=$ to ensure that each input action is assigned exactly once. Finally, if there are fewer input actions than symbols, one could replace the $\leq$ with $\geq$ to ensure that each input action is assigned at least once while still allowing the assignment of two symbols to the same input action (as with ambiguous keyboards).

2.1.3 The objective function

The formulation of the objective function models the structure of the design problem by defining how the design decisions influence the cost of the final design. The goal for the optimization is then to find the design that minimizes the cost. Note that maximizing a function $f(x)$ is the same as minimizing $-f(x)$. Hence, without loss of generality, we can assume that the objective function represents a cost that should be minimized. In general, we can use a linear or quadratic objective function to model the letter-assignment problem.
Background: Combinatorial Optimization for Designing Text Input Methods

**Linear (letter) assignment problem (LAP):** For the LAP case, we assume that each decision to assign a symbol $i$ to an input action $k$ ($x_{ik} = 1$) has cost $c_{ik}$. Then the goal is to minimize the sum of the costs for each decision. Mathematically, this is formulated thus:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} c_{ik} x_{ik}$$

(2.3)

**Quadratic (letter) assignment problem (QAP):** The cost of assigning a symbol $i$ to an input action $k$ might be dependent on the assignment of another symbol, $j$, to an input action, $l$ (for example, when capturing the time it takes to enter one symbol after another). In this case, the goal is to minimize the sum of the costs $c_{ikjl}$ of assigning $i$ to $k$ and $j$ to $l$, formulated as follows:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} c_{ikjl} x_{ik} x_{jl}$$

(2.4)

The additional modeling power given by the quadratic formulation comes at the expense of efficiency, as described below.

The objective functions as presented here are not operational, since they lack a definition of the cost $c$. In Chapter 4, I propose several variations of this objective function that, in addition, define the costs involved in the typing process that are related to motor performance, ergonomics, and learnability. Depending on the type of the cost, they must be formulated in terms of the linear or quadratic letter-assignment problem.

### 2.2 Optimization Methods and Complexity

The linear assignment problem can be solved to optimality in polynomial time, and efficient algorithms, such as the Hungarian method, were already proposed in the 1950s [75, 107]. In contrast, the quadratic assignment problem was shown to be NP-hard, and even approximate solutions (with provable distances to the optimal solution) cannot be found in polynomial time unless $P = NP$ [16, 132]. Already for instances with $N, M > 20$, it takes a long time to compute the optimal solution.

Over the last few decades, many researchers have proposed mathematical and algorithmic methods to find the global optimum or good approximate solutions in reasonable time. Many of these are implemented in commercial solvers and libraries (e.g., Gurobi,\(^1\) CPLEX,\(^2\) or the global optimization

---

\(^1\) See http://www.gurobi.com/.

toolbox of Matlab). Once a combinatorial optimization problem has been defined, they can be easily applied to solve it. In general, these techniques fall into two categories: exact and heuristic methods.

### 2.2.1 Exact optimization methods

Exact methods make use of mathematical properties of the design problem (convexity of the design space, linearity of the objective function, etc.) for subjecting it to numerical solving techniques. They guarantee finding the global optimum in finite time. However, this time may be exponential in the number of decisions, particularly for NP-hard problems such as the quadratic letter-assignment problem.

The simplest exact method is explicit enumeration, wherein each possible solution is evaluated and the current best solution (the incumbent) is updated. Better performance can be achieved through implicit enumeration, which makes use of relaxations. This means it optimizes “easier” problems which can be solved more efficiently. The relaxed solution sets a bound for the goodness of the original problem and provides guarantees as to the quality of incumbent solutions. This allows us to stop the optimization process if a sufficiently good solution is found (e.g., one within 1% of the global optimum). Many solvers use an implicit enumeration technique called Branch & Bound to solve integer programs. It systematically enumerates all solutions by considering subsets (branches) in the solution space defined by iteratively fixing the decision variables. Before the solutions of a branch are explicitly evaluated, the branch is checked against the upper and lower estimated bounds for optimal solutions. Assuming a minimization problem the current incumbent score is an upper bound of the optimization problem and solving the relaxation of the sub-problem denotes a lower bound for that part of the design space. If the lower bound of the subspace is worse than the incumbent, the whole subspace can be quickly discarded without a need for explicitly evaluating each solution.

The process relies on the estimation of strong upper and lower bounds, as well as efficient branching techniques. A common approach to find good lower bounds for the quadratic assignment problem is to linearize it and then use more efficient methods from linear integer programming. Accordingly, new decision variables and constraints replace the quadratic terms of the problem. This may pose a problem, however, if the linearization

---

requires a large number of new variables. Developing efficient linearization techniques is a topic attracting interest in the operations research and optimization community, as are methods for estimating good upper and lower bounds (see, for example, the reviews by Burkard et al. [16] and Loiola et al. [88]).

2.2.2 Heuristic optimization methods

Heuristic methods do not make any assumption about the mathematical structure of the design problem. Instead, they treat the objective function as a black box that can be used to evaluate any given solution. This makes the approach more flexible; the objective function does not need to be defined mathematically. We can, for example, use the output of a simulation model for the objective value. If the structure of the design problem is amenable to this, heuristic methods can be more suitable for quickly finding a good solution within reasonable time, making use of domain knowledge that might be hard to formulate mathematically. However, they generally cannot guarantee finding the global optimum in finite time, and they do not provide bounds on the goodness of a solution.

Most heuristic methods can be classed as either constructive or random search methods. Constructive methods use heuristic strategies to iteratively make decisions about the design parameters and thus construct a solution. An example is the greedy method, which in every iteration chooses the decision that maximally improves the design under the current conditions. Decisions are not revoked later, so an optimal solution might never be constructed. Random or local search methods select and evaluate solutions from the design space to iteratively improve the incumbent. In contrast to exact methods, they pick solutions randomly or by applying heuristics, and they are not guaranteed to cover the full design space in a finite time. Local search methods are attempts to find the local optimum in a given neighborhood. A good definition of the neighborhood is essential for finding a good solution but depends on the problem structure. For example, a common definition of the neighborhood for the assignment problem is any solution that can be reached from a given design by swapping \( n \) assignments (e.g., swapping 2 characters in a keyboard layout). This is called the \( n \)-opt neighborhood. For the quadratic assignment problem, it has been shown that even for just the 2-opt neighborhood, the local optimum cannot be found within polynomial time in the worst case [16, 135]. Moreover, the decision on whether a local optimum is the globally optimal solution
is NP-hard (see literature [116] cited by Burkard [16]). To overcome local optima, *meta-heuristics* exist to improve the local search and guide it to choose solutions from different neighborhoods or accept designs that are worse than the incumbent. Among example algorithms are simulated annealing, genetic algorithms, tabu search, and many others.

A more detailed review of exact and heuristic optimization approaches is given, for example, by Rao [122]; by Wolsey [172]; and, specifically for the assignment problem, by Burkard *et al.* [16] and by Loiola *et al.* [88].

### 2.2.3 Multi-objective optimization

The goodness of a design is generally determined by many factors, related to its usability, the experience of the user, the expectations of stakeholders, etc. In Chapter 4, I present a series of objective functions that capture various aspects of the interaction that have an impact on these factors, covering not only motor performance but also the ergonomics and learnability of a text input method and the recognition of the users’ input. To produce a good design, an optimization process needs to take into account many of these criteria. However, solving a *multi-objective* optimization problem is not straightforward. Typically, objectives conflict such that there is not a single globally optimal design but a set of so-called *Pareto-optimal* solutions – designs for which there are no superior solutions that would obtain a better score with regard to all objectives. These solutions are all optimal with respect to a certain combination or prioritization of objectives, and multi-objective optimization is about finding the best compromise.

There are many approaches to *multi-objective optimization*, a review of which is given by, for example, Marler and Arora [101]. Most methods belong to one of two groups, depending on the point at which designers can specify the importance of the objectives [101]. Methods with *a priori* articulation of preferences require the designer to define the importance of each criterion and then combine the objectives into one optimization problem accordingly. In contrast, methods with *a posteriori* articulation of preferences start with computing a set of Pareto-optimal solutions, which is then presented to the designer. This allows designers to inspect various designs before choosing the one that best reflects their preferences. *A posteriori* methods are computationally intensive, since they require solving many optimization problems. This is particularly challenging in cases of complex problems such as the quadratic assignment problem, wherein even approximate solutions cannot be computed efficiently, let
alone a Pareto-optimal set of solutions. Therefore, to solve the optimization problems presented in this thesis, I chose to use an a priori method – specifically, the weighted sum method.

The weighted sum method creates a single objective function $F^*(x)$ by summing the various objective values $F_i(x)$ of a design $x$ each weighted by its relative importance $w_i$:

$$F^*(x) = \sum_{i=1}^{O} w_i F_i(x) \quad (2.5)$$

where $O$ denotes the number of objective functions. The weights represent the priority that optimization for the respective objective should be given in comparison to others. Weights can, for example, be tuned in accordance with empirical observations or usage statistics for specific tasks or user groups, or they may represent strategic interests of stakeholders [114].

Typically, each objective is defined in terms of its own value range. To make them comparable when combined in a weighted sum, one needs to normalize them. Normalization can be performed by taking the (theoretical) minimum and maximum value, $F_{i\text{min}}, F_{i\text{max}}$, for each objective and then dividing the difference between the objective value $F_i(x)$ and the minimum by the difference between the maximum and the minimum:

$$\hat{F}_i(x) = \frac{F_i(x) - F_{i\text{min}}}{F_{i\text{max}} - F_{i\text{min}}} \quad (2.6)$$

Each objective is thus rendered a dimensionless value ranging from 0 to 1, thereby ensuring that scores are comparable. Maximum and minimum scores can be obtained by maximizing and minimizing the problem for each objective separately. If the complexity of the problem renders this infeasible, estimates must be made by analyzing the problem structure or using approximation methods. In the optimization of input methods presented in Chapter 5, objectives are assumed to be normalized and the explicit formulation is omitted, for better readability.

A priori methods in general and the weighted sum approach in particular have several recognized shortcomings [101]. For example, the preferences of stakeholders might be vague or incompletely represented, and they may change during the process. This makes it hard to set definite weights. Also, the normalization approach does not take into account the distributions of the scores. If these differ between objectives, the process can be biased. These problems notwithstanding, the weighted sum method is a popular approach, because it is easy to implement and allows intuitive expression of the tradeoffs between objectives. Most importantly, it does not decrease
performance in comparison to solving a single-objective optimization problem. To some extent, this approach can be used also to explore the set of Pareto-optimal solutions by iteratively adapting the weights and re-solving the problem. This is done, for example, in Section 5.3. Exploring the use of other multi-objective optimization methods is not within the scope of this thesis and is left for future work.

2.3 Modeling Limitations with the Assignment Problem

The letter-assignment problem, as presented above, directly captures the structure of many design problems in text entry, and the following chapters will show that it can be used to formulate the problem of assigning symbols to input actions with respect to various objectives and input methods. However, certain objective criteria used in prior work cannot be expressed through formulation of an assignment problem. The linear formulation presented above requires conditions in which the cost of an assignment depends only on the individual costs of assigning one symbol to one input action. More modeling power is obtained by using a quadratic formulation to take into account the conditional cost of assigning a symbol to an input action given that another symbol has already been assigned. Neither formulation allows us to account for costs related to the assignment as a whole or that depend on multiple symbol–action pairs. Moreover, as was discussed in Chapter 1, criteria that necessitate a computer simulation and that cannot be expressed in a closed mathematical form cannot be optimized via exact mathematical optimization methods.

There are a few examples from prior work in which these limitations apply. In those cases, formulation of the design problem at hand in terms of letter-assignment problem is impossible and exact methods could not be used to solve the problem. Instead, the authors employed heuristic methods to find good solutions.

Oulasvirta et al. proposed a performance model of two-thumb typing [115] on the basis of which they optimized a soft keyboard. The model captures the performance benefit of pressing keys with alternating thumbs, which depends on the $n$ keys pressed previously before alternation to the other thumb. This cannot be modeled by means of the letter-assignment problem since the cost of an assignment depends on $n$ other assignments.

Another example is the optimization of gesture-typing keyboards [74]. In this case, the user does not press a key to enter a single letter but
continuously traces from one letter key to the next to enter a full word. Here, the goal is to find an assignment of letters to keys on a soft keyboard such that the resulting word gestures are optimal with respect to a certain objective. For example, Smith et al. [150] optimized a keyboard for gesture clarity. This refers to how unique a gesture trace is and thus whether the corresponding word can be correctly recognized by the system or instead is easily confused with other words. This cost cannot be modeled in terms of the letter-assignment problem because it requires comparing the word gestures resulting from the full assignment. The authors instead used a simulation procedure to obtain the respective costs and applied heuristic methods to randomly generate and evaluate designs.

In general, for modeling such costs as those illustrated above, more complex formulations are required, and solving by means of implicit enumeration techniques could be difficult. In such cases, heuristic methods, as used in prior work, are more suitable, since they do not make assumptions as to any mathematical properties of the objective function. Design problems of this kind are not considered in this thesis.

2.4 Summary

In this chapter, I have introduced combinatorial optimization as a rigorous and efficient process for the design of text input methods. I presented the linear and quadratic letter-assignment problem as a mathematical model for the problem of how to best assign symbols to input actions. Any solution can be viewed as a combination of (design) decisions, denoted by binary decision variables that capture whether a symbol is assigned to an input action or not. The constraints define the set of feasible solutions – that is, specify which decisions constitute a valid assignment. The objective function associates a cost with each decision and captures the structure of the design problem. It thereby allows assessing the goodness of a solution.

My formulation of the letter-assignment problem as an integer program allows application of exact methods. These offer a guarantee of finding the global optimum in a finite (but potentially exponential) time and provide bounds for the goodness of incumbent solutions. In the following chapters, I show that a wide range of objective criteria, for various input methods, can be formulated by means of the letter-assignment problem and that this expands the space of optimizable text input methods.
3. Related Work: The History and State of the Art in Text Entry Optimization

This chapter provides a historical overview of the achievements in text entry research that led to the establishment of the model-based optimization approach used today in designing text input methods. I review the methods and objectives applied in prior work and discuss the corresponding space of optimizable text input methods.

Soon after the Qwerty keyboard layout entered the market, in the 1870s, researchers began to question the suitability of this layout for typing English text [111]. They observed that the load on the hands and fingers is not ideal with respect to their strength. The layout overloads the left hand and the little fingers, only a few words can be typed with the fastest row (the middle one) alone, and fingers of the same hand frequently have to switch row to type successive keys [30].

Note that these observations were made with respect to the touch-typing system, wherein a fixed finger-to-key mapping assigns one or two columns of the keyboard layout to each of the fingers. The thumb is used to press the space bar, and the other eight fingers rest on the middle row, with the index fingers resting on the f and the j key. This system was established after the introduction of the typewriter, in the 1880s [174], and has been taught to the present day as the only “proper” way to type on a keyboard.

Since those early days, many researchers have worked on improving the keyboard layout to allow for more efficient text input when the touch-typing system is used. Their approach developed from manual rearrangement of letters in accordance with empirical observations of the user’s movements to what we now call model-based interface optimization [114]. There have been several milestones in this development, which will be further described in the discussion below:

1. Accounting for statistical distributions of the target language.

2. Considering that the time to type a letter is not constant but depends on the preceding key press (in a quadratic relationship).
3. Mathematically formulating the design problem as a quadratic assignment problem.


5. Expressing psychological models as objective functions

6. Performing optimization with respect to multiple objectives.

Table 3.1 provides an overview of the most important work that led to these milestones, which are considered in the following discussion. The table is arranged chronologically and presents the optimized input device and input actions considered by each work. Moreover, it indicates whether the authors formulated the problem as a quadratic or a linear assignment problem, their objectives, and any optimization methods they employed. Note that I refrain from directly comparing the methods in terms of improvement relative to a baseline text input method. Such a comparison would be misleading, since each report has presented the improvements with respect to its own set of objective criteria and input data. Instead, possible improvements for different input methods are discussed at the end of this chapter.

### 3.1 Manual Design of Physical Keyboards

Early in the 20th century, researchers proposed manually improved keyboard layouts based on the statistical distribution of letters and letter pairs (bigrams) in English (see for example the overview by Noyes [111]). However, they lacked a principled understanding of the motor behavior. Therefore, designs were mutually contradictory in that some placed frequently used letters under the stronger fingers (the index fingers) whereas others placed the least used letters there [111].

In the 1930s, August Dvorak, William Dealey, and colleagues conducted thorough studies of the hand and finger movements during touch typing with the Qwerty layout [30]. They observed that typing is a serial process of successive keystrokes wherein the time it takes to type a letter depends on the one preceding it. The interval between key presses varies, depending on the spatial arrangement of the corresponding keys and the resulting use of the fingers under the touch-typing system. These are among the factors they found [30, 29]:

- Hand alternation is faster than bigrams typed with fingers of the same hand.
• Bigrams within the middle (“home”) row are faster than those typed within other rows.

• Bigrams typed by adjacent fingers are slower than those typed by fingers that are further apart.

• “Hurdles” – letters found in different rows and typed by fingers of the same hand – are slow.

• Also slow are “reaches,” in which the next letter requires the finger to move across columns or rows.

Dvorak and colleagues manually rearranged the letters in order to “reduce the frequency of such awkward sequences” [29], on the basis of their behavioral observations and a statistical analysis of the English language. The resulting keyboard layout was never adopted by a larger population, although it is still available for modern computers. Possible reasons for the failure of the Dvorak Simplified Keyboard (DSK) might be the economic cost of changing hardware, the personal cost of learning a new keyboard layout [111], and insufficient gains. Indeed, the superiority of the DSK over the Qwerty layout is disputed in the literature [72, 84, 111]. Although several experimental studies have been conducted, by Dvorak and others, that compare the two keyboard layouts, clear quantitative results remain absent [111]. Nevertheless, the DSK is noteworthy for its early example of a systematic design process wherein decisions are governed by empirical observations and statistical analysis.

Many alternative keyboard layouts followed the DSK, using a similar approach of manually rearranging letters in line with empirical observations of hand and finger movements and frequency analysis of the language to be typed [51, 102, 111, 120]. Common to the associated studies is the observation that the time it takes to enter a symbol is not constant but depends on the preceding symbol and the movements it requires.

### 3.2 Model-based Optimization of Physical Keyboards

In 1974, Pollatschek and colleagues [120] became the first to mathematically formulate the design problem as that of finding a permutation of letters/keys that minimizes the time to move between two successive keys, weighted by the frequency of the corresponding bigram – a quadratic assignment problem. Recognizing its “intractability” [120], they did not
attempt to solve the problem mathematically but instead used it to quantitatively evaluate “promising” layouts created through heuristics [120].

Three years later, Burkard and Offermann [18] pioneered the use of computational optimization methods to solve the letter-assignment problem as formulated by Pollatschek et al. Recognizing that larger instances of the quadratic assignment problem cannot be solved to optimality, they used a combination of Branch & Bound and local search to compute good approximate solutions. They proposed optimized keyboard layouts for German, French, English, and Dutch and computed a tradeoff keyboard that jointly maximizes typing performance for German, French, and English, yielding improvements in the objective value of 8–10% over the Qwerty keyboard.

While Burkard and Offermann’s work relied on preliminary models created as estimations by Pollatschek, their work in text entry optimization was groundbreaking: they used an approach to user-interface design that is now known as model-based interface optimization [114]. In this approach, quantitative models and an explicit mathematical formulation of the design problem and space are used for systematically evaluating solutions and finding the best design. Note that this breakthrough came several years before the seminal work by Card et al. [19], who in 1983 proposed using cognitive models to simulate the user’s actions and on that basis predict the consequences of particular design choices.

In the decades that followed, two streams of work advanced the optimization of text input methods: (1) empirical study of the perceptual, motor, and cognitive processes during typing and (2) the application of advanced optimization methods to better approximate solutions.

On one hand, researchers set out to understand touch-typing performance more fully. Most of them empirically observed phenomena related to motor and cognitive processes during typing, further discussed in Chapter 6. Only a few quantitative models have captured the time to press successive keys as a function of the fingers used for typing and the location on the keyboard. These were developed on the basis of data from a small number of professionally trained touch typists [57, 72], so their generalizability is questionable. Other researchers directly gathered experimental data on the transition times for every possible key pair and optimized the keyboard layouts in terms of these [18, 60, 72, 85].

Some researchers, on the other hand, have applied more advanced optimization techniques, for better approximation of solutions to the letter-assignment problem. They proposed the use of heuristic methods such
as simulated annealing [85], ant colony optimization [31, 163], genetic algorithms [9, 26, 43, 60, 70, 96], and particle swarm optimization [177]. These do not make any assumptions about the mathematical structure of the given problem and do not require explicit formulation as a quadratic or linear assignment problem. This allows more readily optimizing a keyboard layout on the basis of empirical observations, as opposed to formulation as closed mathematical expressions. However, as discussed in the previous chapter, heuristic methods do not guarantee finding the global optimum and provide no estimate of the goodness of approximate solutions. Only recently, Karrenbauer and Oulasvirta [66] proposed the use of integer programming to solve the letter-assignment problem. Being the first to follow on from the work by Burkard et al., they made use of exact methods that guarantee searching the full design space and setting bounds for the goodness of incumbent solutions. The greater computing power available today and advanced methods for implicit enumeration and linearization of the problem have enabled finding very good approximate solutions with this approach for larger instances (e.g., 33 keys and letters) that are provably within 1% of the global optimum after only seven minutes of computation [66]. The layout was predicted to allow input about 7% faster than with the Qwerty layout.

### 3.3 Optimization of Soft Keyboards

From the 1990s onward, the focus of text entry research has been shifting from physical keyboards to mobile input. The problem of designing a soft keyboard layout is a more approachable one: novel layouts can be more easily implemented, tested, and distributed, and interaction with only one end-effector (e.g., an index finger or stylus) is easier to understand and model. A milestone in model-based interface optimization was reached when MacKenzie [93] and later Zhai [180, 182, 181] established Fitts’ Law [36] with weighting by bigram frequency as the objective function for the quadratic letter-assignment problem. While only modeling the distance between keys would suffice for evaluating the performance achievable with a soft keyboard [80], using the Fitts’ Digraph model allows direct estimation of the achievable typing performance in WPM. It predicts that performance improvements of nearly 30% could be achieved by rearranging the letters of the soft keyboard, a benefit much larger than for the physical keyboard layout [66]. However, empirical evaluation of this benefit faces
difficulties due to the long learning times and users' familiarity with the Qwerty layout.

After Fitts' Law was established for the performance measurement, several optimized layouts were obtained for English and other languages [7, 14, 25, 58, 83]. Furthermore, text entry optimization has recently turned to optimizing soft keyboards for multiple objectives. To improve learnability, Zhai et al. attempted to retain an alphabetical ordering [182, 181] while optimizing performance. They used a weighted sum approach to jointly optimize for both objectives. Similarly, Dunlop and Levine sought the best tradeoff between input performance and similarity to the Qwerty layout [27]. In addition, they optimized the arrangement of keys for improved accuracy of autocorrection algorithms by avoiding adjacent placement of letters that when replaced still produce a valid English word. Instead of combining the three objectives in one function, they computed the Pareto-optimal set of designs. The solutions in this set cannot be improved with respect to any objective without worsening the score for other objectives. Computing this set allows exploring the range of optimal tradeoffs without making a priori assumptions about the importance of each objective.

Optimization methods have been applied more recently to optimize the design of soft keyboards when operated by other input actions than pointing and selection with one end-effector. Building on prior work [21, 91], Oulasvirta et al. [115] developed a performance model of two-thumb text input to evaluate and optimize the input performance of soft keyboards when operated with two end-effectors that can be used in alternating fashion. Rick [127] and, later, Smith et al. [150] and Bi et al. [15] optimized a soft keyboard for input by word gestures, where one end-effector performs a continuous drawing gesture over the letters that form a word. The layout has been optimized to improve input performance [127] but also recognition of the gestures, while retaining a similarity to Qwerty [15, 150]. These researchers did not attempt to model the design problem mathematically but used heuristic optimization methods to generate and evaluate layouts algorithmically and approximate a good solution. In fact, as is discussed in Section 2.3, the characteristics of the objective functions used in their work mean that they could not perform the modeling as a letter-assignment problem, and no other formulation has been proposed.

To the best of my knowledge, the only other text input devices that have been designed by means of optimization methods are grouping keyboards. Here, multiple letters are assigned to the same key in order to reduce the
number of keys necessary for typing [80, 46, 81] (e.g., using a 12-button telephone keypad). With this device type, the input action does not unambiguously determine the symbol; a disambiguation algorithm is used to predict the most likely letter sequence on the basis of a language model. The assignment of letters to keys has been optimized to maximize the accuracy of the disambiguation algorithm. Depending on the characteristics of the objective function, the problem can be formulated as an instance of the quadratic letter-assignment problem, as shown in Subsection 4.4.2.

3.4 Summary

The work discussed above has pioneered model-based user-interface optimization. Systematic design approaches based on statistics of the English language and empirical observations of typing performance have been employed by Dvorak and others ever since the early 20th century. In modeling the design problem as a quadratic assignment problem, Pollatschek and Burkard paved the way for the use of mathematical optimization methods already in the 1970s. With the establishment of the Fitts’ Digraph model in 2000, model-based optimization became a standard method in text entry research. Since then, model-based interface optimization has progressed significantly also in other areas [114]. A series of design problems can be mathematically expressed, and psychological, behavioral, and aesthetic objectives have been formulated as objective functions. These developments have advanced the optimization of a wide range of interfaces and applications, from menu systems and widget layouts to information visualization and design of web pages (the literature [113] offers an overview).

3.4.1 The space of optimizable text entry methods

While model-based interface optimization has advanced in recent years, the space of optimizable text entry methods has remained limited. Table 3.1 presents an overview of the key work on text entry optimization that I have discussed. It clearly shows that the optimization of text input methods has been restricted to finding the best spatial organization of letters in a grid. The input methods optimized thus far have included only the following:

1. Touch typing on physical keyboards.
2. Soft keyboards operated with one end-effector (pointing or gesture typing) or both thumbs.
In the case of the physical keyboard, theoretical performance improvements of 7–10% over Qwerty have been reported [18, 66, 85] with use of the touch-typing system. Larger improvements, of 28–50% [66, 181], have been reported for optimized layouts in cases involving only one end-effector being employed to press the keys. The performance gains are high even when tradeoffs are made for taking into account multiple languages (24% [14]) or improving aspects of usability (11–12% [13, 27]).

Although it is well known that text input is a complex skill involving cognitive, motor, and perceptual processes [134], that very complexity renders it difficult to express the psychological theories and observations as objective functions in an optimization process. Therefore, research into text entry optimization has been limited. As Table 3.1 illustrates, it has considered only a few objectives in the optimization of keyboard layouts:

1. **Motor performance** – the time it takes to perform one input action after another.

2. **Similarity to an alphabetical arrangement or the Qwerty layout**, to decrease the learning time.

3. **Recognition of input actions** by rearranging the keys so as to facilitate disambiguation or autocorrection through language models.

The present work expands the space of optimizable text entry methods beyond keyboard layouts and demonstrates how to express novel objective functions for evaluating the performance, ergonomics, and learnability of various input methods in line with models, theories, and empirical observations covering motor control, memory, visual search, and more.
<table>
<thead>
<tr>
<th>Authors</th>
<th>Ref.</th>
<th>Year</th>
<th>Input device</th>
<th>Input action</th>
<th>Math. formula</th>
<th>Objective(s)</th>
<th>Optimization method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dvorak et al.</td>
<td>[29]</td>
<td>1936</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>--</td>
<td>Movement heuristics</td>
<td>--</td>
</tr>
<tr>
<td>Pollatschek et al.</td>
<td>[120]</td>
<td>1974</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>QAP</td>
<td>Movement time</td>
<td>--</td>
</tr>
<tr>
<td>Burkard</td>
<td>[18]</td>
<td>1977</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>QAP</td>
<td>Movement time</td>
<td>Branch and bound</td>
</tr>
<tr>
<td>Glover</td>
<td>[43]</td>
<td>1987</td>
<td>Ambiguous keyboard</td>
<td>--</td>
<td>Disambiguation</td>
<td>Movement time</td>
<td>Genetic algorithm</td>
</tr>
<tr>
<td>Light &amp; Andersson</td>
<td>[85]</td>
<td>1993</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>QAP</td>
<td>Disambiguation</td>
<td>N-opt local search</td>
</tr>
<tr>
<td>Lesher et al.</td>
<td>[80]</td>
<td>1998</td>
<td>Ambiguous keyboard</td>
<td>--</td>
<td>Disambiguation</td>
<td>Movement time</td>
<td>N-opt local search</td>
</tr>
<tr>
<td>Glover</td>
<td>[43]</td>
<td>1987</td>
<td>Ambiguous keyboard</td>
<td>--</td>
<td>Disambiguation</td>
<td>Movement time</td>
<td>Genetic algorithm</td>
</tr>
<tr>
<td>Light &amp; Andersson</td>
<td>[85]</td>
<td>1993</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>QAP</td>
<td>Movement time</td>
<td>Genetic algorithm</td>
</tr>
<tr>
<td>Lewis et al.</td>
<td>[82]</td>
<td>1999</td>
<td>Soft keyboard</td>
<td>One end-effector</td>
<td>QAP</td>
<td>Movement distance</td>
<td>--</td>
</tr>
<tr>
<td>Zhai, Hunter &amp; Smith</td>
<td>[180]</td>
<td>2000</td>
<td>Soft keyboard</td>
<td>One end-effector</td>
<td>QAP</td>
<td>Movement time</td>
<td>Metropolis algorithm</td>
</tr>
<tr>
<td>Zhai, Hunter &amp; Smith</td>
<td>[181]</td>
<td>2002</td>
<td>Soft keyboard</td>
<td>One end-effector</td>
<td>QAP + LAP</td>
<td>Movement time, alphabetical similarity</td>
<td>Multi-start local search</td>
</tr>
<tr>
<td>Eggers et al.</td>
<td>[31, 163]</td>
<td>2003</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>--</td>
<td>Movement heuristics</td>
<td>Ant colony optimization</td>
</tr>
<tr>
<td>Deshwal &amp; Deb</td>
<td>[26]</td>
<td>2003</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>--</td>
<td>Movement heuristics</td>
<td>Genetic algorithm</td>
</tr>
<tr>
<td>Goettl et al.</td>
<td>[44]</td>
<td>2005</td>
<td>Physical keyboard</td>
<td>Touch typing</td>
<td>--</td>
<td>Movement heuristics</td>
<td>Genetic algorithm</td>
</tr>
<tr>
<td>Li et al.</td>
<td>[83]</td>
<td>2006</td>
<td>Soft keyboard</td>
<td>One end-effector</td>
<td>QAP</td>
<td>Movement time</td>
<td>Simulated annealing</td>
</tr>
<tr>
<td>Rick</td>
<td>[127]</td>
<td>2010</td>
<td>Soft keyboard</td>
<td>Gestures</td>
<td>--</td>
<td>Movement time</td>
<td>Simulated annealing, hill climbing</td>
</tr>
<tr>
<td>Yin &amp; Su</td>
<td>[177]</td>
<td>2011</td>
<td>Physical keyboard and</td>
<td>Touch typing</td>
<td>--</td>
<td>Movement heuristics</td>
<td>Particle swarm optimization</td>
</tr>
<tr>
<td>Oulasvirta et al.</td>
<td>[177]</td>
<td>2012</td>
<td>Soft keyboard</td>
<td>One end-effector</td>
<td>QAP + LAP</td>
<td>Movement distance, Qwerty similarity, disambiguation</td>
<td>Multi-start local search</td>
</tr>
<tr>
<td>Karrenbauer &amp;</td>
<td>[115]</td>
<td>2013</td>
<td>Soft keyboard</td>
<td>Two thumbs</td>
<td>--</td>
<td>Movement time</td>
<td>Simulated annealing, hill climbing</td>
</tr>
<tr>
<td>Oulasvirta</td>
<td>[66]</td>
<td>2014</td>
<td>Soft keyboard</td>
<td>Touch typing and one</td>
<td>QAP</td>
<td>Movement time</td>
<td>Branch and bound</td>
</tr>
<tr>
<td>Smith, Bi &amp; Zhai</td>
<td>[150]</td>
<td>2015</td>
<td>Soft keyboard</td>
<td>Gestures</td>
<td>--</td>
<td>Movement time, disambiguation</td>
<td>Simulated annealing, Qwerty similarity</td>
</tr>
</tbody>
</table>

**Table 3.1.** A chronological overview of important earlier work using computational approaches to optimization of text input methods. The table presents the input method optimized, specifies whether the paper mathematically formulates the design problem as a linear or quadratic assignment problem (LAP/QAP), and compares the objectives and optimization methods. Note that the table is not exhaustive but, rather, points to example papers as milestones in text entry optimization.
4. Objective Functions for the Letter-Assignment Problem

In this chapter, I formulate objective functions for optimizing the assignment of symbols to input actions. The objectives can be divided into three categories of elements imposing bounds on the users’ interaction with a system: motor performance, ergonomics, and learnability. In addition, I present objectives related to the uncertainty of the user’s input, which limits the interaction on the system’s side.

Most of the objective functions presented below are an original contribution that I developed on the basis of empirical evidence collected either in my own work or as observations from prior work. They constitute the main contribution of this thesis. These go beyond state-of-the-art text entry optimization by modeling new interaction techniques and objectives. In addition, I review previously proposed models and empirical studies and show how their results can be formulated within the same framework of the assignment problem. Note that this chapter presents the general formulation of these objectives, not concrete instances. Three example cases are presented further on, in Chapter 5.

In my earlier work [34], I identified eight categories of objective criteria for user-interface optimization by reviewing guidelines for UI design and usability engineering [38]. They apply similarly to the optimization of text input methods and are restated in Table 4.1. While this categorization makes no claim of being complete, it emphasizes the broad range of factors affecting the design of user interfaces in general and, more particularly, that of text input methods. The table also draws attention to the categories of objective functions presented in this chapter, specifying whether they are my original contribution and applied to the relevant optimization cases presented in Chapter 5 or instead they were proposed in prior work and are reformulated here in a letter-assignment problem framework. I thereby show how this thesis both advances the field of text entry optimization and
points to directions for future work.

A good text input method allows the user to enter text as quickly and effortlessly as possible and thereby focus better on the higher-level task, such as communication, information retrieval, or programming. The performance achievable with a device is determined in large part by the speed and accuracy with which users can move their hands and arms to operate the device, or motor performance. Ensuring that movements are performed in an ergonomic manner is important if the user is to be able to maintain a high input speed over an extended time.

While performance and ergonomics are important for prolonged (expert) use, learnability is an element especially important for intuitiveness and ease of use by novice users. Text input methods that offer superior performance and ergonomics but are hard to learn (or hard to relearn when the user switches from Qwerty) have repeatedly failed to be adopted by a larger user base (e.g., the Dvorak Simplified Keyboard [30, 29], stenotype machines [23], or the Twiddler one-handed keyboard [89]). In contrast, intuitive systems that build on the experience with Qwerty have been adopted quickly, with one example being gesture typing on soft keyboards [74]. Walk-up usability is particularly important for text input methods that are not used for an extended time, such as those for public displays, within games, in cars, etc.

These user-centered objectives are supplemented by a set of objectives associated with the recognition of the users’ input. It is essential for the usability of a system and the users’ satisfaction that the text input method be able to reliably recognize the actions performed by the user. The design of the system may affect recognition in two ways. Firstly, ambiguous text input methods require the user to perform the same input action to enter two distinct symbols [92]. The assignment of symbols to input actions can influence the success of the disambiguation algorithm used to ascertain the symbol most likely to have been intended. Secondly, noise in the users’ movements or sensor noise affects the recognition of input actions. The design can facilitate the use of intelligent recognition methods, such as the classification of input actions or autocorrection.

As will be explicated below, most of the objectives are quantified in relation to the text input task. The structure of the language used in the typing is an important factor in determining the assignment of symbols to input actions. Most of the objectives give priority to the assignment of frequent symbols or symbol pairs (bigrams). For example, assigning
Objective Functions for the Letter-Assignment Problem

<table>
<thead>
<tr>
<th>Objective</th>
<th>Reference</th>
<th>Applied in . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong> (movement performance, visual search time, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One-finger pointing</td>
<td>[13, 14, 36, 93, 180, 181]</td>
<td>–</td>
</tr>
<tr>
<td>Movement distance</td>
<td>[27, 82]</td>
<td>Publication I, see 5.1</td>
</tr>
<tr>
<td>Chording performance</td>
<td>–</td>
<td>Publication II, see 5.2</td>
</tr>
<tr>
<td>Touch-typing performance (heuristics)</td>
<td>[29, 30, 31, 57, 163]</td>
<td>–</td>
</tr>
<tr>
<td>Touch-typing performance (movement time)</td>
<td>[18, 60, 72, 120]</td>
<td>–</td>
</tr>
<tr>
<td>Multi-finger performance for special characters</td>
<td></td>
<td>Unpublished, see 5.3</td>
</tr>
<tr>
<td><strong>Ergonomics</strong> (probability of injury, energy expenditure, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strain</td>
<td></td>
<td>Unpublished, see 5.3</td>
</tr>
<tr>
<td>Muscle fatigue</td>
<td>[56]</td>
<td>–</td>
</tr>
<tr>
<td>Finger individuation</td>
<td>–</td>
<td>Publication II, see 5.2</td>
</tr>
<tr>
<td><strong>Learnability</strong> (walk-up usability, memorability, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complexity</td>
<td>–</td>
<td>Publication II, see 5.2</td>
</tr>
<tr>
<td>Skill transfer</td>
<td>–</td>
<td>Publication I, see 5.1</td>
</tr>
<tr>
<td>Memorability</td>
<td>–</td>
<td>Publication II, see 5.2</td>
</tr>
<tr>
<td>Intuitiveness</td>
<td>–</td>
<td>Unpublished, see 5.3</td>
</tr>
<tr>
<td>Familiarity</td>
<td>[27]</td>
<td>Unpublished, see 5.3</td>
</tr>
<tr>
<td><strong>Input recognition</strong> (error prevention, recognition accuracy, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autocorrection</td>
<td>[27, 28]</td>
<td>–</td>
</tr>
<tr>
<td>Input uncertainty</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Disambiguation</td>
<td>[80]</td>
<td>–</td>
</tr>
<tr>
<td><strong>Mental workload</strong> (consistency, simplicity, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Accessibility</strong> (perceptibility, operability, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>User experience</strong> (fun, aesthetics, etc.)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1. An overview of the objective categories for optimization of text input methods. The criteria for which I present an objective function in this chapter are further expanded. These are based on prior work (see the middle column) or my own contribution and are applied to the design of novel text-input methods presented in Chapter 5 (see the rightmost column).
frequent bigrams to fast input actions is necessary for achieving overall high performance. Hence, it is important to obtain appropriate language data, representing the target language but also the tasks’ characteristics. If several tasks are involved, different corpora can be combined within a weighted sum during preprocessing of the input data. The weights chosen determine how well the optimized design supports the given task. For example, Bi et al. [14] have optimized a soft keyboard for five individual languages, proceeding from statistical distributions from several corpora. The outcome is a tradeoff that best supports all languages considered. In contrast, when optimizing soft keyboards for English, Zhai et al. [180] argued that the differences among various styles of English (chat room vs. news articles vs. literary works) are quite small. Therefore, they used statistics from only one standard corpus. For input methods other than soft keyboards, though, even small differences could have a large impact on what constitutes the optimal design. In Chapter 5, I will discuss the collection of language statistics specific to my particular example cases and discuss their role in the optimization process.

Most if not all of the prior work in the field of text entry optimization has focused on at least one of the criteria mentioned above. However, the optimization has been largely confined to considering the performance of keyboard layouts (see Chapter 3). The objectives presented below allow optimization of text input methods based on other sensing technologies and form factors, and the formulation encompasses advanced criteria that capture the complex process of text input.

### 4.1 Motor Performance

For most users, text input is a subtask of work toward a higher-level goal, such as communication or information retrieval. A high-usability text input method allows the user to perform this subtask in as quick and effortless a manner as possible. The (expert) performance achievable with a device is determined largely by the speed and accuracy with which a user can move the hands and arms to operate the device: motor performance.

Generally, we can formulate this objective by using the quadratic formulation of the assignment problem. The to-be-minimized cost function is the time it takes to perform one input action after another, weighted by the frequency of the corresponding symbols that are assigned to these input actions (the bigram frequency). Following Equation 2.4, we can formulate
this as a minimization problem:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} t_{kl} x_{ik} x_{jl}$$

(4.1)

where $x_{ik}, x_{jl}$ are the binary decision variables denoting that a symbol $i (j)$ is mapped to an input action $k (l)$, $p_{ij}$ is the frequency of the letter pair $ij$, and $t_{kl}$ is the time it takes to perform the input action $l$ after performing the action $k$.

In this section, I will discuss various ways to assess this time, depending on the input strategy and design task. Table 4.3, at the end of the section, reiterates the proposed objective functions in summary form.

### 4.1.1 One-finger input

The state of the art in text entry optimization has focused on maximizing the performance of soft keyboards operated with one end-effector [13, 14, 27, 81, 93, 180]. Fitts’ Law [36] has become well established for measuring the time it takes to point from one key to another and can be used to estimate the upper limit of performance achievable with a given keyboard layout. Under Fitts’ Law, the time it takes to move to a target key depends on the size and distance of the target. This can be formulated thus [90]:

$$t_{kl} = a + b \log_2 \left( \frac{D_{kl}}{W_l} + 1 \right)$$

(4.2)

where $a$ and $b$ are empirically determined parameters, $D_{kl}$ is the distance between initial key $k$ and target key $l$, and $W_l$ is the width of the target key. After instantiating the parameters $a$ and $b$, we can directly use this equation to obtain the movement time for Equation 4.1. This yields the following objective function:

$$\min a + b \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} \log_2 \left( \frac{D_{kl}}{W_l} + 1 \right) x_{ik} x_{jl}$$

(4.3)

This formulation is also known as the Fitts’ Digraph model [180, 181]. It allows directly quantifying an upper bound to the movement time and thereby the performance achievable with a soft keyboard in characters per second (CPS) or words per minute.

If all of the keyboard’s keys are of the same size, the time to move between two keys, as quantified via Fitts’ Law, depends on only the distance between the keys. Hence, an alternative objective could be to minimize the distance between frequent characters. This can be formulated in the following way:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} D_{kl} x_{ik} x_{jl}$$

(4.4)
This simpler formulation is used by, for example, Dunlop et al. [27] and Lewis et al. [82]. Note that it yields a different optimal solution than Equation 4.3 does.

Fitts’ Law provides a general model for the performance of pointing movements with one end-effector. The objective function presented with Equation 4.3 is not limited to soft keyboards operated with one finger – it can be used to optimize other methods of text input operated via pointing movements with one end-effector. For accurate estimation of the input performance, the parameters must be instantiated accordingly.

4.1.2 Chord input

In this subsection, I propose a new objective for optimizing the motor performance of chording gestures, based on my work reported upon in Publication II. Chord input refers to entering one symbol by moving multiple fingers in parallel, similarly to playing a chord on a piano keyboard or producing a sound on a trumpet by pressing a combination of keys or buttons. I will use the term “chording gesture” to denote this simultaneous movement of multiple fingers. Chording gestures can be useful if, for example, there are fewer keys available than symbols, or to increase performance by enabling the input of entire words or phrases with a single input movement. On account of the smaller number of buttons or other control elements, chording devices can often be operated eyes-free. However, chording gestures are not self-explanatory, and they require initial training, which may take longer than other text input methods demand.

The most commonly used chording keyboard is the stenotype, used by stenographers (e.g., in the courtroom) to transcribe speech at very high rates. Pressing a combination of keys enters phonemes, words, or entire phrases. However, this comes at the cost of requiring learning. Professional stenographers need to undergo rigorous training, which usually takes several years, at the end of which they are required to achieve typing rates of 180–225 WPM [23].

Engelbart introduced a chording keyboard comprising five keys, one for each finger [32]. This allowed for 31 key combinations to enter letters, control commands, etc. The set of possible combinations was further extended by inclusion of the three buttons of the mouse. While he acknowledged that this input is less efficient than typing with a keyboard [32], the primary idea was to allow text input without requiring the user to shift attention away from the screen for switching between mouse and keyboard.
Objective Functions for the Letter-Assignment Problem

Figure 4.1. I use angular Fitts’ Law models to quantify the time it takes for each joint \( \theta \) to move from one chording gesture \( k \) to a second gesture \( l \). This time depends on angular distance \( D_{\alpha_k, \alpha_l, \theta} \) between the source and target angle and angular width \( W_{\alpha_l, \theta} \) of the target region. The movement time of the gesture is then determined by the slowest joint.

Other examples of chording systems for text input are the Twiddler one-handed chord keyboard [89] and the two-handed chord keyboard introduced by Gopher et al. [48]. In addition, multitouch chording gestures have been used for marking menus [79] or direct command selection [41, 162] on multitouch devices.

While there are many examples of chord input in text entry, before the work reported on here there was no model that quantifies the time it takes to perform one chording gesture after another. In Publication II, I have proposed a way to assess this time for mid-air chording gestures. This is based on angular Fitts’ Law models for each joint involved in a gesture.

A mid-air chording gesture \( k \) is uniquely defined by the hand configuration \( \Theta_k \), which specifies the angles \( \alpha \) of each joint \( \theta \in \Theta \):

\[
\Theta_k = \{ \alpha_{k, \theta} | \forall \theta \in \Theta, \alpha_{k, \theta} \in \mathbb{R} \}
\]

For example, the angle of a finger corresponds to the angle of its metacarpophalangeal joint in relation to its neutral position (intuitively, how far it is moved “down”; see Figure 4.1). We relax this definition by discretizing the angular range: the possible angle regions are binned into a small number of angular regions, limiting the number of possible gestures. For example, a simple gesture in which a finger is either “up” or “down” can be defined by discretizing the angular range into two bins. Angular values
within the same bin correspond to the same gesture. A more detailed definition of mid-air gestures is addressed in Publication II.

To quantify the time it takes to move all joints from one configuration to another, I consider a chording gesture to be a combination of aimed movements of single joints. The performance of each joint $\theta$ moving from a chording gesture $k$ to a gesture $l$ can be assessed with individual Fitts’ Law models quantifying the time it takes to move from the source angle $\alpha_{k,\theta}$ to the target angle $\alpha_{l,\theta}$:

$$t_{\theta}^{k,l} = a_{\theta} + b_{\theta} \log_2 \left( \frac{D_{\alpha_{k,\theta} \alpha_{l,\theta}}}{W_{\alpha_{l,\theta}}} + 1 \right)$$

where $a_{\theta}$ and $b_{\theta}$ are the Fitts’ Law parameters for the joint $\theta$, $D_{\alpha_{k,\theta} \alpha_{l,\theta}}$ denotes the angular distance between the two configurations, and $W_{\alpha_{l,\theta}}$ is the width of the target region (see also Figure 4.1).

To enable generalizing to multi-finger movements, we assume that the time for several joints to reach a given end posture is bounded by the slowest contributing joint $\theta \in \Theta$. This assumption is based on evidence that complex arm movements are timed such that all joints reach their final position at the same time [64, 130]. Proceeding from the general formulation given in Equation 4.1, I define the objective function for optimizing the assignment of symbols to mid-air chording gestures as follows:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} \max_{\theta \in \Theta} \left( t_{\theta}^{i,j,k,l} \right) \pi_{ij} x_{ik} x_{jl}$$

This decomposition allows me to create a model of multi-finger chording gestures without having to empirically assess the performance of transitioning between each of the many combinations of multi-finger gestures.

The objective function is generally formulated to cover many definitions of chording gestures. A gesture set can be defined with regard to any combination of joints and is not restricted to mid-air gestures. A similar formulation could be used for optimizing the performance of chording gestures with physical keys or touchscreens, where a combination of fingers is moved to enter different joint configurations for performing an input action. The angular Fitts’ Law functions (see Equation 4.5) must be instantiated accordingly to model the performance of individual joints. The formulation of the objective function stays the same. In Section 5.2, I describe the study performed for Publication II to obtain the parameters $a_{\theta}$ and $b_{\theta}$ for the metacarpophalangeal joint of each finger and present results from optimizing the assignment of mid-air chording gestures to symbols. I used the instantiated model to assess the performance of
Engelbart’s chording keyboard and of fingerspelling (here, the American manual alphabet, a part of sign language which defines hand gestures for letters and digits). The assignment of letters to fingerspelling gestures was predicted to allow text input at 43.9 WPM, which is in the same range as the 40–45 WPM achieved by experienced practitioners [126] (see Section 5.2).

4.1.3 Multi-finger input

Multi-finger input is the sequential performance of input actions using different fingers. Hereby, the input actions can overlap in their start and end time. For example, during pressing of a key on a physical keyboard with a finger of the left hand, a finger of the right hand can already be moving toward the next key and even press it before the first key is released. This is different from multi-finger chording or one-finger input, in which one input action must be completed before the next can be started. This overlapping behavior confers a performance benefit on multi-finger input. However, it makes estimation of the time for movement between two input actions difficult, because which cognitive and anatomical factors limit the overlap of input actions is unclear. For this reason, to formulate an objective function, I rely on heuristics and lookup tables based on empirical observations. The drawbacks of this approach are discussed in Chapter 7. In the discussion that follows, I will start by focusing on multi-finger input via physical keyboards performed with the touch-typing system and present objective functions based on evidence from prior work. I will then discuss differences between touch typing and other strategies, referring to observations from my original work, and present an objective function for a special case of text input: special-character entry.

**Touch typing**

Touch typing is the only commonly recognized input strategy for operating a physical keyboard, and it has been taught in all typing courses for more than 100 years. It not only is aimed at the ability to “type by touch” (i.e., without looking at the keyboard) but also enforces a movement strategy wherein each key is always stroked by the same finger. The fingers rest on the middle row of the keyboard (the aforementioned home row), with the index finger of the left and the right hand placed on the f and j key, respectively, and the other fingers resting on the keys next to them. From this position, each finger is responsible for roughly one column of keys (the
<table>
<thead>
<tr>
<th>Criterion</th>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigrams typed with the same finger</td>
<td>F</td>
<td>Binary</td>
</tr>
<tr>
<td>Bigrams typed with the same hand</td>
<td>H</td>
<td>Binary</td>
</tr>
<tr>
<td>Bigrams typed with the left hand</td>
<td>L</td>
<td>Binary</td>
</tr>
<tr>
<td>Bigrams typed from inward to outward</td>
<td>I</td>
<td>Binary</td>
</tr>
<tr>
<td>Distance for bigrams typed by the same finger</td>
<td>dF</td>
<td>Integer</td>
</tr>
<tr>
<td>Vertical movement distance (same hand, different fingers)</td>
<td>dV</td>
<td>Integer</td>
</tr>
<tr>
<td>Horizontal movement distance (same hand, different fingers)</td>
<td>dH</td>
<td>Integer</td>
</tr>
<tr>
<td>Finger cost</td>
<td>f</td>
<td>Real</td>
</tr>
<tr>
<td>Row cost</td>
<td>r</td>
<td>Real</td>
</tr>
</tbody>
</table>

Table 4.2. The heuristic criteria and the corresponding variables used in the objective function described by Equation 4.7 to optimize the assignment of letters to the keys of the physical keyboard when used with the touch-typing strategy. The criteria are based on observations from prior work [29, 30, 57, 162]. The objective formulation in the form of an assignment problem is my contribution.

index fingers are responsible for two columns each and the little finger of the right hand handles all other keys at the right side of the keyboard.

While the finger movements of touch typing are well defined, the complex interactions between fingers that culminate in overlapping movements and the factors influencing typing performance are not yet well understood. Only a few researchers have proposed mathematical models of touch typing that predict the time between two key presses. For instance, Hiraga et al. [57] proposed a linear model based on observations from one expert typist. In contrast, many have observed phenomena involved in typing that contribute to characterizing the finger movements [29, 30, 39, 134, 145]. The observations have been used in heuristics to optimize the keyboard layout [31, 163]. From these observations, scholars conclude that an optimal keyboard should minimize the elements listed in Table 4.2. The movement distance is the Manhattan distance between keys, whereas the finger and row costs are determined on the basis of observations from keystroke data (the literature [57, 163] provides concrete values).

We can formulate these criteria as cost factors in an objective function, with the set of weights $\omega$ for each factor depending on the relative time by which typing is slowed down if such a bigram is typed in comparison to other bigrams. This yields a formulation similar to that produced with the model suggested by Hiraga et al. [57] and the criteria optimized by Wagner.
Objective Functions for the Letter-Assignment Problem

\[
\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} (\omega_F F_{kl} + \omega_H H_{kl} + \omega_L L_{kl} + \omega_I I_{kl} + \omega_d dF_{kl} + \omega_d V_{kl} - \omega_d H_{kl} + \omega_f (f_k + f_l) + \omega_r (r_k + r_l)) x_{ik} x_{jl} \quad (4.7)
\]

The terms are explained in Table 4.2. The weights \(\omega\) can be determined in line with observations from keystroke data by such means as linear regression [57] or on the basis of experts’ opinions [31, 162].

Instead of optimizing a linear combination of heuristic criteria, designers can take another approach: directly determine the movement times between any two keys on the basis of video analysis of typing [120, 18] or automatically recorded timestamps of key presses [72, 60]. The movement times can be stored in a lookup table \(T\) where the time to move from key \(k\) to key \(l\) is found in cell \(T_{kl}\). With these conditions, we can formulate the objective function for optimizing the motor performance for a letter-to-key assignment thus:

\[
\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} T_{kl} x_{ik} x_{jl} \quad (4.8)
\]

In principle, such a lookup table can be obtained for any set of input actions, and the same formulation can be used to optimize the performance of any text input method. However, the number of actions may render it costly and time-consuming to obtain a number of observations for each pair that is large enough to enable reliable estimates of their movement times.

**Multi-finger typing on physical keyboards**

The heuristics used for optimization that are presented in the previous section are based on work that is over 30 years old. Studies in those days used professionally trained typists, often typing on mechanical or electric typewriters whose physical and sensing properties are very different from those of today’s modern keyboards. Many computer users do not undergo any formal training in touch typing. Instead, their typing strategy – how they move their hands and fingers over the keyboard and which finger presses which key, etc. – emerge and manifest themselves during everyday computer use. People not using the touch-typing method have been called “idiosyncratic” [128] or “nonstandard” [87]. Here, I will refer to *self-taught* or *untrained* typists, to reflect simply the fact they never took a formal typing course in which they learned the touch-typing system.

As publications III and IV attest, people use anywhere from two to 10
fingers and show large variations in typing strategy. Nevertheless, the
work identifies some commonalities in their behavior. For Publication III,
typists were clustered on the basis of their finger-to-key mapping, and four
groups were found for the left hand and six for the right. The people in each
group used the same fingers to press the relevant keys on the keyboard.
However, independent of the finger-to-key mapping, other aspects of the
motor behavior were found to contribute to high-performance typing, such
as overlap between input actions, minimal global hand movement, and
consistent use of the same finger for a given key.

The heuristics described in Table 4.2 cannot necessarily be used for the
optimization of modern keyboards used by people with varying typing
strategies. Although I have conducted extensive research to understand
modern typing behavior, it is still unclear which factors contribute to high
motor performance, and no model exists that predicts the time for move-
ment between key presses by self-taught typists. Therefore, no objective
function can be proposed. That said, proceeding from the work presented in
publications III and IV, I will provide a more in-depth analysis of modern
typing behavior in Chapter 6.

Special-character entry
In the following discussion, I will formulate a new objective function for
optimizing the assignment of special characters to input actions with re-
spect to a fixed assignment of alphabetic letters (e.g., a Qwerty keyboard).
This is based on my original (thus-far unpublished) work, which is further
described in Section 5.3. Entering special characters such as punctuation
marks, mathematical symbols, and currency or other symbols is a special
case of text input. There are important differences from entering the let-
ters of the alphabet: (1) most symbols are highly infrequent in comparison
to most letters, (2) they are mostly entered before or after a letter (i.e.,
seldom in combination with other symbols), and (3) the set of symbols
is typically very large. With sets of 100 characters or even more, it is
too computationally complex to solve or even approximate the general
quadratic assignment problem as introduced by Equation 4.1. However,
if the problem is to assign a set of special characters with respect to an
already fixed assignment of letters, then we can linearize the quadratic
problem by considering only the time it takes to perform an input action
$k$ before or after any fixed letter $c$. Transitions between two input ac-
tions corresponding to special characters are so infrequent that they can
be overlooked. Accordingly, the objective function can be formulated as follows:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{c=1}^{L} (p_{ic} T_{ck} + p_{ic} T_{kc}) x_{ik}$$  (4.9)

where $p_{ic}$ denotes the frequency of the symbol–letter pair $i–c$ and $T_{kc}$ quantifies the time it takes to perform input action $c$ after input action $k$. Note that the movement time is not symmetric; that is, $T_{kc} \neq T_{ck}$.

In a parallel with Equation 4.8, it is determined empirically and stored in a lookup table. This formulation is linear in the decision variables and hence allows much faster computation of an optimal solution (see Chapter 2) while still capturing the relevant factors that influence motor performance. In Section 5.3, I will apply this objective to find an optimal assignment of special characters to physical-keyboard slots. However, the same formulation can be used for other types of input action used for entering special characters, such as hand or drawing gestures.

4.2 Ergonomics

Human anatomical characteristics constrain the design of a text input method. These may impose hard constraints, rendering it physically impossible to perform a given movement, or soft constraints, under which a movement is possible but repeated performance puts strain on muscles and tendons. This might lead to repetitive-strain injuries such as carpal tunnel syndrome or simply bring about fatigue for a user more quickly. Text input is a highly repetitive task, and input is typically performed for several hours per day (see, for example, the results from a survey reported on in publications III and IV). Therefore, ensuring that the movements are performed in an ergonomic way is important if the user is going to be able to maintain a high input speed over a long time.

In this section, I formulate assignment problems that minimize the ergonomic cost of entering text, particularly with respect to uncomfortable and fatiguing movements that pose a risk of injury and affect usage over a longer time. We generally formulate this objective with the linear formulation of the assignment problem presented in Equation 2.3. The goal is to minimize the ergonomic cost of performing an input action, weighted by the frequency of the symbol mapped to the action. This assumes that the ergonomic cost is not dependent on the previous input action but, rather, related to the end posture of an action. We can formulate the general
<table>
<thead>
<tr>
<th>Motor performance</th>
<th>Objective function</th>
<th>Equation</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-finger input (movement model)</td>
<td>$\min \ a + b \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} \log_2 \left( \frac{D_{kl}}{W_l} + 1 \right) x_{ik} x_{jl}$</td>
<td>4.3</td>
<td>[13, 14, 36, 93, 180, 181]</td>
</tr>
<tr>
<td>One-finger input (distance)</td>
<td>$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} D_{kl} x_{ik} x_{jl}$</td>
<td>4.4</td>
<td>[27, 82], Publication I – see Section 5.1</td>
</tr>
<tr>
<td>Chord input</td>
<td>$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} \max_{\theta \in \Theta} \left( t^{\theta}<em>{kl} x</em>{ik} x_{jl} \right)$ where $t^{\theta}<em>{kl} = a</em>{\theta} + b_{\theta} \log_2 \left( \frac{D_{\alpha_k, \alpha_l, \theta}}{W_{\alpha_l, \theta}} + 1 \right)$</td>
<td>4.5, 4.6</td>
<td>Publication II – see Section 5.2</td>
</tr>
<tr>
<td>Touch typing (heuristics)</td>
<td>$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} \left( \omega_F F_{kl} + \omega_H H_{kl} + \omega_L L_{kl} + \omega_I I_{kl} + \omega_d F_{kl} + \omega_d V_{kl} \right)$ $-\omega_d H_{kl} + \omega_j (f_k + f_l) + \omega_r (r_k + r_l) \right) x_{ik} x_{jl}$</td>
<td>4.7</td>
<td>[29, 30, 31, 57, 163]</td>
</tr>
<tr>
<td>Touch typing (movement time)</td>
<td>$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} T_{kl} x_{ik} x_{jl}$</td>
<td>4.8</td>
<td>[18, 60, 72, 120]</td>
</tr>
<tr>
<td>Multi-finger input of special characters</td>
<td>$\min \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{c=1}^{L} (p_{ci} T_{ck} + p_{ic} T_{kc}) x_{ik}$</td>
<td>4.9</td>
<td>My unpublished work – see Section 5.3</td>
</tr>
</tbody>
</table>

Table 4.3. An overview of the objective functions presented in Section 4.1. They allow optimizing the assignment of symbols to input actions with respect to motor performance, in keeping with the input method employed. The objective functions are based on my work presented in this thesis or are formulated in line with observations from prior work, referenced in the rightmost column.
objective as the following integer program:

$$
\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i e_k x_{ik}
$$

(4.10)

where $x_{ik}$ is the binary decision variable denoting whether or not a symbol $i$ is mapped to an input action $k$, $p_i$ the frequency of the symbol $i$, and $e_k$ the ergonomic cost of performing the input action $k$.

Below, I will discuss several ways to assess this cost in terms of fatigue and anatomical limitations. I have formulated the objectives presented but mostly on the basis of empirical data and models offered by others. There is little work on explicitly optimizing the ergonomic factors of a text input method; however, optimizing for motor performance correlates to some extent with the optimization of ergonomics, since high input speed can be sustained for a longer time only if the movements are performed ergonomically. At the end of the section, I provide a summary of the proposed functions, in Table 4.4.

### 4.2.1 Strain

In this subsection, I propose a new objective function for minimizing the risk of repetitive-strain injuries, stemming from heuristics derived from prior work. Repetitive motions with awkward postures put strain on tendons and joints, which can cause repetitive-strain injuries such as carpal tunnel syndrome [175]. While there has been extensive research on the risk of such injuries from keyboard and mouse use [1, 33], almost no work has been done on text entry optimization to minimize the stress on tendons and joints.

In a similarity to the performance optimization in Equation 4.7, I use a set of heuristic criteria to compute an ergonomic score for performing an input action on a physical keyboard. These criteria are based on studies that assessed the ergonomic risk factors during typing on physical keyboards with regard to the strain put on tendons and joints [1, 33]. The objective function penalizes two types of movements:

1. **Ulnar or radial deviation of the wrist** (extreme outward or inward movements) – e.g., to reach the keys in a far-left or far-right corner.

2. **Flexion and extension of fingers** (e.g., toward the uppermost row).

Repeating such extreme movements places strain on tendons and joints which can lead to injuries. Targeted at typing on a physical keyboard, the
objective function additionally penalizes the use of one or two modifier keys (Alt, Ctrl, Shift, etc.). Typing strategies employed by the user vary in how the modifier keys are handled. These keys are held down by one finger while a finger of the same hand makes the to-be-modified key press. Depending on the typing strategy (see Chapter 6), this might require extreme movements of the wrist and fingers, which put additional strain on joints and tendons. The objective function therefore minimizes the weighted sum of these heuristics:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i (\omega_W W_k + \omega_F F_k + \omega_M M_k) x_{ik}$$

where $W_k$ and $F_k$ are binary variables denoting extreme movements of the wrist and fingers as described above and $M_k$ represents the number of modifier keys used. $\omega$ can be used to give different weights to the factors. However, it is unclear whether any factor represents a more severe ergonomic risk than others and thereby justifies uneven weighting. An application of this objective function in the optimization of a physical keyboard layout is described in Section 5.3.

Although targeted for text input on a physical keyboard, the objective function introduces the general use of a weighted linear sum of heuristics assessing the ergonomic risk of an input action. The formulation can be adapted to minimize the ergonomic cost of other input methods. Their corresponding metrics might come from heuristics, as described above, or be based on empirical measurements or movement models explicitly quantifying the stress on muscles and tendons. For example, using the thumb to press buttons on a soft keyboard of a mobile phone while holding the phone in the same hand may require similarly extreme movements. Their frequency should be minimized for reducing the stress put on tendons and joints during the button presses. In this case, results from studies of the functional area of the thumb [11] could be applied to adapt the objective function accordingly.

### 4.2.2 Muscle fatigue

Recently, authors have proposed non-intrusive methods of assessing the strength and fatigue of arm muscles during interaction – in particular, with mid-air and touch interfaces [4, 56]. They use motion capture technology to record the movements of the arm and then utilize biomechanical models to estimate the muscle load during interaction.

Hincapié-Ramos et al. [56] have presented a model for assessing shoulder
fatigue during interaction in mid-air. Working from a biomechanical model of the arm, they computed the torque applied to the shoulder from only the position and acceleration of the shoulder during interaction. This allowed them to quantify the ergonomic cost of performing an input action as the torque applied during interaction, expressed as a percentage of the maximum possible torque. The objective function for minimizing the muscle fatigue with a given assignment of symbols to input actions can be formulated accordingly:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i \left( \frac{\|T_k\|}{T_{max}} \cdot 100 \right) x_{ik}$$

(4.12)

where $\|T_k\|$ is the torque applied to the shoulder during the input action $k$ and $T_{max}$ is the maximum torque. This metric has been used to optimize the assignment of letters to the buttons of a virtual keyboard operated in mid-air [56], but it could be used also to assess the shoulder fatigue for other input actions involving shoulder movement – for example, drawing gestures in mid-air or making button presses on a public display.

A more fine-grained assessment of muscle fatigue for the whole arm has been proposed by Bachynskyi et al. [4], who used biomechanical simulation to infer the muscle load of all arm muscles during mid-air interaction. Taking the tracked position of joints and the distribution of human body mass, the simulation allows determining the required torsion and forces at joints and is thereby able to estimate muscle activations during movement. The authors used this method to compare movements in different parts of the 3D space around the user [6] and compare the interaction-related effort for different touch surfaces [5] (tablets, laptops, public displays, etc.).

In principle, this method could be used to assess the arm’s muscle load from any manual input action for text entry. The general-form Equation 4.10 can be used to optimize the assignment of input actions to symbols, where $e_k$ then represents the muscle load determined to occur during input action $k$. Thus far, the work by Bachynskyi et al. has considered only the activation of the arm muscles so remains most suitable in relation to pointing-based input actions. A biomechanical model of the hands and fingers would be necessary for purposes of assessing fatigue with multi-finger input also (for example, for typing on physical keyboards). Such a model would allow comparisons between the various typing strategies discussed in Publication III in addition (see Chapter 6).
4.2.3 Finger individuation

In Publication II, I have presented an objective function for optimizing the ergonomic “comfort” of a multi-finger chording gesture by maximizing the individuation with which fingers can be moved as part of a gesture. For example, for anatomical reasons, it is not possible to bend the ring finger separately from all others; the little and middle finger are always dragged along to a certain extent. If performed carefully, such gestures can be recognized, but they are harder to perform, since the user has to counteract the enslaving of the relevant fingers, which puts additional strain on the joints and tendons.

Schieber and colleagues proposed an individuation index to quantify the degree of this effect for individual fingers [54, 137]. They calculated the relative coactivation $C_{\theta \phi}$ of non-instructed joint $\theta$ during the instructed movement of joint $\phi$ as the correlation between the position of $\theta$ and the position of $\phi$. The individuation index of moving an individual joint $\phi$ is then computed as the average relative coactivation of all other joints during the movement of $\phi$:

$$I_\phi = 1 - \frac{1}{|\Theta| - 1} \sum_{\theta \in \Theta \setminus \{\phi\}} |C_{\theta \phi}| \quad (4.13)$$

where $\Theta$ denotes the set of all joints. A relative coactivation index of 1 denotes that the non-instructed joint $\theta$ moves to the same extent as the instructed joint $\phi$, whereas an individuation index of 1 means that $\phi$ moves independently of all non-instructed joints.

I extend this index to multi-joint gestures in light of the following observation: If moving joint $\phi$ is part of a gesture $k$ and $\theta$ is not, coactivation $C_{\theta \phi}$ contributes to the individuation index of the gesture. In contrast, if these two joints are moved together as part of the gesture, this coactivation is irrelevant. Therefore, to compute the individuation index $I_k$ of a multi-finger gesture $k$, firstly we compute the maximum relative coactivation of any non-instructed joint $\theta$ during the movement of any instructed joint $\phi$, and then we take the average over all non-instructed joints:

$$I_k = 1 - \frac{1}{|\Theta| - |\Theta_k|} \sum_{\theta \in \Theta \setminus \Theta_k} \max_{\phi \in \Theta_k} |C_{\theta \phi}| \quad (4.14)$$

where $\Theta$ denotes the set of all joints and $\Theta_k$ the set of the joints instructed as part of gesture $k$. An individuation index of 1 means that none of the non-instructed joints move when the user performs the gesture, whereas
Objective Functions for the Letter-Assignment Problem

an index of 0 means that all joints move to the same extent, even if they are not part of the gesture.

The objective function then maximizes the ergonomic comfort of performing the gesture by assigning more frequent symbols to gestures with higher individuation:

$$\max \sum_{i=1}^{N} \sum_{k=1}^{M} \pi_i I_k x_{ik}$$

(4.15)

In Section 5.2, I will describe the work I conducted for Publication II to obtain the relative coactivation of each joint and optimize assignment of multi-finger gestures to symbols for maximal individuation.

While this work was done for mid-air gestures, the so-called enslaving effect has been observed also in pressing on a surface [179]. Hence, similar models could be used to quantify the finger individuation during other input actions, involving tactile feedback from pressing on a surface.

4.3 Learnability

Learnability has to do with the intuitiveness and ease of use of a text input method, particularly for novice users. These are important factors in UI design [38] and are among the elements that determine whether a new technology gets adopted [159]. The effort to learn a new technology plays an especially important role in text input. The Qwerty keyboard has been the prevailing text input method for over a century. The arrangement of letters in a three-row grid is familiar to most people, thanks to the typewriter and, more recently, the desktop computer, so it has been adopted for almost all computing devices, from the smartwatch to public terminals and virtual-reality systems.

This has contributed to users’ reluctance to learn a new system with which their initial performance would drop upon switching from Qwerty, even though they might be faster or gain ergonomic benefits in the long term (e.g., the Dvorak Simplified Keyboard [29, 30], stenotype machines [23], or the Twiddler one-handed keyboard [89]). In contrast, systems that build on and extend the Qwerty layout have been adopted quickly. One example is gesture typing on soft keyboards [74], which exploits the user’s existing knowledge of the Qwerty design and thereby yields high performance rates from the very first use [73].

Novices’ performance is governed largely by visual search for the right input action to select for entering a symbol [94, 142]. In this context,
<table>
<thead>
<tr>
<th>Ergonomics</th>
<th>Objective function</th>
<th>Equation</th>
<th>References</th>
</tr>
</thead>
</table>
| Strain          | \[
\min_{i=1}^{N} \sum_{k=1}^{M} (p_i (\omega_W W_k + \omega_F F_k + \omega_M M_k)) x_{ik}
\] | 4.11     | My unpublished work – see Section 5.3         |
| Muscle fatigue  | \[
\min_{i=1}^{N} \sum_{k=1}^{M} p_i \left( \frac{\|T_k\|}{T_{max}} \times 100 \right) x_{ik}
\] | 4.12     | [56]                                           |
| Finger individuation | \[
\max_{i=1}^{N} \sum_{k=1}^{M} p_i I_k x_{ik} \quad \text{where} \quad I_k = 1 - \frac{1}{|\Theta| - |\Theta_k|} \sum_{\theta \in \Theta \setminus \Theta_k} \max_{\phi \in \Theta_k} |C_{\phi \theta}| 
\] | 4.14, 4.15 | Publication II – see Section 5.2               |

Table 4.4. An overview of the objective functions presented in Section 4.2. They allow one to optimize the assignment of symbols to input actions with respect to the ergonomics of the movements. The objective functions are based on my work as presented in this thesis or are formulated in line with models from prior work, referenced in the rightmost column.
a familiar arrangement of symbols yields higher performance [142]. In cases wherein no visual search is possible (e.g., with chording gestures), the assignment of input actions to symbols must be memorized before the system can be used, with the memorization time growing with the number of input actions (in accordance with the Hick–Hyman law) [55, 59]. To address these issues, in this chapter I present objective functions for minimizing the efforts related to visual search and memorization. In addition to memorization of the assignment of input actions to symbols, a text input method might require learning of new motor skills. Proceeding from theories of motor learning [105, 130, 138], I argue that we can optimize the assignment of input actions to symbols to reduce users’ learning time. In the subsections that follow, I present two corresponding objective functions. Table 4.5 summarizes the objective functions presented in this part of the chapter. Almost all were developed in conjunction with the publications connected with this thesis and were informed by literature on motor learning, visual search, and memorization. While memorization and motor learning involve complex cognitive processes, the literature’s heuristics, which I used to assess the input actions, proved to be rather simple. Additional empirical studies should be conducted to validate their applicability to the specific optimization cases presented in Chapter 5. This work was beyond the scope of the thesis project.

4.3.1 Motor learning and complexity

In this subsection, I present a metric developed for Publication II that quantifies the complexity of performing an input action. A classical problem in motor control research is the degrees of freedom problem [130]: how is a movement selected from among the many distinct possible movements that could be performed to achieve a certain task? Consider, for example, the pressing of a key on the physical keyboard. A user can use any of the 10 fingers to move to the key, press it, and release it, where each finger has at least three degrees of freedom (for the $x$, $y$, and $z$ positions in space). To address this problem, Bernstein has proposed the concept of synergies – interactions between joints that reduce the number of degrees of freedom (see literature [12] cited by Rosenbaum [130]). These synergies are developed during motor learning, and thus the number of degrees of freedom to be controlled is reduced with practice [105]. In Publication III, I reported that faster typists are more consistent in the finger they use to press a certain key, indicating that they have developed synergies wherein
always the same finger is moved to reach the given target key while the
others are kept static (resting at a key or moving alongside the aiming
finger). This decreases the number of degrees of freedom to be managed.

Applying this theory, I assume that the cost of motor learning is lower if
the input action itself does not demand that as many degrees of freedom
be controlled. Thus, simpler synergies can be developed in a shorter time.
In Publication II, I have proposed a score representing the complexity of
multi-finger chording gestures. It quantifies two aspects:

1. $|\Theta_k|$: the number of joints involved in the gesture $k$, which determines
   the number of degrees of freedom.

2. $|A_k|$: the number of separate target angles the joints must reach
   when performing the gesture $k$. This assumes that the movement of
   multiple joints can be collapsed into fewer degrees of freedom if all
   joints are moved as a whole to the same configuration.

The sum of the two is normalized with respect to the maximum number
of joints $|\Theta|$ and multiplied by 0.5 to yield a complexity score between 0
and 1, where 1 indicates a gesture configuration for which all joints must
reach different target angles. Through this, I can optimize the assignment
of symbols to gestures with respect to complexity and motor learning by
using the following objective:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i \cdot 0.5 \cdot \frac{|\Theta_k| + |A_k|}{|\Theta|} x_{ik} \quad (4.16)$$

where the goal is to minimize the complexity of a gesture as weighted by
the frequency of the symbol it is assigned to. An example optimization
case utilizing this objective is presented in Section 5.2.

Note that the same formulation can be used for any multi-finger chording
gesture, not just those performed in mid-air.

### 4.3.2 Transfer of motor skills

This subsection presents a novel objective function for optimizing the as-
signment of symbols to input actions such that a user can take advantage of
existing motor skills honed in other tasks and thereby reduce learning time.
Research on the transfer of motor skills usually concerns itself with the
question of how learning one task affects the performance of another task.
Such work forms the foundation for simulation-based training (e.g. done
by pilots) and the creation of training programs wherein subtasks are
practiced individually before being combined in a higher-level task [138].
I follow a more direct strategy to transfer motor skills. Given the movements performed to achieve a certain task, the goal is to utilize the same movements to perform another task, with the hypothesis that a similar motor performance can be achieved. For the design of a text input method, this means that the user can perform the same input actions for entering symbols that he or she does when pursuing another task. An example case is considered in Publication I (see Section 5.1), involving optimizing the assignment of symbols to the keys of a piano such that a pianist can utilize the same movements for entering text as when playing music. This is achieved by assigning frequent symbol pairs to frequent note pairs, which correspond to well-practiced movements in playing of music. This is formulated as the following quadratic assignment problem:

$$\max \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} f_{kl} x_{ik} x_{jl}$$

(4.17)

where $p_{ij}$ denotes the frequency of the symbol pair $i-j$ and $f_{kl}$ that of the note pair corresponding to the movement from the piano key $k$ to key $l$. The goal is to maximize the frequency of movements often performed in playing music, which essentially aligns the distributions of input actions between the two tasks.

I present the data collection and optimization of the piano keyboard for text input in Section 5.1. It should be noted, however, that the same formulation can be used to optimize a text input method for skill transfer between any two tasks that utilize the same input actions.

### 4.3.3 Memorability

I will now present an objective function for maximizing the memorability of the assignment of symbols to multi-finger chording gestures. It is based on my work reported on in Publication II but formulated as an integer program. Multi-finger chord input, as used with stenographs [23] or Twiddler [89], has many potential benefits in comparison to pointing-based input methods, such as higher input speed, better ergonomics, and eyes-free performance. However, methods of this sort have a steep learning curve. Without any visual guidance, users have to explicitly memorize each gesture before they can enter text. In order to reduce this effort, I propose optimizing the assignment of symbols to input actions with respect to the memorability of the gesture set. Studies of human memory suggest that organization, categorization, and chunking are essential to formation of more durable long-term memory traces [97]. Building on these findings,
I propose a memorability score that uses a categorization of gestures into mnemonic sets, \( m \in \mathcal{M} \), where

\[
m(k) = \begin{cases} 
1 & \text{if gesture } k \text{ is part of the set } m \\
0 & \text{otherwise}
\end{cases}
\]

These are groups of gestures that share a certain mnemonic structure, such as use of neighboring fingers or combinations with the thumb [162].

The goal is to maximize the memorability of a gesture set by maximizing the proportion of gestures that belong to mnemonic sets while minimizing the total number of mnemonic sets covered by an assignment. The latter part of the memorability score quantifies an aspect of the gesture set as a whole rather than of an individual symbol–gesture pair. The general formulation of the assignment problem as introduced in Chapter 2 does not suffice for modeling this. Therefore, the objective function presented below utilizes additional binary decision variables \( y_m \) that denote whether any of the assigned gestures belongs to the mnemonic set \( m \):

\[
y_m = \begin{cases} 
1 & \text{if } \exists k \in \{1 \ldots M\} : m(k) = 0 \\
0 & \text{otherwise}
\end{cases}
\]

Then we can model the objective as follows:

\[
\max \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{m \in \mathcal{M}} \frac{m(k)}{M} x_{ik} - \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{M}|} y_m
\]  
\tag{4.18}

The first term quantifies the percentage of assigned gestures belonging to any mnemonic set (between 0 and 1), while the second one gives the number of individual mnemonic sets that together constitute the set of assigned gestures (normalized, between 0 and 1). In addition, we must apply the following constraints to link the new decision variables \( y_m \) to the decision variables \( x_{ik} \):

\[
y_m \geq m(k) x_{ik} \quad \forall i \in \{1, \ldots N\}, k \in \{1, \ldots M\}, m \in \mathcal{M}
\]  
\tag{4.19}

Since the objective function is intended to minimize the number of mnemonic sets, a decision variable \( y_m \) will have the value 0 if no symbol \( i \) is mapped to a gesture \( k \) that is in the set \( m \). Otherwise, the constraint enforces the condition \( y_m = 1 \).

I applied a variation of this function to the optimization of mid-air gestures for Publication II, as discussed in Section 5.2. However, this formulation can be used to optimize the memorability of any set of chording gestures for other input devices too, such as the Twiddler one-handed keyboard [89] or Engelbart’s chording keyboard [32].
4.3.4 Intuitiveness and familiarity

In this subsection, I propose two objective functions to minimize the effort related to visual search upon encountering a novel graphical text input method for which the input action assigned to a symbol can be ascertained by visually searching the interface (e.g., a keyboard layout). In this situation, novice users do not need to explicitly memorize an assignment; they can learn it by using the system. On the other hand, their performance is slowed down by the time they spend visually searching for a symbol [94]. Prior work has shown that a familiar assignment reduces this search time, with, for example, the Qwerty arrangement or an alphabetical layout being faster to search than a random layout is [142], and less learning is required when just a few changes have been made to a familiar layout [63]. An easy-to-search assignment is particularly important in situations wherein a high walk-up performance level is necessitated by the text input method being used only briefly (e.g., for public terminals) or when infrequently used symbols must be entered (e.g., special characters such as less common punctuation marks).

**Intuitiveness**

Spatial grouping of similar symbols allows the user to exploit top-down learning strategies, which have been shown to decrease the time for memorizing the letter positions in a virtual keyboard layout [77]. Moreover, intuitive spatial organization can help the user find infrequently used symbols more quickly if they are placed in the vicinity of similar symbols, where users may expect them. Accordingly, the objective function for improving the intuitiveness of a text input method minimizes the distance between similar characters. This can be formulated as a quadratic assignment problem:

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l=1}^{m} s_{ij} D_{kl} x_{ik} x_{jl}
\]

Here, \(s_{ij}\) denotes the similarity between symbols \(i\) and \(j\), and \(D_{kl}\) quantifies the distance between two input actions \(k\) and \(l\). In the case of a graphical keyboard layout, this distance is calculated as a number of keys, in terms of the Manhattan distance. Note that in this case, the importance of grouping two symbols in mutual proximity is determined not by their frequency, as for other objectives, but by only their similarity. In consequence, grouping of infrequent characters has the same priority as grouping of frequent ones. The similarity score needs to be determined empirically from user input or...
Objective Functions for the Letter-Assignment Problem

based on experts’ ratings. It is a value between 0 and 1, where 0 refers to no similarity and 1 to maximum similarity. It can be defined with respect to various criteria, such as visual, semantic, and phonetic similarity.

In Section 5.3, I will delve further into the various criteria we used to define a similarity score and optimize the grouping of special characters on a physical keyboard. However, the same formulation can be used for other text input methods, for which an underlying model of distance between input actions exists. For example, it is more intuitive to assign similar symbols to chording gestures that are “close to” each other, where the distance function could be defined, for example, in terms of the number of fingers by which two gestures differ from each other.

**Familiarity**

As was discussed above, a familiar assignment reduces visual search time and thereby improves the performance of novice users [94, 142]. For example, if users are experienced with a certain keyboard layout, they will seek the position of a given symbol in a new layout by starting at its location in the old layout. If the symbol is at the same position or in its peripheral vicinity, the user can find it with ease. If, on the other hand, it is further from this starting point, the user has to search the entire keyboard. Moreover, in relearning of a layout, the old and new position of a symbol will compete in their activation from long-term memory [63]. A user will always look in the wrong location at first, before the new location has gained sufficient activation in the long-term memory [63]. Hence, relearning will be faster if the symbol is located close to the original position, where it can be found more quickly.

Scholars active in keyboard layout optimization have proposed optimization in terms of a tradeoff between performance and similarity to the Qwerty keyboard [13, 15, 27] or to an alphabetical organization [182]. The similarity to a layout can be optimized by minimizing the distance of frequent characters from their position in the familiar layout (here, Qwerty or a local variant). Accordingly, the corresponding objective function can be formulated as a linear assignment problem as follows:

\[
\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i D_{kL(i)} x_{ik} \tag{4.21}
\]

where \(D_{kL(i)}\) denotes the distance between the key slot \(k\) assigned to symbol \(i\) and the position of symbol \(i\) in the familiar layout, denoted by \(L(i)\). The distance is weighted by the frequency \(p_i\) of symbol \(i\). Dunlop and Levine
use the Euclidean distance [27], whereas Smith, Bi, and Zhai use the Manhattan distance for reason of the grid-like organization of letters on a keyboard [150]. That is also what I use in my work, in which I apply this objective function to optimize special characters’ input with a physical keyboard layout, further described in Section 5.3.

Another approach is taken by Bi et al. [13, 15], who constrain the design space to solutions that are similar to the Qwerty layout. They propose several constraints, by which letters cannot be placed further than one key away from their position in the Qwerty layout, they can only be moved horizontally and by one key at maximum, and only swapping of position within two-letter pairs is allowed [15]. These can be easily formulated as additional constraints in the letter-assignment problem. For example, the following constraint inhibits a character from being assigned to a key whose distance from the position in a familiar layout is more than one unit:

\[
D_{kL(i)}x_{ik} \leq 1 \quad \forall i \in \{1, \ldots, N\}, k \in \{1, \ldots, M\}
\]  

(4.22)

where \(D_{kL(i)}\) denotes the Manhattan distance (i.e., distance as the sum of the number of rows and columns) between key \(k\) and the position of character \(i\) in the reference layout.

While prior work has applied these objectives and constraints to optimize the spatial arrangement of a keyboard layout, they can be used to optimize any letter-assignment problem for which we can define a meaningful function for the distance between input actions (e.g., for continuous gestures [150]).

4.4 Input Recognition

While the preceding parts of this chapter all were concerned with objectives related to how a user learns and controls an interface, the final category of objectives is somewhat orthogonal to this, in that it entails considering how the system can interpret the signal in the most reliable way. I primarily review objectives used in prior work and formulate them in terms of the assignment problem, in order to show that it can serve as a common framework for solving various problems related to the design of text input methods. However, I have not considered this category in my own work, so it is not covered by the cases described in Chapter 5.

As discussed in the beginning of this chapter, prior work mainly addresses two aspects of input recognition: noise and disambiguation. With
<table>
<thead>
<tr>
<th>Learnability</th>
<th>Objective function</th>
<th>Equation</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>$\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i 0.5 \frac{</td>
<td>\Theta_k</td>
<td>+</td>
</tr>
<tr>
<td>Skill transfer</td>
<td>$\max \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} f_k x_{ik} x_{jl}$</td>
<td>4.17</td>
<td>Publication I – see Section 5.1</td>
</tr>
<tr>
<td>Memorability</td>
<td>$\max \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{m \in \mathcal{M}} m(k) x_{ik} - \sum_{m \in \mathcal{M}} \frac{1}{</td>
<td>\mathcal{M}</td>
<td>} y_m$</td>
</tr>
<tr>
<td>Intuitiveness</td>
<td>$\min \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} s_{ij} D_{kl} x_{ik} x_{jl}$</td>
<td>4.20</td>
<td>My unpublished work – see Section 5.3</td>
</tr>
<tr>
<td>Familiarity</td>
<td>$\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i D_{kL(i)} x_{ik}$</td>
<td>4.21</td>
<td>[27] and my unpublished work – see Section 5.3</td>
</tr>
</tbody>
</table>

Table 4.5. An overview of the objective functions presented in Section 4.3. These allow optimizing the assignment of symbols to input actions with respect to the result’s learnability in terms of motor learning but also memorability and visual search time. The objective functions are either based on my work, presented in this thesis, or formulated on the basis of observations from prior work, referenced in the rightmost column.
regard to both, the assignment of symbols to input actions is optimized to make the recognition of input more reliable. On one hand, problems in recognizing an input action can arise from noise in users’ input. Algorithmic techniques such as autocorrection make use of the redundancy in natural languages [146] to predict the most likely symbols from noisy input actions, using models of the language and users’ movements [47, 165]. On the other hand, the design of the text input method might require more: disambiguation of the user’s input. This occurs in cases of the same input action being used to enter several symbols. In any case, the right assignment can facilitate the prediction of symbols, as is explained below.

4.4.1 Noise

Following the paradigm of the communication channel discussed in Chapter 1, there can be many sources of noise during text input, including the users’ movements as well as the input sensing by the system. This is a particular problem for systems wherein the input is not binary. Unlike the input sensing in the binary case of button presses on a physical keyboard, sensing based on touch, vision, or an accelerometer has to convert a continuous input signal to a discrete input action, which can introduce many sources of error and leads to uncertainty about the user input from the system’s perspective [141, 165].

Autocorrection

Autocorrection algorithms have been introduced as a technique for soft keyboards to deal with the uncertainty arising from typing mistakes, noisy input by the user, and/or misrecognition of an input action by the system [47]. Given a sequence of potentially noisy input actions, they use models of the language and users’ movements to predict the most likely sequence of symbols entered [47, 165]. For example, a user trying to type “and” might miss the a key and type “snd” instead. The movement model predicts that a user is frequently going to miss the a key and press the neighboring s instead, thereby producing a less frequent word. Thus the algorithm detects this as a mistake and automatically corrects the sequence to the more frequent word (here, “and”). This works well if the mistyped symbols result in a word that is unlikely according to the language model. However, there are many pairs of letters for which valid and frequent words in the English language are produced when one letter is replaced with the other (e.g., i and o, in cases such as “in”)
and “on”). An automatic correction attempt fails in these cases. Dunlop and Levine termed such letter pairs “badgrams” [27] and proposed an objective function for optimizing a soft keyboard layout for autocorrection by minimizing the frequency of badgrams placed next to each other. Their objective can be formulated as a quadratic assignment problem:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} b_{ij} N_{kl} x_{ik} x_{jl}$$

(4.23)

where $b_{ij}$ denotes the badgram frequency of $i$ and $j$ and where $N_{kl}$ is 1 if keys $k$ and $l$ are located next to each other and 0 otherwise. According to Dunlop and Levine, the badgram frequency is calculated by scanning all same-length words in a corpus and assigning a badgram a frequency corresponding to the frequency of the more common of the two valid words that differ only by the letters making the badgram [27].

While this objective function optimizes the assignment of letters to keys on a soft keyboard, it can be generalized to other symbols and input actions wherein noise in the users’ input leads to the system misrecognizing one input action as another that could produce a valid word. For example, two chording or drawing gestures might be frequently misclassified by the system. We can capture this misrecognition in terms of a confusion matrix $C$, where $C_{kl}$ denotes the probability of input action $k$ being recognized as input action $l$. Accordingly, a more general version of Equation 4.23 can be formulated thus:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} b_{ij} C_{kl} x_{ik} x_{jl}$$

(4.24)

**Input uncertainty**

Advanced technologies for input sensing, such as computer vision sensors and eye tracking, offer new possibilities for text input methods. However, their input recognition is error-prone. For example, in a publication unrelated to this thesis, I have shown that the tracking quality yielded by state-of-the-art eye trackers varies greatly between users and with the tracking conditions [35]. However, most gaze-enabled applications do not take this uncertainty into account when processing the user’s input or in the design of their interface. In consequence, many applications are unusable by large sets of people or under certain conditions. A quantitative model of uncertainty for a given sensor would allow optimization of text input methods with respect to the accurate recognition of input actions by the system. A straightforward way to minimize recognition problems,
and thereby typing mistakes, would be to assign frequent symbols to well-recognized input actions. This can be formulated as the following linear assignment problem:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_{i} U_{k} x_{ik}$$  \hspace{1cm} (4.25)$$

where $U_{k}$ quantifies the uncertainty of sensing input action $k$. If the recognition varies with the previous input action, a quadratic formulation can be used (see Equation 2.4, in Chapter 2). There are several ways in which this input uncertainty could be quantified, depending on the input method. Among them are measurements of the accuracy and precision of eye tracking systems [35], accuracy of touch input [165], and accuracy of recognition of hand gestures in computer vision systems (as used for Publication II). However, to my knowledge, no text input method has yet been optimized to minimize sensing problems.

### 4.4.2 Disambiguation

Ambiguous text input methods require the user to perform the same input action to enter two distinct symbols [92]. They allow, for example, the design of smaller keyboards with larger buttons. A *disambiguation* algorithm is used to predict the most likely word entered, on the basis of a language model. A common case is grouping keyboards, such as the nine keys of the telephone keypad, for which the T9 algorithm\(^1\) is often used for disambiguation. After a sequence of key presses, the algorithm searches a word dictionary for all words corresponding to the sequence entered and orders them by usage frequency. Character-level algorithms have been proposed that predict which character has been entered by looking at the $n$ previously typed characters. This can be a fixed number, as in the use of trigrams or quadgrams for prediction [37], or depend on the number of previously typed characters belonging to the same word [80].

There have been efforts to optimize the one-to-many assignment of input actions to symbols by minimizing the frequency with which a character or word is wrongly predicted [28, 46, 80, 81]. For example, Lesher *et al.* proposed computing a confusability matrix $C$ that captures how frequently a symbol $i$ is predicted to be a symbol $j$ for the given sample text and character-level disambiguation algorithm [80]. The objective is then to minimize the confusability of characters mapped to the same key, which

\(^1\) Originally developed by Tegic Communications, now a part of Nuance Communications (see [http://www.nuance.de/for-business/by-product/t9](http://www.nuance.de/for-business/by-product/t9)).
can be formulated as the following quadratic assignment problem:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} (C_{ij} + C_{ji})x_{ik}x_{jk}$$

(4.26)

Note that in comparison to other quadratic assignment problems presented here, this one only considers quadratic terms that denote that two symbols are assigned to the same input action (both decision variables are indexed by the same $k$). This significantly reduces its complexity.

### 4.5 Summary

This chapter has presented objective functions that allow optimizing the assignment of symbols to input actions with respect to motor performance, ergonomics, learnability, and input recognition. Here, I have proposed several novel objectives and shown that the assignment problem can be used to formulate models and objectives from prior work by means of integer programming. This contribution enables us to use any of the commercially or freely available mathematical solvers to find the optimal solution and obtain guarantees as to the goodness of intermediary outcomes.

State-of-the-art keyboard layout optimization is restricted mostly to the physical keyboard and soft keyboards operated by one end-effector or in touch typing. In contrast, the objective functions proposed here span a wide range of input methods operated by one or several fingers. Although most are targeted at a specific text-input method, I offer discussion of their applicability to the optimization of other methods by instantiating the parameters with the respective models. This expands the space of optimizable text-input methods.

The formulation of proposed objectives proceeds from empirical evidence and models from my own work, criteria from prior work, or theories and heuristics found in the literature. In many cases, I have formulated concepts that have never been considered before in the optimization of text-entry methods. While those objectives are based on peer-reviewed research and well-established findings, additional work is needed to validate their applicability in optimization frameworks. Formulating them as assignment problems allows their easy combination in a multi-objective optimization approach that accounts more fully for the diverse elements influencing the process of text input and thereby aids in creating better designs than available before. The application of these objectives will be illustrated in the next chapter with optimization for three example cases.
<table>
<thead>
<tr>
<th>Input recognition</th>
<th>Objective function</th>
<th>Equation</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autocorrection</td>
<td>( \min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} b_{ij}N_{k\ell}x_{ik}x_{jl} )</td>
<td>4.23</td>
<td>[27, 28]</td>
</tr>
<tr>
<td></td>
<td>- generalized</td>
<td>( \min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} b_{ij}C_{\ell\ell}x_{ik}x_{jl} )</td>
<td>4.24</td>
</tr>
<tr>
<td>Input uncertainty</td>
<td>( \min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} b_{ij}U_{k\ell}x_{ik}x_{jl} )</td>
<td>4.25</td>
<td>–</td>
</tr>
<tr>
<td>Disambiguation</td>
<td>( \min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} (C_{ij} + C_{ji})x_{ik}x_{jk} )</td>
<td>4.26</td>
<td>[80]</td>
</tr>
</tbody>
</table>

Table 4.6. An overview of the objective functions presented in Section 4.4. These allow one to optimize the assignment of symbols to input actions with respect to accurate recognition or prediction of the users’ input. The formulations for the objective functions are based on models and objectives from prior work referenced in the rightmost column or are original formulations I have created.
5. Application of Objectives to the Optimization of Novel Text Input Methods

The goal in my work has been to advance the field of text entry optimization by expanding the space of optimizable text input methods and proposing new objective functions that capture the complex processes involved in text input. Therefore, with the previous chapter I have shown how to express models and heuristics as objective functions, going beyond the criteria considered by prior work. In this chapter, I present results from applying these objective functions to the optimization of novel text input methods that could not be optimized before. I offer three optimization cases, with the central focus of each as follows:

1. **Skill transfer:** The aim was to utilize a user’s existing motor skills from another domain (specifically, playing the piano) for reduction of the learning time and to allow high-performance text input. The novelty lies in the formulation of the letter-assignment problem as the problem of finding the best assignment with respect to two statistical distributions from different domains. The resulting design utilizes frequent input actions from the non-text entry domain to enable fast text input. This case has been presented in Publication I.

2. **Mid-air chording gestures:** The second case entailed venturing beyond the realm of keyboard layouts and spatial arrangement of symbols as considered in prior work, to addressing input via multi-finger chording gestures performed in mid-air. The optimization of mid-air gestures is possible only through the generalization presented at the beginning of this thesis: the letter-assignment problem models the design problem of finding the best assignment of symbols to input actions (rather than that of letters to keys on the keyboard). This case, in turn, has been addressed in Publication II.
3. **Entry of special characters:** While previous work has restricted itself mostly to optimizing the location of the 26 letters of the English alphabet, with this case I considered a large-scale real-world scenario. The goal was to optimally assign numerous special characters (> 100) to key slots for the physical keyboard, taking into account performance and ergonomic criteria but also the intuitiveness of the layout. The input of special characters is different from typing normal letters so requires different optimization models, not considered in prior work. My results for this case are unpublished thus far but have been proposed as an approach to the development of a new French keyboard standard, an effort led by the French standardization organization – Association française de normalisation (AFNOR) – which was still in progress at the time of submission of the thesis.

Here, I will firstly define the design task for each of these cases, in turn, as an instance of the letter-assignment problem in terms of (1) the design space and its constraints, (2) the objective function as a weighted linear combination of multiple objectives, and (3) the task instance – the task-specific parameters and input data. I will then describe the methods I used to solve the optimization problem and present the final outcome.

Each case necessitates considering multiple objective criteria related to the performance with the input method and to that method’s learnability and ergonomics. Some of these objectives may be in mutual conflict. To find the best balance among the objectives, I combine them in a linear sum, where each objective is weighted by its relative importance. This yields one objective function, which can then be optimized via methods of single-objective optimization. Changing the weights and re-solving the problem also allows one to further explore the space of Pareto-optimal solutions. The weighted sum approach has the advantage that it can be easily applied even to computationally complex optimization tasks, such as the quadratic assignment problem. Its benefits and shortcomings are explained in greater depth in Subsection 2.2.3. There are other approaches to multi-objective optimization too, which are reviewed by Marler and Arora [101], among others. Exploring their suitability for user-interface design, while beyond the scope of this thesis, is an important topic for future work.
5.1 Transfer of Motor Skills

When performing music, pianists are able to accurately coordinate the whole musculoskeletal system of the upper extremities to control the position, force, and rhythm of key presses. At the same time, they reach astonishing keying rates. For example, when playing “Flight of the Bumblebee” at regular speed, a pianist has to press 17 keys per second. Such a rate would translate to 204 WPM. This prompts us to ask whether we can utilize that motor skill for efficient entry of text via the piano.

I explored this question with Publication I. The goal was to find an optimal assignment of key presses on the piano to letters and letter sequences in English. In order to make use of existing motor skills in the piano-playing domain, the assignment maximized the frequency of input actions that are used for playing music. This required the objective function to consider two frequency distributions, that of music and that of language, whereas prior work in text entry optimization considered only the statistical distribution of language. The resulting skill transfer allowed experienced pianists to adopt the method very quickly. In addition, I optimized the movement distance of the hands and fingers, so as to improve the achievable input performance even further. The design made use of redundancy and chords, two concepts inspired by music and piano playing. These have been exploited by prior text input methods [89, 115] but not in this combination. Redundancy allows letters to be assigned to multiple keys for purposes of reducing travel distance on the long piano keyboard. In a chord, several keys are pressed at once, as is common for creating harmonies in a piece of music. When entering text, the user employs them as shortcuts, for entering frequently used letter sequences and words quickly with only one input action.

The resulting design, called PianoText, introduced two components: (1) an optimized letter mapping, wherein each letter is mapped to one or more piano keys, and (2) a chord mapping that assigns frequent musical chords to frequent letter sequences (n-grams, whether words or syllables). In a feature exploiting the periodicity of the piano, a chord can be entered in any octave. This further reduced the hand-travel distance.

In Publication I, I reported on two user studies that evaluated the performance achievable and the design’s learnability. They show that PianoText can double the rates reported earlier for piano-based typing. It exceeds 80 WPM, comparable to expert use of a physical Qwerty keyboard. In
the following discussion, I will define the design task as an instance of the letter-assignment problem and briefly describe both the optimization method and evaluation of the outcome. My emphasis is on the design of the letter mapping, which is more complex than the chord mapping. A more detailed description of both can be found in Publication I.

Note that the original work did not explicitly formulate the design task as an assignment problem. I reformulate it here to show the applicability of the objectives presented in the previous chapter and of the letter-assignment problem as a unified formalization for designing text input methods. Were this formulation to have been used in the original work, exact mathematical solvers could have been utilized to solve the problem and potentially design an even better assignment. Instead, I developed a greedy algorithm that constructed an assignment based on heuristics, optimizing the two objectives step-wise, initially for transfer of expertise and then for minimizing the movement distance. This method was suitable for constructing a “good enough” mapping but could not guarantee the solution being in the Pareto-optimal set. Moreover, using the formulation of the letter-assignment problem and the weighted sum method would have allowed exploring different tradeoffs between the two objectives.

5.1.1 The design space and constraints

The design task is to find an assignment between the $N = 26$ letters of the English alphabet and the $M = 88$ keys of the piano keyboard. The design space $X$ consists of all possible combinations of decisions, expressed as the design vector $x = (x_{11}, x_{12}, \ldots, x_{NM}) \in X$, where $x_{ik}$ is the binary decision variable that refers to whether a letter $i$ is assigned to a key $k$ or not. Then, the feasible solution space is defined by the following constraints:

- Each letter must be assigned to at least one key (but may be assigned to multiple keys):
  \[ \sum_{k=1}^{M} x_{ik} \geq 1 \quad \forall i \in \{1, \ldots, N\} \tag{5.1} \]

- Each key is to be assigned to only one letter at most (but does not have to be assigned to any):
  \[ \sum_{i=1}^{N} x_{ik} \leq 1 \quad \forall k \in \{1, \ldots, M\} \tag{5.2} \]

This design space is very large. Requiring each letter to be mapped exactly once already yields \(\frac{88!}{(88-26)!}\) possible assignments. The remaining 62 keys
can be mapped to any (or no) letter, yielding $27^{62}$ ways to assign them. Together, this multiply out to more than $10^{137}$ feasible solutions.

5.1.2 Objectives and the optimization model

The design for this case should minimize the movement time between two successive key presses, alongside the learning time, by exploiting the existing motor skills of a pianist. Therefore, I formulate the objective function as a linear combination of two objectives in line with what was presented in Chapter 4:

**Motor performance**

The goal is to minimize the distance between consecutive key presses performed by the same hand. We assume here that the performance for pressing two keys on the piano is directly correlated with the distance between these keys. That simplification is necessary since there are no quantitative predictive models for movements on the piano. This allows me to use the objective as formulated in Equation 4.4:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} D_{kl} x_{ik} x_{jl}$$

where $p_{ij}$ denotes the frequency of letter $j$ following symbol $i$ and where $D_{kl}$ denotes the distance between the keys $k$ and $l$ on the piano keyboard.

**Learnability**

Frequent symbol pairs should be mapped to frequent intervals (note pairs) in music. This objective stems from the hypothesis that frequently encountered intervals in music afford the fastest response when one is entering text. Taking advantage of well-practiced movements reduces the need to learn new motor skills and thereby decreases training time. This is modeled by Equation 4.17, in the previous chapter:

$$\max \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} f_{kl} x_{ik} x_{jl}$$

where $p_{ij}$ denotes the frequency of the symbol pair $i$–$j$ and $f_{kl}$ that of the note pair $k$–$l$.

Taken together with the constraints defined above, combining the two objectives into one objective function yields the following optimization model, which I formulate as an integer program:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} w_P p_{ij} D_{kl} x_{ik} x_{jl} - w_L p_{ij} f_{kl} x_{ik} x_{jl}$$

(5.3)
subject to \[ \sum_{k=1}^{M} x_{ik} \geq 1 \quad \forall i \in \{1, \ldots, N\} \] (5.4)

\[ \sum_{i=1}^{N} x_{ik} \leq 1 \quad \forall k \in \{1, \ldots, M\} \] (5.5)

\[ x_{ik} \in 0, 1 \quad \forall i \in \{1, \ldots, N\}, k \in \{1, \ldots, M\} \] (5.6)

where \( w_P, w_L \) specify the relative importance of each objective.

5.1.3 The task instance

To arrive at a solution to the actual design problem, we need to instantiate the optimization model with concrete data and parameters that capture the characteristics of the problem. Therefore, I collected the following input data.

\( p_{ij} \): The frequency distribution of English

I obtained the frequency distributions of letters and letter pairs (bigrams) in English from a corpus of classic literature [24].

\( f_{kl} \): The frequency distribution of music

The probability of a key \( l \) getting pressed after key \( k \) is determined by the frequency of occurrence of the corresponding note pair in playing of music. I obtained the distribution of notes and note pairs from sheet music in 10 sight-reading practice books. These books contain basic musical structures common to many genres and types of pieces so correspond well to what is practiced by pianists.

Figure 5.1 shows the distributions for English and music. One can see that the keys toward the middle of the piano keyboard are used more frequently. The narrow diagonal band indicates that note transitions in music are composed only up to a certain distance. The categorical organization of letters does not allow easy matching between the two frequency distributions in an assignment of letters to piano keys. However, this is crucial for allowing a pianist to take advantage of the domain-specific motor expertise, which is based on frequent note transitions in music. The introduction of redundant keys facilitates this task.

\( D_{kl} \): Distance between keys

I computed the 1D euclidean distance between two keys \( k \) and \( l \). This is a simplification, disregarding the fact that the black keys on the piano (half tones) are shorter than the white keys, for a grid with 1.5 rows.
The optimization model given above formulates the design task as a quadratic assignment problem. As described in Section 2.2, there are many heuristic or exact methods for finding a good approximate solution to this NP-hard problem. For Publication I, I was unable to formulate the problem as an integer program; hence, it was impossible to use exact methods. Instead, I used a heuristic approach. I developed a greedy algorithm to construct a letter-to-key mapping, following three steps:

1. Going through the letters in order of their frequency, firstly assign the most frequent letter to the most frequent note.

2. Then, check all frequent bigrams containing that letter for whether they correspond to a note pair that occurs frequently in music. If one does not, the letter is reassigned, to the next most frequent note.

3. When all letters have been mapped once, go through all frequent letter pairs and check whether the distance between the corresponding keys is below a given threshold. If this is not the case, an additional key is assigned for the more frequent letter, to reduce the distance.

The step-wise assignment gives preference to the learnability objective over the performance one without there being a need to specify explicit weights as required by the integer program. This preference is justified by the fact that the transfer of expertise not only is beneficial for learnability but also can be assumed to improve performance, since frequently occurring note pairs afford the fastest response by pianists. In light of this, the distance
minimization is regarded as only a secondary objective. On the other hand, this does not allow exploring different tradeoffs between the two objectives.

The algorithm is applied twice, once for the left and once for the right side of the keyboard. This ensures that most letters are assigned at least two times and are easily accessed by each hand. In the manner described above, the letter mapping is extended by a chord mapping, which assigns the most frequent $n$-grams in English to two- and three-key chords of the major and minor scales in music in accordance with the predicted performance gain conferred. See Publication I for more details on the mapping procedure.

5.1.5 Outcome

The resulting assignment is shown in Figure 5.2. In it, 55 keys are used and each letter is assigned, on average, 2.1 times, with the most frequent letters, such as e, being mapped to four keys. This reduces the average distance for frequent bigrams to six keys, producing a major/minor sixth, an interval encountered frequently in music. Figure 5.3 shows that the frequency distribution of note pairs in typing via PianoText is a good match for that of the note transitions in music. The diagonal structure in both plots reveals that note pairs of a similar distance are used, where keys toward the center of the piano keyboard are more frequent. This allows pianists to exploit their existing motor skills when entering text with the piano keyboard. In Publication I, I evaluated the performance and learnability of PianoText further through two studies. The main findings from these are briefly presented below.

Study 1: Skill transfer and mapping comparison

The purpose of the first study was to assess whether PianoText allows pianists to take advantage of the existing domain motor skills to efficiently enter text with the piano keyboard. To this end, I designed an experiment to measure the input performance achievable in a case involving no prior practice. I compared two optimized assignments, the PianoText assignment as described above and an alternative outcome wherein the optimization
algorithm was applied only once, to all 88 keys. Two variations of these letter assignments were tested, once with and once without the additional chord mapping. I created a music transcription task in which eight phrases from the Enron email dataset [160] were translated into sheet music in accordance with the mappings. These were then played by a professional sight-reader, a pianist trained in playing music directly from the sheet without practicing beforehand. Such a task emulates skilled-level performance and allowed me to estimate the expert performance achievable with piano-based typing. Because of the rarity of such pianists, the study was done with only one participant, a lecturer at the local university of music who participated without remuneration. The pianist was instructed to play as quickly and accurately as possible and to disregard tempo and rhythm, which is uncommon in music performances. Therefore, in each condition, he was initially given four practice sheets, of increasing complexity, to adapt to the atonality of the music and the goal of rapid playing. Then, he played from two previously unseen sheets. For each, 10 s after the notes were shown to him, he had to start playing immediately, and then he repeated the sheet, playing a second time. The performance was measured in words per minute, with an assumption of five characters per word. The error rate was computed as the proportion of letter-level omission, commission, and substitution errors, on the basis of MIDI recordings.

The best performance was achieved with the mapping implemented in PianoText extended by the chord mapping, which yielded an average performance of 71 WPM in the first trial and up to 84 WPM in the second. Chords sped up performance at the cost of higher error rate (6.8% vs.
Application of Objectives to the Optimization of Novel Text Input Methods

2.97%). The performance gain was highest for chords assigned to longer letter sequences. The PianoText assignment allowed user transcription of text that was 16% faster than that with the other assignment tested, even though the two used the same number of keys and of chords. This shows the benefit of redundant letters mapped to keys from each side of the piano, which enables hand alternation at any time. The professional pianist reported that the translated music was similar to atonal pieces he had played and therefore easy to adapt to. He saw the main challenge as lying in disregarding the rhythm aspect and playing notes as rapidly as possible, which is uncommon in music.

In summary, the study showed that PianoText allows pianists to take advantage of their existing motor skills to achieve a high text input rate, comparable to that of expert typists [52]. The sheets translated into musical notation favor note structures that are common in music and hence are cognitively as well as motorically fast to respond to without any practice in the technique.

Study 2: Learnability and skilled performance
The purpose with the second study was to assess the learnability of PianoText, given the large number of keys and chords, and to assess the performance level achievable after practice. Therefore, I conducted a long-term experiment in which I trained a hobby pianist to memorize the letter and chord mappings of PianoText and practice text input, eliminating the need to translate the text into sheet music before playing. This required the pianist to resolve the redundancy herself since the keys to be pressed were not determined by a sheet of music. On the other hand, this allowed her to choose the movements she was most familiar with, whereas the pianist in study 1 was restricted to the notes dictated by the sheet music.

Overall, the pianist trained for 140 hours, which is roughly comparable to the intermediate range of typing expertise with a standard keyboard [140]. Her typing performance with a standard keyboard was 72 WPM for typing German and 44 WPM for English.

The pianist trained for, on average, six hours per week over a duration of 25 weeks. I developed a training program in Java to display and control the exercises. They increased in complexity, beginning with memorizing the letter assignment and proceeding to practicing efficient ways to enter frequent letter sequences, words, and phrases. Chords were introduced during the seventh week, and the participant was trained to memorize
Application of Objectives to the Optimization of Novel Text Input Methods

and practice the use of 4–10 new chords per week. Exercise sessions were self-paced, without the presence of an experimenter, and new exercises were given on the basis of the participant’s feedback and performance in the previous week. Controlled performance assessments were conducted once a week. The participant had to transcribe 35–40 randomly presented sentences from the Enron email dataset [160], which had never been encountered during practice sessions. Performance was measured in words per minute with an assumption of five characters per word. The error rate was measured in terms of the Damerau–Levenshtein edit distance. More details are provided in Publication I.

After 140 hours of training, the pianist was able to consistently transcribe text at a performance rate of over 80 WPM with an error rate of 4%. In comparison, it takes in excess of 200 hours of deliberate practice to reach a level of 50 WPM on a standard keyboard [140]. Transfer of motor skills played a significant role in this development. I compared the inter-key interval (IKI) of letter pairs between the very first test, conducted after the pianist was able to memorize the full mapping, and the second test. For very frequent letter pairs corresponding to note transitions common in music, the IKI dropped from above 500 ms to expert-rate levels below 100 ms. This shows that only the cognitive training in linking letter pairs to musical intervals was needed for reaching locally optimal performance; the motor skill was successfully transferred from music playing. Similarly, the pianist could quickly implement “touch typing” after having been instructed to focus on the screen while transcribing text, rather than look at the hands and keys. Since piano playing too is carried out without looking at the keys, the pianist could exploit her existing orientation to the piano. As a result, a performance increase of over 10 WPM was visible after only five hours of training, between the eighth and ninth test.

A series of controlled tests conducted over several days at the end of the training revealed that the use of redundant keys improved performance by 18% and reduced hand-travel distance by 26%. At the same time, resolving redundancy was not found to create any additional cost. Rather, the pianist exhibited a fixed response to frequent letter sequences. She utilized redundant keys but always used the same key to enter a letter in a specific context. Moreover, chords were found to be 90% faster than entry of the corresponding letter sequence by means of single key presses, thereby improving performance by 30.2%.
5.2 Mid-air Text Input with Chording Gestures

Computer-vision-based hand tracking has significantly advanced over the last few years. While it was previously limited to recognizing gross movements of the arm and a few basic hand movements, such as pinching, recent work has developed general hand models that allow the recognition of any hand and finger movements [69, 149, 155]. This enables use of the hands and fingers for computer input without a mouse, a keyboard, or any other intermediary device. With their many degrees of freedom (DOFs) and their fast yet precise movements, free-hand input is a promising modality particularly for text input on novel and emerging mobile devices such as smartwatches and other wearables [7, 71, 154], virtual- and augmented-reality systems [104], and large interactive displays [109, 158].

Previous research, while exploring the use of free-hand input for entering text, did not take advantage of the potential of the hands’ dexterity. Most work simply applied methods from 2D to 3D, using a single end-effector for selecting external target keys or performing continuous drawing gestures [100, 136, 148]. This does not exploit the full capacity of the hand, and it is slow and tiring. In Publication II, I examined the use of mid-air chording gestures as defined in Subsection 4.1.2. These make use of several DOFs to express the information of a symbol in one movement that does not require any visual attention. Two example chording gestures are shown in Figure 5.4. The problem is that it is unclear how to assign letters of the alphabet to these gestures. Since a natural mapping does not exist, elicitation studies, as conducted in prior work [106, 118, 158], are
unsuitable. Moreover, they typically do not consider performance aspects of gestures, which are especially important for high-performance tasks such as text input.

The work I report on here is the first to use computational optimization methods to design mid-air gestures. In this part of the chapter, I start by defining the design task of finding the best assignment of letters of the English alphabet to mid-air chording gestures, using four objectives defined in the previous chapter: motor performance, finger individuation, gesture complexity, and memorability. Then, I present details of the experimental studies conducted in connection with Publication II to develop and parameterize these models. Users were asked to move a single finger between two angular configurations as quickly and accurately as possible. This allowed me to assess the performance and individuation of each finger individually. For then generalizing to multi-finger gestures, I built on literature in the domain of motor learning to assume that the performance of multiple fingers moving simultaneously is determined by the slowest joint [64, 130]. Moreover, I observed that individuation constraints do not apply if co-dependent fingers both participate in a gesture. The derivation of models for multi-finger gestures is hence less expensive than assessing the performance of all combinations of fingers separately.

Note that, in a similarity to the case considered in the previous section, for Publication II I did not explicitly formulate the design task as an integer program. I reformulate it here to demonstrate the applicability of the assignment problem and of the objectives developed in the preceding chapter. In this case, the formulation of the letter-assignment problem presented in Chapter 2 had to be extended, to enable incorporation of the costs as defined in the original work. This is explained in the discussion that follows.

Finally, I present the outcome of the optimization approach, which was empirically studied by means of a prototype system for tracking and recognizing the hand with a Leap Motion sensor.\footnote{See https://www.leapmotion.com/}. Participants showed promising input rates, with an average of 22 WPM and peak performance of up to 38 WPM. However, this still falls short of the predicted input performance of over 50 WPM, which might be due to the brief training times and tracking difficulties.
5.2.1 The design space and constraints

Given a set of $M$ chording gestures and $N$ letters of the English alphabet, the design task is to find the best assignment of gestures to letters. I should reiterate here that I use the term “chording gesture” to denote a hand configuration in mid-air wherein a combination of fingers (or joints) is flexed to a discriminable end posture (a target angle) while the other joints remain fully extended. This is further defined in Subsection 4.1.2 and in Publication II.

The number $M$ of chording gestures is given by the number of independent degrees of freedom and the associated discretization level of joint angles. In Publication II, I considered six joints that could be reliably tracked and that form a “class” of input actions: the metacarpophalangeal (MCP) joints of the four fingers, the interphalangeal (IP) joint of the thumb, and the carpometacarpal (CMC) joint of the thumb. Together these span seven degrees of freedom (DOFs), with the CMC joint of the thumb being able to move on two axes (up–down and left–right). The left pane of Figure 5.5 shows the location of these joints, along with the colors and names used in Publication II and the rest of this section. The angular range of each DOF is divided into a discrete number of bins (e.g., five bins in the left pane of Figure 5.5). Two hand configurations whose joint angles fall into the same bin denote the same chording gesture. With seven degrees of freedom and five discretization levels, there are already $M = 5^7 = 78,125$ possible chording gestures.
The design space $X$ of all assignments from letters to gestures consists of all possible combinations of decisions, expressed as the design vector $x = (x_{11}, x_{12}, \ldots, x_{NM}) \in X$, where $x_{ik}$ is the binary decision variable for whether a letter $i$ is assigned to a gesture $k$ or not. Then, the feasible solution space is defined by the following constraints:

- Each letter must be assigned to exactly one gesture:
  \[ \sum_{k=1}^{M} x_{ik} = 1 \quad \forall i \in \{1, \ldots, N\} \quad (5.7) \]

- Each gesture can be assigned to one letter at most (but does not have to be assigned to any):
  \[ \sum_{i=1}^{N} x_{ik} \leq 1 \quad \forall k \in \{1, \ldots, M\} \quad (5.8) \]

Depending on the number of gestures and letters, this design space can be extremely large. There are $\frac{M!}{(M-N)!}$ possible assignments. For the extreme case of assigning 27 letters (26 letters of the English alphabet plus the space character) to gestures defined by seven DOFs and five discretization levels, this comes to more than $10^{132}$ feasible solutions.

### 5.2.2 Objectives and the optimization model

The design goal here is to find an assignment of letters to chording gestures that allows efficient and comfortable text input with hand movements that are easy to learn and memorize. Therefore, I define the objective function as a weighted sum over four criteria. These were formulated and explained in detail in Chapter 4, so I will only briefly summarize them below.

**Movement time**

The first is to minimize the time it takes to move all joints from one chording gesture $k$ to another gesture $l$. I use angular Fitts’ Law models to quantify the time $t_{\theta}^{kl}$ it takes to flex or extend a joint $\theta$ from a starting angle $\alpha_{k,\theta}$ to a target angle $\alpha_{l,\theta}$. This depends on the angular movement distance $D_{\alpha_{k,\theta}\alpha_{l,\theta}}$ and angular width of the target bin $W_{\alpha_{l,\theta}}$, as described by Equation 4.5:

\[ t_{\theta}^{kl} = a_{\theta} + b_{\theta} \log_2 \left( \frac{D_{\alpha_{k,\theta}\alpha_{l,\theta}}}{W_{\alpha_{l,\theta}}} + 1 \right) \]

The Fitts’ Law parameters $a_{\theta}$ and $b_{\theta}$ are determined empirically in the manner explained in the next subsection. The time to move all joints $\theta \in \Theta$ from one configuration to another is dictated by the slowest contributing
joint. This yields the following objective, as defined by Equation 4.6:

$$\min_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} \max_{\theta \in \Theta} (t_{kl}^{\theta}) x_{ik} x_{jl}$$

where $p_{ij}$ is the frequency of the bigram $i$–$j$.

**Finger individuation**

Another objective is to optimize the movement “comfort” by maximizing the individuation with which fingers can be moved as part of a gesture. On account of anatomical constraints, it is not possible to, for example, bend the ring finger without also moving the little and middle finger. Schieber and colleagues [54, 137] have proposed an individuation index to quantify the extent of this effect for individual fingers. I have extended this to multi-finger gestures as described by Equation 4.14:

$$I_k = 1 - \frac{1}{|\Theta|} \sum_{\theta \in \Theta \setminus \Theta_k} \max_{\phi \in \Theta_k} |C_{\theta \phi}|$$

On an intuitive basis, the index quantifies how much, on average, a non-instructed joint $\theta$ moves during the movement of any joint $\phi \in \Theta_k$ that is part of the gesture $k$. This is quantified in terms of the relative coactivation $C_{\theta \phi}$, which represents the correlation between the position of $\theta$ and of $\phi$. An individuation index of 1 denotes that only those joints that are part of the gesture move, whereas an index of 0 means that all joints move to the same extent. The objective is then to maximize the individuation index of frequent gestures. This is formulated via Equation 4.15:

$$\max_{i=1}^{N} \sum_{k=1}^{M} p_i I_k x_{ik}$$

where $p_i$ is the frequency of letter $i$.

**Complexity**

The third aim is to minimize the complexity of the gestures so that they become easier to learn. As discussed in Subsection 4.3.1, I assume on the basis of theories of motor learning [12, 105, 130] that the cost of motor learning is smaller if the gesture does not require operate as many degrees of freedom. Therefore, with the objective I quantify two aspects of a gesture $k$: (1) $|\Theta_k|$, the number of joints involved in the gesture, and (2) $|A_k|$, the number of distinct target angles the joints must reach. This yields the following objective as formulated by means of Equation 4.16:

$$\min_{i=1}^{N} \sum_{k=1}^{M} p_i 0.5 \frac{|\Theta_k| + |A_k|}{|\Theta|} x_{ik}$$

where $p_i$ is the frequency of letter $i$ and $|\Theta|$ is the total number of joints.
Memorability

Finally, we wish to optimize the memorability of a gesture set by maximizing the proportion of gestures that belong to few mnemonic sets. As presented in Subsection 4.3.3, a mnemonic set \( m \in \mathcal{M} \) is a group of gestures that share mnemonic structures such as using neighboring fingers. This commonality makes them easier to memorize and remember [162]. The objective is intended to maximize the number of gestures that belong to a mnemonic set while at the same time minimizing the number of different mnemonic sets the gestures belong to. Equation 4.18 formulates this as follows:

\[
\max \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{m \in \mathcal{M}} \frac{m(k)}{|\mathcal{M}|} x_{ik} - \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{M}|} y_m
\]

where \( \mathcal{M} \) denotes the set of all mnemonic sets, with \( m(k) = 1 \) if gesture \( k \) is in the mnemonic set \( m \) and 0 otherwise. The binary decision variable \( y_m \) specifies whether any of the assigned gestures belong to the mnemonic set \( m \). This is expressed by the following additional constraint:

\[
y_m \geq m(k) \ x_{ik} \quad \forall i = 1 \ldots N, k = 1 \ldots M, m \in \mathcal{M}
\]

More details are given in Subsection 4.3.3.

The four criteria are combined into a weighted sum to yield one objective function. Together with the constraints given above, the following optimization model defines the design task of optimally assigning letters in the English language to mid-air chording gestures:

\[
\begin{align*}
\min \ w_P & \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} p_{ij} \max_{\theta \in \Theta} \left( t_{kl}^{\theta} \right) x_{ik} x_{jl} \\
& - w_A \sum_{i=1}^{N} \sum_{k=1}^{M} p_i I_k x_{ik} \\
+ w_{L} & \sum_{i=1}^{N} \sum_{k=1}^{M} \frac{|\Theta_k| + |A_k|}{|\Theta|} x_{ik} \\
& - w_M \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{m \in \mathcal{M}} \frac{m(k)}{|\mathcal{M}|} x_{ik} - \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{M}|} y_m
\end{align*}
\]

subject to

\[
\begin{align*}
\sum_{k=1}^{M} x_{ik} &= 1 \quad \forall i \in \{1, \ldots N\} \quad (5.11) \\
\sum_{i=1}^{N} x_{ik} &\leq 1 \quad \forall k \in \{1, \ldots M\} \quad (5.12)
\end{align*}
\]
Application of Objectives to the Optimization of Novel Text Input Methods

\[ y_m \geq m(k) \ x_{ik} \quad \forall i \in \{1, \ldots, N\}, k \in \{1, \ldots, M\}, m \in \mathcal{M} \]  
(5.13)

\[ x_{ik} \in \{0, 1\} \quad \forall i \in \{1, \ldots, N\}, k \in \{1, \ldots, M\} \]  
(5.14)

\[ y_m \in \{0, 1\} \quad \forall m \in \mathcal{M} \]  
(5.15)

The parameters \( w_P, w_A, w_L, \) and \( w_M \) are used to define the relative importance of each objective, so these should add up to 1.

5.2.3 The task instance

Publication II describes the data collection for instantiating the optimization model presented above. An empirical study was required for obtaining the performance and individualization parameters, whereas the input data for the complexity and memorability scores could be obtained from the literature or were given by the specifics of the task (e.g., the number of joints and discretization levels). This process is briefly described below.

\( a_{\theta}, b_{\theta} \): Angular Fitts’ Law parameters

To obtain the parameters for the performance models for each joint, I conducted a Fitts’ Law study following common experimental practices in the HCI field [90, 151]. The study was conducted with 13 participants and followed a 7 × 4 within-subjects design with seven DOFs (as described above) and four index-of-difficulty (ID) conditions. All conditions were randomized to minimize order effects. The task was a reciprocal target selection task in one dimension, where a target corresponded to a joint angle. Participants positioned their hand in horizontal orientation above a Leap Motion sensor. Moving a finger up or down moved a cursor on a screen, whose position corresponded to the angular motion of the relevant joint. Participants were instructed to move between the two marked targets as quickly and accurately as possible. To account for anatomical differences, we first determined the movement range of each of a user’s joints and adapted the target widths and distances accordingly. The ranges were uniformly divided into two, three, four, and five bins, and the task was to move between all bin combinations. This resulted in four unique IDs for every user (1, 1.6, 2, and 2.3). More details are given in Publication II.

I followed commonly applied guidelines for analyzing results in Fitts’ Law studies [153] to obtain the parameters \( a_{\theta} \) and \( b_{\theta} \) because the slope and intercept of the linear regression models fit the movement times observed for each ID condition (see Publication II for analysis details and the values recorded). Their fitness score \( R^2 \) ranged from 0.82 to 0.99. One-way
repeated-measures ANOVA showed significant performance differences among the joints. Overall, Index was the fastest and Thumb-IP the slowest, where differences were larger for “easy” movements ($ID = 1$). This can be seen from Figure 5.6a, which shows the models for each DOF.

$C_{\theta \phi}$: Relative coactivation

During the Fitts’ Law study described above, I tracked the full movement of the hand rather than only the endpoints of the instructed joints. This allowed me to assess the involuntary movement occurring elsewhere while the instructed joint was performing the target selection tasks. I followed the protocol set forth by Schieber [137] to calculate the relative coactivation as the slope of the linear regression model fit to the normalized angular values of the non-instructed joint as a function of the position of the instructed joint.

The relative coactivation values and corresponding individuation indices for each joint are given in Publication II. Overall, Thumb-IP was found to be the most individuated joint. The two DOFs of the thumb’s CMC joint (Thumb-Down and Thumb-Right) were closely correlated ($C = 0.69$), indicating that participants could not control these DOFs independently. High coactivation was observed also for Ring during the instructed movement of Middle. This is shown in Figure 5.6b. After this, the relative coactivation values were used to calculate the individuation indices for the chording gestures, as shown above.
$p_i$, $p_{ij}$: Frequency distributions of English
I obtained the frequency distribution of English letters and bigrams from a corpus of classic literature.

$\Theta$, $\mathcal{A}$: Joints and discretization levels
The study described above also informed the selection of joints and discretization levels used to define the set of chording gestures. The robustness of the tracking, along with the performance and individuation data obtained, led me to exclude Thumb-IP and to collapse Thumb-Down and Thumb-Right to one DOF. This yielded $|\Theta| = 5$: the MCP joints of each finger and the CMC joint of the thumb. Publication II presents solutions for several instances, including a case with only three joints and with discretization levels varying from two to five bins per joint. Note, however, that discretization into five angular bins is more speculative, since the tracking might not be reliable enough and it could be hard for users to learn to distinguish reliably among five angular levels.

$\mathcal{M}$: Mnemonic sets
Following prior work [162], I included the following mnemonic sets: neighboring finger (e.g., index and middle finger together), base (e.g., thumb together with any other finger), and single finger.

5.2.4 Optimization technique

For the optimization, I formulated the design task as a quadratic letter-assignment problem, which permits the use of exact mathematical methods for approximating the optimal solution to the NP-hard problem while providing guarantees as to the goodness of such solutions. Originally, Publication II did not formulate the problem as an integer program and a heuristic approach was applied to find a good solution. I used a multi-start local search method that started with a random design and found the local optimum in its neighborhood. The solution is stored as the incumbent, and the search is restarted at another random point. A similar approach has been employed in efforts to optimize soft keyboard layouts [27]. Good solutions were obtained within one day on a cluster computer.

5.2.5 Outcome

In a divergence from that in the previous case, this method relied on the definition of weights specifying the importance of each objective, and
the outcome was sensitive to the definition of these weights. By varying
the weights, one can employ this method to explore the solution space.
However, the focus of the work was more on demonstrating the applicability
of the models developed here for design of mid-air gestures that uses
optimization. The weight combinations were chosen so as to demonstrate
the use of the technique for optimizing diverse designs, with a focus on one
or all of the objectives. Publication II presents several outcomes, optimized
for different character and gesture sets, with the number of joints and
discretization levels varying accordingly. The performance models could be
used to predict the achievable input speed in WPM for each assignment.

NUMPAD only assigns the numbers 0–9 to chording gestures formed by
five joints whose angular range is discretized into two bins \(2^5 = 32\)
gestures). By dint of the small character set, the predicted performance level
is very high, at 113 WPM. The same gestures were used for FASTTYPE
to assign the 27 letters found in English (including the space character).
The assignment was optimized with strong weighting for performance and
memorability. This yielded a predicted input speed of 54.7 WPM. FULL-
TYPE is an instance wherein 48 characters in all (letters, numbers, and
punctuation marks) were assigned to a set of chording gestures defined by
five joints and five discretization levels \(5^5 = 3125\) gestures). The objectives
were weighted evenly, yielding predicted performance of 50.7 WPM. Fi-
nally, THREE TYPE assigned the 27 letters to chording gestures composed
with the three joints having the highest individuation (Thumb, Index, and
Middle), using five discretization levels \(5^3 = 125\) gestures). This mapping
yielded the highest predicted performance level, 65.1 WPM. However, five
discretization levels could not be robustly tracked and would require a long
learning time.

The representation of chording gestures as a combination of joint angles
could be used also for theoretically based evaluation of the objective cri-
teria for existing chording methods. Engelbart’s chording keyboard [32]
was predicted to produce slower performance than any of the optimized
assignments described above \(i.e., 49\) WPM). In addition, I investigated
fingerspelling in American Sign Language, which uses finger gestures
to communicate individual letters, similarly to the chording gestures de-
scribed in this work. When one thus considers only those gestures that
could be assessed via the models presented \(e.g.,\) ones that did not in-
clude movements as part of a letter-gesture), the entry rate is predicted
to be 43.9 WPM. This falls within the 40–45 WPM range reported in the
literature for experienced practitioners [126].

In a preliminary user study with 10 users, I empirically investigated the performance achievable with one of the mappings, FASTType. For this, I followed an accelerated learning protocol as used in prior work [14, 73], wherein randomly presented words are repeatedly entered until performance peaks. While such a method limits the external validity of the findings, it allows exploring the upper limits of performance rates achievable with a system while keeping experimental costs low. A simple prototype system was built to recognizes chording gestures from joint angle data transmitted by the Leap Motion sensor. This system took a combination of dwell times and signal peak detection, converting these into text displayed on the screen (see Publication II). Average peak performance was observed to be around 22 WPM at a character error rate of 2.3%. Large individual-to-individual differences were observed, with average peak performance ranging from 13 to 38 WPM. While these performance rates were promising, they fall well short of the predicted rate of 54.7 WPM. This can be attributed in part to the lack of training but also is a manifestation of tracking problems, which were observed to affect users’ performance.

In addition, I investigated whether performance and individuation are compatible design goals. Optimizing for each objective separately yields different assignments: the performance-optimized design employs fewer multi-joint gestures, whereas the assignments optimized for individuation only or for both performance and individuation utilize more gestures that involve neighboring fingers. That said, the differences in each score are negligible. This provides evidence that performance and ergonomics criteria such as finger individuation can be compatible design goals.

5.3 Special-Character Entry

Most work in the text entry optimization field focuses on the assignment of small character sets. In addition to the 26 letters of the English alphabet, some include a few punctuation marks [14, 44] or optimize the assignment of non-Latin character sets [9, 58, 70, 96], but their size never exceeds 30–40 characters. Yet a glance at the physical keyboard reveals a much larger number of symbols that can be entered. Special characters, such as brackets, currency and mathematical symbols, diacritical marks, and the @ and # sign, may get used frequently. This depends on the input task and user group. Deciding to assign a certain symbol to an input action might
favor one way of typing but compromise that of another group. Assignment of special characters requires particular care because the frequency of their use varies significantly across tasks and user groups. Moreover, the large number of characters might require the use of slow or unergonomic input actions, which could be left unassigned in cases of smaller character sets.

I investigated the question of how to best assign a large number of special characters (e.g., > 100) to key slots for the physical keyboard. The work was motivated by a real-world application case: the French standardization organization AFNOR decided to define a standard for the French keyboard layout, with the goal being to facilitate typing of correct French and providing access to frequently used characters of other European languages and input tasks (e.g., programming, mathematical applications, etc.). As of the time of writing, no standard for the layout of the physical keyboard exists in France. What is worse, most keyboards with the so-called Azerty layout (shown in Figure 5.7) do not allow the direct input of basic French characters, such as accented capitals (e.g., À, É, and Ç), French quotation marks («, », ‹, and ›), or ligatures (e.g., œ and æ).

What is presented in this section of the chapter was developed in interaction with a committee of experts from several fields, who all volunteered to lend their expertise to the development of a new keyboard standard. Their expertise informed the definition of the design space, including the character set, objective criteria, and task instances. After I set up the optimization process, it was used to explore various instances and tradeoffs as defined by the committee (e.g., favoring layouts with high input speed or those similar to the current Azerty) in an iterative process. Assessment of the optimized designs by the committee led to changes in the task instances, which then yielded new optimized layouts, and so on. The experts on the committee then proposed final manual swaps and displacements of characters, in order to locally adjust the tradeoffs between criteria. In this step, the objective functions were used to quantify the effect in terms of the evaluation criteria and ensure that no detrimental changes were made. To the best of my knowledge, this work is the first to apply computational optimization methods to inform the design of a national keyboard standard.

At the time this thesis was submitted, the standardization effort was still in progress. While a final design therefore cannot be presented here, I am able to discuss the definition of the design task and collection of input data. I do so below and also present example outcomes of the optimization.
5.3.1 The design space and constraints

The design task here is to find the best assignment of $N$ special characters to $M$ key slots for the physical keyboard with respect to the fixed position of the letters of the alphabet as given by the Azerty layout. The set of characters, defined by the expert committee, comprises up to 122 special characters (as compared to the 46 available in the Azerty layout shown in Figure 5.7). It consists of the following:

- Accented characters and ligatures frequently used in French, such as à, é, ç, ã, and Æ.
- Diacritical marks, such as `', ´, and `, which function as so-called dead keys (i.e., with a corresponding key press, no symbol is displayed; only upon pressing of the modifier-subject letter key does one appear, contingently).
- Punctuation marks, such as ., ;, and !.
- Mathematical, unit, and currency symbols, such as +, -, ×, %, and $.
- Quotation marks used in French and other languages, such as «, », ′, ″, ′′, “, and ”.
- Brackets and parentheses.
- Hyphen and dashes.
- Other elements.

Each key on the keyboard has up to four key slots, which can be used
Application of Objectives to the Optimization of Novel Text Input Methods

for entering four individual symbols either by simply pressing the key (unmodified) or by pressing the key in combination with Shift, the Alt or (especially for Windows) Alt Gr modifier, or both. The set of available key slots is given by the alphanumerical area of the keyboard compliant with the so-called harmonized 48-key keyboard arrangement, as defined by the relevant ISO/IEC standard.\(^2\) Not included are the key slots that are occupied by fixed symbols (e.g., the positions of the letters A–Z and the numbers 0–9 are not changed). This results in up to 129 distinct key slots.

In a parallel to the optimization cases presented in the previous sections, I use binary decision variables \(x_{ik}\) that denote whether a letter \(i\) is assigned to a key slot \(k\) or not. The design space \(X\) consists of all possible combinations of decisions expressed by the design vector \(x = (x_{11}, x_{12}, \ldots, x_{NM}) \in X\). The feasible solution space is then defined by the following constraints:

- Each character must be assigned to exactly one key slot:
  \[
  \sum_{k=1}^{M} x_{ik} = 1 \quad \forall i \in \{1, \ldots, N\} \quad (5.16)
  \]
- Each key slot is to be assigned to only one character at most (but does not have to be assigned to any):
  \[
  \sum_{i=1}^{N} x_{ik} \leq 1 \quad \forall k \in \{1, \ldots, M\} \quad (5.17)
  \]

There are \(\frac{M!}{(M-N)!}\) feasible solutions in the design space. With up to 129 key slots and 122 characters, this weighs in at more than \(10^{213}\) possible keyboard layouts.

5.3.2 Objectives and the optimization model

The goal behind the standard was to define a new keyboard layout that (1) facilitates typing correct French; (2) enables access to more special characters frequently used in European languages, for programming, mathematical expressions, etc.; and (3) is intuitive to use and easy to adapt to. I captured these design goals in an objective function that evaluates the cost of typing a special character in combination with a fixed letter as a weighted sum over four objective criteria. These have been formulated and explained in detail in Chapter 4 and are summarized below.

Movement time

The first is to minimize the movement time required for entry of special characters. As observed in Subsection 4.1.3, special characters are fre-

\(^2\) See https://www.iso.org/standard/51644.html.
quentely entered before or after a “normal” letter of the alphabet. Therefore, to evaluate the performance achievable with a certain keyboard layout, I add up the times $T_{kc}$ ($T_{ck}$) for moving from a key slot $k$ to any of the other 26 letters $c$ or the space bar, then back, as formulated by Equation 4.9:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{c=1}^{27} (p_{ci} T_{ck} + p_{ic} T_{kc}) x_{ik}$$

where $p_{ci}$ ($p_{ic}$) denotes the frequency of the pairs $c-i$ ($i-c$). Note that the movement time is not symmetrical; that is, $T_{kc} \neq T_{ck}$. The objective disregards movements between special characters. Since the assignment of letters is fixed, the objective function is linear in its decision variables. This greatly reduces the complexity of the design problem in comparison to a quadratic formulation.

**Strain**

Secondly, we want to minimize the frequency of repetitive and extreme movements of the wrist and fingers. As was explained in Subsection 4.2.1, such movements put strain on the tendons and joints, which can cause repetitive-strain injuries, such as carpal tunnel syndrome [175]. Moreover, high input speed can be sustained for a longer time only if the movements are performed ergonomically. The objective, as given by Equation 4.11, computes an ergonomics score for pressing a key on the physical keyboard. It equally penalizes extreme outward or inward movements of the wrist (for example, to reach keys on the far left or right), flexion and extension of fingers (for example, toward the upper row), and the use of one or two modifier keys. This is formulated as follows:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_{i}(W_{k} + F_{k} + M_{k}) x_{ik}$$

where $W_{k}$ and $F_{k}$ are binary variables denoting extreme movements of the wrist and fingers as described above, while $M_{k}$ represents the number of modifier keys used.

**Intuitiveness**

The third objective is to minimize the distance between similar characters and thereby ensure a consistent and intuitive layout that is easy to use and learn. As was explained in detail in Subsection 4.3.4, the spatial proximity of similar characters is particularly important for infrequently used characters, for which users rely on visual search for the relevant position on the keyboard. Placing them in the vicinity of more frequent, similar characters
facilitates discovery and learning of the layout. Accordingly, Equation 4.20
minimizes the distance $D_{kl}$ between key slots $k$ and $l$, weighted by the
similarity $s_{ij}$ between the special characters $i$ and $j$ assigned to them. In
addition, I want to minimize the distance between characters and letters
similar to them. Thus, the objective is a combination of a linear and a
quadratic term:
\[
\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} s_{ij} D_{kl} x_{ik} x_{jl} + \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{c=1}^{27} s_{ic} D_{kc} x_{ik} \tag{5.18}
\]
where $D_{kc}$ denotes the distance between the key slot $k$ and the letter $c$,
while $s_{ic}$ refers to the similarity between character $i$ and letter $c$. Note
that, although this is a quadratic function, the number of quadratic terms
can be kept small since the similarity score is 0 for most character pairs
(see below).

**Familiarity**

Finally, we want to minimize the distance of a character from its position
in the Azerty layout. As was discussed in Subsection 4.3.4, a familiar
organization of characters decreases visual search time and thus speeds
up learning of a new layout. When visually searching for a character in
a layout new to them, users start by looking at the previously learned
position [63]. If the character is placed in the vicinity of that, it can be
found sooner. Therefore, the goal with this objective is to minimize the
distance between the key slot $k$ assigned to character $i$ and the position
of that character in the Azerty layout, denoted by $\mathcal{L}(i)$. This was formulated
via Equation 4.21.
\[
\min \sum_{i=1}^{N} \sum_{k=1}^{M} p_i D_{k \mathcal{L}(i)} x_{ik} \tag{5.19}
\]
where $p_i$ denotes the frequency of character $i$.

The optimization model is then given by the constraints discussed above
in combination with an objective function consisting of a weighted sum of
the four objective criteria. Thus, the design task of optimally assigning
special characters to key slots for the physical keyboard with respect to a
fixed set of letters can be modeled as follows:

$$\begin{align*}
\min & \ w_P \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{c=1}^{27} (p_{ci}T_{ck} + p_{ci}T_{kc})x_{ik} \\
+ & \ w_S \sum_{i=1}^{N} \sum_{k=1}^{M} p_i(W_k + F_k + M_k)x_{ik} \\
+ & \ w_I \left( \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} s_{ij}D_{kl}x_{ik}x_{jl} + \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{c=1}^{27} s_{ic}D_{kc}x_{ik} \right) \\
+ & \ w_F \sum_{i=1}^{N} \sum_{k=1}^{M} p_iD_{k\ell(i)}x_{ik} \\
\text{subject to} & \ (5.20) \\
\sum_{k=1}^{M} x_{ik} = 1 & \ \forall i \in \{1, \ldots N\} \quad (5.22) \\
\sum_{i=1}^{N} x_{ik} \leq 1 & \ \forall k \in \{1, \ldots M\} \quad (5.23) \\
x_{ik} \in \{0, 1\} & \ \forall i \in \{1, \ldots N\}, k \in \{1, \ldots M\} \quad (5.24)
\end{align*}$$

For similarity to how a–z and A–Z are entered, further constraints can be added to ensure that other lowercase–uppercase character pairs (e.g., æ and &) are assigned to the same position with the difference of only the Shift key. The constraints applied for this case can be formulated thus:

$$\begin{align*}
\forall i, j \in \{1, \ldots N\}, \text{where } j = \text{upper}(i) & \\
\forall k, l \in \{1, \ldots M\}, \text{where } l = \text{shifted}(k) \\
x_{jk} = 0 & \quad (5.25) \\
x_{il} = 0 & \quad (5.26) \\
x_{ik} = x_{jl} & \quad (5.27)
\end{align*}$$

The first and second constraint ensure that uppercase characters are not assigned to non-Shift key slots and that lowercase characters do not get assigned to Shift slots. The third dictates that if a lowercase character is placed in a given key slot, the corresponding uppercase character is placed in the same position but with Shift applied.

### 5.3.3 The task instance

In collaboration with the expert committee, several task instances were defined in the course of the design process, differing in their character
Application of Objectives to the Optimization of Novel Text Input Methods

set, objective weights, and capitalization constraints. The outcome of optimizing these instances, in turn, informed the creation of new ones. For all of these, the work utilized the same input data capturing the characteristics of the general design task, which are described next.

$p_i$, $p_{ci}$, and $p_{ic}$: Frequency distributions of French

The frequency of special characters, letters, and bigrams was obtained from various corpora reflecting realistic contemporary use of French. In collaboration with the expert committee, bodies of text were gathered in three distinct categories:

1. **Formal:** This set is focused on correct spelling and use of special characters. It is composed of materials in French either prepared by professional typists or curated by a dedicated community on a day-to-day basis. It includes all French Wikipedia articles prior to June 2014, legal texts about environmental and work-related legislation, written questions submitted to the *EU Official Journal* and the corresponding answers, transcripts of radio broadcasts, anonymized email messages, and news articles from well-regarded French newspapers.

2. **Popular:** The second set contains everyday expressions, use of emerging vocabulary, and special characters. It consists of 1,000 posts, from 10 popular Twitter and 10 popular Facebook accounts, made in French. This material was collected in August 2016. So as to reflect real-world use, posts were not post-curated for content, grammatical correctness, or use of French symbols. While URLs were observed to occur with high frequency in tweets, they can be expected to not be entered manually but get copy-pasted. Therefore, 75% of the URLs in tweets were selected at random for removal from the corpus for purposes of not biasing the dataset.

3. **Code:** With the final corpus, emphasis is on the use of programming-related characters. It consists of text files in six common programming or layout languages: C++, Java, Python, JavaScript, HTML, and CSS. They were extracted from 10 public GitHub projects per language, with at least 1000 lines of code each. The contents of the comments were removed, while the comment delimiters were not.

Corpora were either compiled through web crawling or provided by the European Language Resources Association (ELRA). Frequencies of characters, letters, and bigrams were calculated for each corpus separately.
Table 5.1. For each category of text, the 20 most frequent special characters and their frequency.

and then averaged within a category. The frequency of special characters varied significantly between the various categories. Table 5.1 compares the 20 most frequent special characters per category. The table reveals large differences, highlighting the importance of choosing corpora that reflect realistic usage of the keyboard. The final distribution was then obtained by assigning a weight to each corpus. The weights, reflecting the importance accorded to each category in the optimization process, were defined by the expert committee. They varied with the scenario optimized for.

$T_{ck}, T_{kc}$: Movement times

The time to move between a fixed letter key and a key slot to be assigned was determined experimentally. In a crowdsourcing-based study, I gathered movement times for all transitions between the 27 regular alphabetic keys on a French keyboard (a–z, plus the space bar) and the 140 special-character key slots (the 44 unmodified and key+Shift slots not occupied by a–z and A–Z and the 96 key slots accessible via Alt and Alt+Shift). This yields 7,560 distinct transitions.

So that I could obtain movement times that reflect realistic typing behavior, I developed the following transcription task. Participants were shown stimuli consisting of two four-letter sequences that were pronounceable within the English syllable structure (pseudo-syllables). These were sepa-
The transcription task in the study consisted of repeatedly typing two pseudo-syllables connected by a special-character key slot. With this example, we gathered data about two transitions: (1) from the letter w to the key slot corresponding to Shift+Alt+f on the participant’s keyboard and (2) between that slot and the space bar (represented by ).

Rated by a special-character slot, thereby yielding performance data for two transitions per stimulus. The key slot was represented as the unmodified character on the relevant key within the participant’s keyboard layout. Modifier keys were identified explicitly. See Figure 5.8 for an example. For each letter a–z, I generated five pseudo-syllables starting and five ending with the respective letters. They were created randomly in accordance with the character distribution in English, with some manual curation to remove syllables that form actual words or that looked unnatural. For transitions to and from the space bar, we picked a random syllable and replaced the first or last character with a space.

The use of pronounceable pseudo-syllables ensured that people employed their usual typing behavior and that effects of language familiarity did not emerge. Prior research showed that skilled-level typing performance did not change for non-words in comparison to words if the bigram frequency was preserved [76]. Therefore, the movement times observed for special characters should reflect realistic performance, similar to what one can expect from French users.

I recruited more than 900 international participants, from all over the world. Over 630 of them were recruited and paid through a crowdsourcing website, and approximately 270 were unpaid participants attracted by advertising the study on a webpage that offers typing tests. To make the study accessible to a large user group, instructions were presented in English. Before starting the study, we recorded which keyboard layout the participants were using and tested their regular typing speed through an initial test wherein they had to transcribe 10 English-language phrases. Participants then had to transcribe 10 (for unpaid volunteers) or 20 (for

3 At https://www.crowdflower.com/.
4 This is https://www.typingtest.com/.
Table 5.2. Criteria for rating the similarity between two symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Score</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capitals</td>
<td>1</td>
<td>One character is the uppercase version of another</td>
<td>œ CE</td>
</tr>
<tr>
<td>Semantic proximity</td>
<td>0.9</td>
<td>The symbols have similar meanings</td>
<td>· × *</td>
</tr>
<tr>
<td>Inclusion</td>
<td>0.9</td>
<td>One symbol is part of another one</td>
<td>a a in c ç</td>
</tr>
<tr>
<td>Completion</td>
<td>0.9</td>
<td>Symbols complete or mirror each other</td>
<td>( ) « »</td>
</tr>
<tr>
<td>Use</td>
<td>0.7</td>
<td>Symbols are associated in use or custom</td>
<td>n ´ c ,</td>
</tr>
<tr>
<td>Alphabet</td>
<td>0.6</td>
<td>A character from another alphabet is equivalent to a French one</td>
<td>ß s</td>
</tr>
<tr>
<td>Grouping</td>
<td>0.6</td>
<td>Symbols in a set can be associated with a specific use</td>
<td>- = - $ £ €</td>
</tr>
<tr>
<td>Visual similarity</td>
<td>0.5</td>
<td>The symbols look like each other</td>
<td>ß ∞</td>
</tr>
<tr>
<td>Phonetic similarity</td>
<td>0.4</td>
<td>The symbols are pronounced similarly or in the same way</td>
<td>s ß r √</td>
</tr>
</tbody>
</table>

paid CrowdFlower users) stimuli each. For each possible transition, we gathered data from at least three participants, with most transitions being typed by more than four participants.

Participants were instructed to correctly transcribe each stimulus 10 times as quickly as possible. Errors could not be corrected. To obtain the movement time between a letter and special-character key slot for a user, I averaged their five fastest attempts. This excludes repetitions wherein the user was learning or became distracted and thereby approximates experienced input performance. This is similar to the accelerated learning protocol used in prior work [14, 73]. Before averaging, individual-specific times were rounded to the nearest 30 ms to smooth out fluctuations. The user’s average was then normalized in line with his or her regular typing speed. This prevented very fast or slow users from overly biasing the results. The movement time for a letter–key-slot pair was then computed as the average over the mean transition times of all users who responded to the stimulus, where we discarded outliers with unusually high (> 1200 ms) or low (< 50 ms) times (cutoffs determined after inspection of the distributions). An overview of the resulting transition times is given in Figure 5.9. Note that this presents only the average time to move between a certain special-character key slot and any of the letter keys. Transitions to or from specific letters, as applied in optimization, might be faster or slower.
Figure 5.9. The average movement time for each special-character key slot used before or after any of the letters a–z or a space character.
Character–character and character–letter similarity

The similarity between two special characters or between a special character and a letter is defined by an expert in accordance with several ad-hoc criteria, each associated with a similarity score. They are specified in Table 5.2. When a symbol pair fits more than one criterion, it is assigned the maximum of the corresponding similarity scores. For instance, « and ‹ look alike (visual similarity 0.5), both are quotation marks (grouping 0.6), and the two are very similar in meaning (both opening a quotation) (semantic proximity 0.9); therefore, the resulting weight for this association is 0.9.

Defining similarities between symbols soon becomes a subjective process, and it was not an exhaustive one. The symbol pairs rated and their ratings are results of discussions with the expert committee. The resulting intuitiveness objective is, in essence, a guideline for bringing groups of characters together so as to facilitate discoverability.

Distances

The distance between two key slots is measured in this connection via modified Manhattan distance. Similarly to other researchers [27], I quantify the number of rows and columns between two keys. In addition, there is a cost for key slots accessible via any of the various modifiers. This differs with the objective. For the familiarity score, I assume that users search the various levels of the keyboard in the order of usage frequency – that is, first unmodified, then Shift level, then Alt level, and lastly Alt+Shift level. Accordingly, the modifier distance value corresponds to the number of additional levels the users have to search, starting with the level at which the character resides in the Azerty layout. The distance score used for the intuitiveness objective is a simple count value, the difference in the number of modifiers used to access the two key slots.

5.3.4 Optimization technique

As presented above, the design task was formulated as a quadratic assignment problem. This allowed me to use exact methods for finding good approximate solutions that also yield concrete bounds for their goodness. Even though the similarity score used in the intuitiveness objective function (the only quadratic one) is 0 for most character pairs, the large set of characters and key slots also yields a large number of quadratic terms. To cope with this complexity, I applied the commonly used Kaufman–Broeckx linearization method [67] and a recently developed semidefinite
programming relaxation based on cut pseudo bases [62]. Both techniques contributed to solving of the initial problem in a Branch & Bound process. The former led to a relaxation that was beneficial for state-of-the-art heuristics for the primal problem, and the latter produced good lower bounds to prove the near-optimality of the solution. They allowed the use of the commercial solver Gurobi\textsuperscript{5} to find good keyboard layouts in only a few days of runtime while also obtaining mathematical guarantees of the optimality of the designs.

5.3.5 Outcome

In interaction with the expert committee, we defined several instances of the optimization problem, which differed in their character sets and constraints. The objective weights were varied systematically to explore the effect of preferring certain objectives over others. In particular, some experts would favor high familiarity and intuitive layouts whereas others would put greater emphasis on performance and ergonomics. The optimized designs were assessed by the committee and prompted changes in the instances, producing new designs in an iterative process. The proposed layouts all were optimized over several days and were no more than 10% away from the globally optimal solution, with smaller instances reaching objective values within 1–5% of the global optimum.

By means of the evaluative functions described above, the layouts could be quantitatively compared to the Azerty layout and those created manually by the expert committee (before the introduction of our approach). They were superior in all cases, except with regard to the familiarity objective, for which Azerty naturally obtained a higher score. Figure 5.10 shows an example comparison, of the optimized design (top) with Azerty (bottom left) and with the manually created layout (bottom right). Note that the layouts could be compared only with respect to those keys in common between the two relevant assignments. This additionally biases the comparison in favor of the Azerty layout, since the optimized designs entail assignment of nearly three times as many characters.

Finally, the expert committee made manual adjustments to one of the proposed designs, where these consisted of one or more swaps or displacements within that proposed layout, in order to locally adjust tradeoffs between criteria. The evaluative functions of the optimization process

\textsuperscript{5} See http://www.gurobi.com/.
Figure 5.10. Comparison of an optimized proposed design (top) with (a) Azerty and (b) a manually created layout (bottom). A difference below 0 for an objective ("P" = performance, "I" = intuitiveness, "F" = familiarity, and "E" = ergonomics) indicates that the optimized design is better with regard to it. Note that comparison is possible only with respect to the characters in common between the two designs, and the visualization does not feature other characters.
were used to quantify the effect with regard to each of the four objectives and ensure that no detrimental changes had been made.

A public inquiry was carried out, with a proposal of the standard that included a layout developed by means of the optimization process as described above. Over a span of several weeks, the French public had the opportunity to comment on the proposed standard. Nearly 3,000 qualified comments were collected, the majority of which were positive. The constructive criticism was summarized by the committee and informed the definition of new task instances, including a smaller character set, updated similarity scores, and new constraints aimed at, for example, consistently placing completing characters on keys that are located next to each other (e.g., \{ `with` \} and « `with` »). At the time this thesis was submitted, the process of defining the final layout was still ongoing.

5.4 Summary

In this chapter, I have presented the optimization of three text input methods. They advance efforts toward text entry optimization by going beyond letter entry on physical and soft keyboards. The cases presented involve new form factors and interaction techniques, with application of the novel objective functions presented in Chapter 4. I optimized for input performance but also aspects of ergonomics and learnability, such as finger individuation and strain, motor learning, memorability, and intuitiveness. Table 5.3 summarizes the three cases presented, in terms of the optimized input device and method, the objectives, and the optimization techniques.

The goal with Publication I was to design an assignment of letters to the keys on the piano that enables fast and easy-to-learn text input by allowing pianists to exploit their existing motor skills in the music domain. While still looking at text input done with button presses, the work for this case considered a form factor very different from the grid of a standard keyboard. Optimization of the transfer of motor skills from another domain had never been considered in text entry research. The outcome allows high-performance input of over 80 WPM, comparable to that of expert typists using the Qwerty keyboard, after a much shorter training time.

The work reported on in Publication II allows computationally designing and evaluating chording gestures for text input in mid-air. I developed mathematical models to quantify the performance and ergonomics of these gestures. Using heuristic optimization methods, I designed an assignment
optimizing in terms of movement speed, finger individuation, gesture complexity, and memorability. The corresponding objectives greatly expand the space of optimizable text input methods and generally enable the computational design of mid-air gestures.

The third case is best described as one of optimizing the entry of special characters on a physical keyboard. This is a large-scale real-world application case requiring the assignment of over 100 characters to key slots. The input of special characters differs from entry of alphabetical letters in that the frequency distributions vary hugely, depending on the input task, and these characters are entered mostly in combination with normal letters. The optimization process developed served as a solid approach to designing the new French keyboard standard. Prior work has considered neither such large cases nor the input of special characters.

I formulated the full optimization model for all three cases as instances of the letter-assignment problem. This allowed me to draw on a large body of research, concerned with the approximation of good solutions to the NP-hard quadratic assignment problem. The methods I used to solve the problems presented range from heuristic methods using a constructive greedy approach to random search through to solving the actual integer program with a mathematical solver.

Comparison of what is presented in Table 5.3 with the related work as presented in Table 3.1, in Chapter 3, clearly shows that the cases dealt with here go beyond prior work in terms of the optimized input devices and movements, as well as the objective criteria. Solutions could not have been reached with models from prior work, and my contributions vastly enlarge the space of optimizable text input methods.
Table 5.3. An overview of the optimization cases presented in this chapter. The table presents the optimized input device and corresponding input actions, the mathematical formulation used to model the design task, and the objectives and the optimization method used to solve the problem. The design problems and objectives go beyond prior work as listed in Table 3.1.
6. Observations of Modern Typing Behavior and Implications for Optimization

Model-based optimization of text input methods has its roots in the observation that the Qwerty layout is not ideal for entering text on a typewriter. Over the last century, keyboard layout optimization has advanced greatly from the first manual attempts to rearrange the letters, to formulating the problem mathematically and using advanced optimization techniques to approximate better solutions to the NP-hard quadratic assignment problem. However, the evaluative knowledge used in assessment of the performance of a physical keyboard layout operated by multiple fingers has stayed largely the same since August Dvorak’s book *Typewriting Behavior* [30], published in 1936. In brief, a physical keyboard is optimized such that the following conditions [110] hold:

1. The load is the same for each hand.
2. The most frequent characters are typed in the middle (fastest) row.
3. The frequency of bigrams typed by alternating hands is maximized.
4. The frequency of bigrams in which the letters are typed by the same finger is minimized.

Even fairly recent attempts to optimize physical keyboard layouts are based on models and phenomena established many decades ago [9, 26, 31, 44, 66, 163]. The phenomena addressed were identified from empirical observation of professionally trained touch typists mostly operating typewriters [30, 39, 57, 72, 131, 134]. A few attempts have been made to collect performance data with modern keyboards, but the resulting models still focus solely on touch typing as the input method [60]. There is a lack of scientific understanding of typing in the era of modern computer keyboards. In comparison to the typewriter era, the input device and typing behavior have changed dramatically with respect to the following elements:
1. *The physical form factor:* Modern keyboards are flatter, and the rows are at similar height. Pressing their keys does not require as much force and have a relatively high activation point.

2. *The typing tasks:* The keyboard is used for many more tasks than the entry of formal text, among them gaming, programming, and informal communication via social-networking or online-chat applications.

3. *The user group:* Typing long remained a task of professionally trained secretaries. Today, typing on a keyboard is one of the most prevalent activities in computer use, conducted for several hours a day by everyone from schoolchildren to elderly people, done at work and at home. However, many people do not undergo any formal training in the touch-typing system. Rather, typing styles emerge and manifest themselves as highly varied strategies employing anywhere from two to 10 fingers (see Publication III).

The goal with this chapter is to advance the understanding of modern computer users' typing behavior and performance. I present findings in new areas from two empirical studies. The first is an in-depth investigation of typing strategies employed by trained and untrained typists. Using motion capture technology and eye tracking, I analyzed the hand and finger movements, gaze deployment, and the impact of both on typing performance. The second study provides a broader understanding of modern typing performance, through analysis of a large dataset (with data collected on a large scale from more than 168,000 computer users). This allows robust statistical analysis of keystroking patterns that reveals additional aspects of typing behavior that are predictive of performance. The datasets obtained are unique in their size and level of detail, and both have been made available to support further research on this topic.

With the following discussion, I give a brief overview of the typing phenomena and models of touch typing presented in earlier work. I then present the two datasets and summarize the most important findings on modern typing behavior. Finally, I discuss their implications for the optimization of physical keyboard layouts and for text input in general.

### 6.1 Phenomena and Models of Touch Typing

Typing is a complex skill that involves perceptual, cognitive, and motor processes. Similarly to piano playing, it requires many hundreds of hours
of practice before mastery is achieved. Still, its practical relevance has led to many people being skilled in typing. This has ensured that typing has held interest as a topic of psychological study for over a century [22]. For a long time, expertise in typewriting implied being a professionally trained touch typist. Even today, many researchers distinguish between "standard" and "non-standard" typing [87], notwithstanding the large percentage of computer users who have never taken a typing course (see publications III and IV). Accordingly, our understanding of typing is derived largely from empirical observation of touch typists. The phenomena clearly identified in this work are summarized below.

6.1.1 Empirical observations

Table 6.1 (adapted from material presented in Publication III) summarizes the main phenomena related to the performance and motor behavior of touch typists. These are well-established findings that have been replicated by several studies, as reviewed, for example, by Salthouse [134]. Although the origins of the findings lie in work conducted during the 1920s–1980s (in the typewriter era), research on modern text input still builds on them [9, 31, 61, 163, 173]. Note that the table omits phenomena related to cognitive and perceptual processes, such as the units of text, movement planning, and eye movement patterns – there is no principled reason to expect that those processes would be different in modern typing. I focus instead on the observable aspects of motor control and performance.

The performance found for typists usually was around 60–80 WPM [133, 52], with average inter-key intervals of less than 160 ms [57, 145, 144]. Several researchers observed that typing is not a sequential process; fingers move in parallel [76, 110, 131]. Accordingly, many studies have shown that bigrams typed with different hands are 30–60 ms faster than those using two fingers of the same hand and about 80 ms faster than use of the same finger for both letters [30, 53, 72, 131, 134, 156]. Also, bigrams that occur more frequently in language were typed more quickly than infrequent ones [30, 53, 133, 156]. I would expect typing on modern keyboards to influence the hand alternation benefit in that the computer keyboard's flatter keys with less travel distance make it easier to prepare keystrokes by the same hand as well as those typed by alternating hands.

When users have been asked to type as quickly and accurately as possible while not correcting mistakes, they showed low error rates, in the < 1–3% range [52, 133, 145], although sometimes participants were excluded if
Observations of Modern Typing Behavior and Implications for Optimization

<table>
<thead>
<tr>
<th>Phenomenon or measurement</th>
<th>$M$</th>
<th>$SD$</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Background factors:</strong></td>
<td>Most studies were conducted with professionally trained typists.</td>
<td>[30, 52, 133, 145]</td>
<td></td>
</tr>
<tr>
<td>Participants touch typing (%)</td>
<td>100</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Weekly amount of typing (h)</td>
<td>11</td>
<td>19</td>
<td>[133]</td>
</tr>
<tr>
<td><strong>Performance:</strong></td>
<td>The average inter-key interval (IKI) is a small fraction of typical choice reaction times (e.g., 560 ms). The typing rate is lower for random letter sequences.</td>
<td>[133]</td>
<td></td>
</tr>
<tr>
<td>Words per minute</td>
<td>75</td>
<td>9.7</td>
<td>[52]</td>
</tr>
<tr>
<td>IKI for easy prose (ms)</td>
<td>140</td>
<td>–</td>
<td>[145]</td>
</tr>
<tr>
<td>IKI for random strings (ms)</td>
<td>326</td>
<td>–</td>
<td>[145]</td>
</tr>
<tr>
<td><strong>Alternation of hands/fingers:</strong></td>
<td>Letter pairs (bigrams) typed with fingers of different hands are 30–60 ms faster than those typed with one hand.</td>
<td>[134]</td>
<td></td>
</tr>
<tr>
<td>Hand alternation (%)</td>
<td>48</td>
<td>–</td>
<td>[30]</td>
</tr>
<tr>
<td>- IKI (ms)</td>
<td>155</td>
<td>43</td>
<td>[133]</td>
</tr>
<tr>
<td>Finger alternation (%)</td>
<td>34</td>
<td>–</td>
<td>[139]</td>
</tr>
<tr>
<td>- IKI (ms)</td>
<td>194</td>
<td>45</td>
<td>[133]</td>
</tr>
<tr>
<td>The same finger (%)</td>
<td>4.6</td>
<td>–</td>
<td>[30]</td>
</tr>
<tr>
<td>- IKI (ms)</td>
<td>223</td>
<td>41</td>
<td>[133]</td>
</tr>
<tr>
<td>Letter repetition (%)</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>- IKI (ms)</td>
<td>176</td>
<td>26</td>
<td>[133]</td>
</tr>
<tr>
<td><strong>Bigram frequency:</strong></td>
<td>Bigrams occurring more frequently in language are typed more swiftly than infrequent ones.</td>
<td>[30, 53, 133, 156]</td>
<td></td>
</tr>
<tr>
<td><strong>Errors:</strong></td>
<td>40–70% of errors are detected without reference to the transcription output. Most errors are insertion errors.</td>
<td>[134]</td>
<td></td>
</tr>
<tr>
<td>Error rate (%)</td>
<td>0.95</td>
<td>0.5</td>
<td>[52]</td>
</tr>
<tr>
<td>- Substitution (%)</td>
<td>22.7</td>
<td>22.3</td>
<td>[52]</td>
</tr>
<tr>
<td>- Insertion (%)</td>
<td>42.4</td>
<td>18.7</td>
<td>[52]</td>
</tr>
<tr>
<td>- Omission (%)</td>
<td>13.8</td>
<td>5.3</td>
<td>[52]</td>
</tr>
<tr>
<td>- Transposition (%)</td>
<td>7.2</td>
<td>3.5</td>
<td>[52]</td>
</tr>
</tbody>
</table>

Table 6.1. Selected phenomena of hand movements and performance in touch typing. Example results are given from research with professional touch typists, and other findings may differ from these. Adapted and extended from material in Publication III.
their error rate was beyond a certain threshold – for example, 2% [145]. Salthouse noted that only 40–70% of errors could be detected without reference to the typed text [134, 166]. Most errors were insertion errors (typing an extra letter), with an adjacent key getting pressed in addition, which resulted in extremely small IKIs [52, 134]. Substitution errors (typing the wrong letter) often involved horizontally adjacent keys or, less frequently, vertically adjacent ones [52]. Omission errors (leaving a character out) were frequently observed for keys that are hard to reach (e.g., m and n) [29, 134] and were accompanied by particularly long IKIs [143]. The least frequent error type was transposition errors, which most of the time involved bigrams typed by different hands [52, 134, 143].

Salthouse speculated that errors are partially due to faulty executions of keystrokes, such as inaccurate movements, incorrect placement of the fingers (substitution errors), simultaneous depression of two adjacent keys (insertion), inadequate force on a key (omission), or preparation of keystrokes out of sequence (transposition) [134]. The different form factor of modern keyboards and variations in input strategies employed by users are likely to affect the presence and frequency of such error types.

### 6.1.2 Theories and models

Many researchers have developed models and theories to explain or predict the cognitive, perceptual, and motor processes during touch typing. With known text to be typed, quantitative models are applied to predict the inter-key interval or movement dynamics of the hands and fingers on the basis of the phenomena described above. They range from simulation tools [61, 110, 173] through concise mathematical models [57] to simple lookup tables of key transition times [60, 72, 120]. Qualitative theories have been developed in attempts to explain the cognitive processes that organize and schedule the motor movements [86, 134, 156]. I will now further consider some of the quantitative models, the ones most suitable for keyboard layout optimization.

The model of Rumelhart and Norman [110, 131] assumes that hand and finger movements are planned and executed in parallel through so-called motor schemas. These are interactive, parameterized programs with an activation value whose rising or falling causes or inhibits execution of the program. These schemas are interconnected, and the execution of one program may lead to increase or decrease in the activation value of another. For a letter sequence to be typed, the simulation model organizes
the activations and the interaction among these motor programs and ultimately outputs a sequence of finger movements. The simulation model is able to reproduce several phenomena, such as various observed patterns of errors and the benefit of hand alternation.

The Typist model proposed by John [61] has been created within the cognitive modeling framework of the Model Human Processor [19]. With its predicted movement times, the model reproduces 20 phenomena of transcription typing as reviewed by Salthouse [134]. Similarly, the more recent model by Wu and Liu [173] applies a queuing-network-based theory of cognition within the framework of the Model Human Processor to predict intervals between key presses, reproducing 32 phenomena of transcription typing, with varying levels of success.

Hiraga [57] proposed a mathematical model that predicts the interval between key presses as a linear function of various characteristics of the fingers chosen to press the keys, among them hand alternation, finger, and row transition, as well as the frequency of the letter pair. To the best of my knowledge, this is the only preexisting functional equation model. Hiraga used linear regression to fit the parameters to the typing data of a single professional typist.

The different form factor, the lack of deliberate training, and the various input tasks are likely to influence how we move our hands and fingers for use of modern keyboards in comparison to the typewriter. However, there is no model that captures the movements of non-touch typists, who may use any number of fingers from two to 10. The datasets and findings presented in publications III and IV are important enablers for revisiting modeling of the typing behavior of modern computer users.

6.1.3 Studies of non-touch typing

Even though typing is one of the most basic activities in interaction with computers, only a few studies have investigated the movement behavior associated with modern keyboards. In contrast to the research discussed above, some have not only investigated skilled typists but assessed the performance of regular computer users. In a study of 60 undergraduate students, Keith and Ericsson [68] found that the average typing speed was only 33 net WPM (“net WPM” is computed by subtracting five characters for each mistake). They found that performance was correlated with the amount of deliberate typing practice reported by the students, where those participants who had taken a typing course were fastest. Grabowski [49]
assessed the performance of 32 female university students not trained in typing. Unsurprisingly, he observed that they were not as efficient as the professionally trained typists studied by Salthouse. They made more typing mistakes and required more visual guidance. Nevertheless, he found that some IKIs were as low as 170 ms, irrespective of the lack of deliberate training. Rieger and Bart [128] recently conducted a survey involving 298 touch and non-touch typists to assess their use of various sources of information (attention to the text, screen, keyboard, kinesthetic feedback, etc.) during transcription typing, free typing, and error correction. They found that those who identified themselves as fast typists relied less on visual information about the typing process (e.g., the location of their fingers on the keyboard). This was independent of the number of fingers used. On the other hand, differences in typing behavior (for touch and non-touch alike) were found to be related to differences in the balance among the sources of information. However, the results were all based on self-reporting. Logan et al. [87] investigated how typists trade off between Fitts’ Law and Hick’s Law to find an optimal mapping from fingers to keys. Their argument is that touch typists are more efficient because they use more fingers (minimizing travel distance), in a more consistent way (minimizing choice reaction time).

These studies highlight the great variation in typing skill and behavior among everyday computer users. However, we still lack a principled understanding of the input movements employed by modern typists and their impact on performance, which could inform the design of keyboards. Hence, for publications III and IV, I studied the movement behavior of trained and untrained typists in more detail.

### 6.2 The HOW-WE-TYPE and 136M-KEYSTROKES Datasets

To assess the typing behavior and performance of modern computer users, I collected two datasets: (1) the HOW-WE-TYPE dataset, presented in Publication III, provides motion capture data of the hand and finger movements of 30 participants and thereby allows more detailed analysis of the movement behavior; (2) the 136M-KEYSTROKES dataset, presented in Publication IV, contains key press data (inter-key intervals) from more than 168,000 users who participated in an online typing test, which affords robust statistical analysis of modern typing performance. Both datasets have been made publicly available. In this section, I briefly describe the collection of said
data, while the remainder of the chapter summarizes the findings and implications for keyboard optimization.

### 6.2.1 The HOW-WE-TYPE dataset

The HOW-WE-TYPE dataset consists of typing data from 30 participants (17 of them female) transcribing 50 easy and memorable sentences from the Enron email dataset [160]. Participants were sampled so as to cover a wide range of typing speeds, 34–79 WPM. Their ages were between 20 and 55 years (mean: 31). To enable observation of the typing behavior they were using in their everyday computer interaction, participants could choose to type sentences in either their mother tongue or the language they most commonly used for typing, which was either English or Finnish. The Finnish-language sentences were the same as the English ones in content and were translated by a native speaker of Finnish instructed to use simple, everyday language. The sentences were shown one at a time, in random order. Participants were instructed to read through a sentence and, after that, type it as they would normally, without paying special attention to speed or accuracy. They were told to correct errors if they noticed them within a few characters but not to go back several words to correct mistakes.

In addition to the key presses, I recorded the movements of the hands and fingers precisely by using a motion capture system. It consisted of eight high-speed cameras recording the position of the keyboard and 52 reflective markers placed on anatomical landmarks of the two hands. This allowed me to identify the finger that pressed a given key and gave me detailed information about the movement behavior of all fingers at each key press and between presses. Additionally, I used eye-tracking glasses to record the participants’ gaze and capture the movement of attention between keyboard and display. More details are given in Publication III.

In summary, the HOW-WE-TYPE dataset consists of data from 30 users, including motion capture data (x-y-z coordinates of 52 markers recorded at 240 fps), key press data, eye tracking videos (annotated with gaze points), and reference videos of hand movements recorded at 60 fps.
6.2.2 The 136M-Keystrokes dataset

The 136M-Keystrokes dataset was collected in an online study. In collaboration with a commercial organization\(^1\) that offers touch-typing courses and typing tests, we launched a “scientific typing test,” which was offered on the company’s website. The test was designed in line with common practices in text entry research and used analysis of standard measurements of text input performance \([92, 170]\), with those values being presented to users at the end of the experiment. Participants were asked to type 15 sentences, one after another. Instructions stated to read through a sentence and then transcribe it as quickly and as accurately as possible. These stimuli were randomly sampled from a set of 1,525 sentences composed of phrases from the Enron Mobile Email corpus \([160]\) and the English Gigaword Newswire corpus \([50]\). When all sentences were transcribed, participants had to fill in a demographic questionnaire asking about their age, gender, country of residence and native language, keyboard type and layout, typing experience, and typical number of fingers for typing. Only then were they shown the final results (speed in WPM, error rate, the slowest and fastest sentence, and the sentence with the most errors).

Data were collected for a duration of three months, during which more than 193,000 people, from 218 countries, completed the test and questionnaire. Participants were self-selected from the company’s user base. When arriving at the webpage, they could choose between the standard typing test, in which fixed text is typed for one minute, and our experiment, advertised as a “scientific” alternative. Participants spanned a wide age range, 8–60 years, and subjects reported typing anywhere from less than one to more than 20 hours per day. However, most were English-speaking teenagers and young adults, with a higher percentage of females. Over 70% reported having taken a typing course – a large number in comparison to the 43% in the How-We-Type dataset. After exclusion of incomplete, inaccurate, and corrupted data, the final dataset included observations from 168,960 participants. More details are given in Publication IV.

The study was designed in accordance with guidelines from several online study platforms \([123]\). Typing 15 sentences was a short enough task to not become tedious or exhausting, and participants were shown their progress throughout the study. Users were motivated to finish the tasks in order to receive statistics on their typing performance in comparison to others’.

\(^1\) Typing Master Finland Oy.
Observations of Modern Typing Behavior and Implications for Optimization

Such large-scale datasets from general typing are rare in the literature. Even though online studies do not permit as rigorous control, the large sample increases the statistical power and yields better estimates and shapes of distributions [124]. Although self-selecting, participants in online studies have been shown to constitute more variety-rich samples, and the results produced are similar to those of laboratory experiments [40, 123]. The 136M-KEYSTROKES dataset consists of data from over 168,000 users, including the keystroke data (character, press and release times, and IKI), as well as demographic data of participants (e.g., age, gender, country, native language, typing experience, and number of fingers used for typing).

6.3 Findings

In this section, I summarize important findings from analyzing the datasets presented above. More details on the analysis and results are given in the respective publications.

6.3.1 Performance measurements

The large size of the 136M-KEYSTROKES dataset allows provision of the first reliable estimates and distributions for basic performance measurements specific to modern typing.

Typing speed

On average, participants typed at 51.56 WPM ($SD = 20.2$), with some reaching speeds above 120 WPM. The distribution of speed across participants is shown in Figure 6.1a. As is common in metrics of human performance, it shows a slight positive skewness. The performance is slower than that observed for the (smaller) HOW-WE-TYPE dataset ($M = 58.4$, $SD = 12.9$) and shows more variation. In comparison to the participants in the work by Salthouse and others, the sample in these two studies showed greater diversity, resulting in slower performance rates than those observed with expert typists alone (see Table 6.1).

The corresponding average time found between two key presses is 238.7 ms ($SD = 111.6$). There is a lower bound of about 60 ms, similar to that found in studies of expert typists [57]. However, there is a very large range of variation, with the average IKI of slow typists being about 480 ms and sometimes even exceeding 900 ms. Accordingly, the distribution shown in Figure 6.1b is skewed to the right with a large kurtosis. In contrast, the
Figurine 6.1. At top is the distribution of typing performance, and on the bottom is a density estimate for the inter-key interval and key press duration, based on data of over 168,000 users from the 136M-KEYSTROKES dataset as presented in Publication IV.

duration of key presses is more narrowly distributed, with an average duration of 116 ms and a standard deviation of only 23.88 ms. Press duration showed only a small correlation with performance ($r = -0.29$), indicating that most gains in speed are achieved elsewhere.

**Typing errors**

The uncorrected error rates in both datasets are generally low, around 0–2.7%. However, the average number of error corrections (2.29 Backspace or Delete key presses per sentence) varies greatly across participants in the 136M-KEYSTROKES dataset ($M = 6.3\%$, $SD = 4.5\%$). In a contrast to prior studies [52, 65, 133], substitution errors were more frequent than omission or insertion errors, and showed a relatively high correlation with performance ($r = -0.45$). My hypothesis is that slower typists are less
Observations of Modern Typing Behavior and Implications for Optimization

Figure 6.2. Average inter-key intervals of users typing letters in the bottom, middle, and upper row of the keyboard. Significant differences were found for neither touch nor non-touch typists.

consistent in their typing behavior and have a poor mental representation of the fingers’ position on the keyboard, resulting in more substitution errors.

Differences between rows

For the HOW-WE-TYPE dataset, there was no significant difference in participants’ average IKI for bigrams confined to the upper, middle, or bottom row of the keyboard. As shown in Figure 6.2, typing letters in the upper row was found to be slightly faster, especially for non-touch typists. However, a Kruskal–Wallis test found no significant differences between the average IKIs of user typing letters in each row, for either the touch ($H(2) = 1.03, p = 0.6$) or the non-touch typists ($H(2) = 0.93, p = 0.63$). Note that this analysis was not reported upon in Publication III; the first reporting is in this thesis.

The finding described above stands in contrast to prior work from observation of expert touch typists, such as that by Dvorak et al. [30], who stated that the middle row was the fastest to use since this is the home row for the fingers to rest on. Accordingly, the DSK has the most frequent letters placed in the middle row. However, the analysis above suggests that people typing on modern keyboards tend not to use the middle row as a starting and ending point for finger movements. Hence, there is no advantage in placing frequent letters there.

6.3.2 Hand and finger usage

The motion capture data collected for Publication III allowed detailed analysis of typists’ hand and finger usage. This was not possible in prior
research, which recorded only the keystrokes, without information about which finger was used to press a given key. While this is predefined for the touch-typing system, finger-to-key mappings have never been studied for non-touch typists. In my study, the results were used for classifying frequent bigrams as left- or right-hand or as hand-alternation bigrams if more than 90% of occurrences were typed accordingly Publication IV. In addition, this allowed me to assess differences between hands on the basis of the larger (136M-KEYSTROKES) dataset.

**Number of fingers**

On average, in the 136M-KEYSTROKES dataset the participants reported using about seven fingers, where those who had taken a typing course reported using more fingers than those who had not had formal training (8 and 6.5, respectively). These numbers are similar to those determined through motion capture analysis from the HOW-WE-TYPE dataset (8.5 for touch and 6.2 for non-touch typists), which shows in addition that participants used more fingers of the left than of the right hand.

In the work for Publication III, I found no correlation between input performance and the number of fingers used for typing. In particular, non-touch typists were, on average, just as fast as touch typists. However, the typing speed of participants in the 136M-KEYSTROKES dataset showed a clear correlation with the number of fingers used for typing ($r = 0.34$). This may be explained by the differences in participants arising from the sampling choices. I conclude that, although it is generally beneficial to use more fingers, employing six or seven can be enough to reach typing speeds similar to those obtained by expert touch typists.

**Global hand movement and differences between the hands**

Although the HOW-WE-TYPE dataset revealed use of fewer fingers of the right hand than of the left, the participants operated more keys with the right. They also globally moved the right hand more than the left one, which was found to be slower than keeping the hand static and moving fingers individually. In contrast to these observations, prior work found the right hand to move more quickly [30], which was observed, to a small extent, also with the 136M-KEYSTROKES dataset (average difference: approx. 7–15 ms).

The extent of global hand movement was found to correlate strongly with the average IKI ($r = 0.6$ for the left hand, $r = 0.67$ for the right). Fast typists could keep the hand at a static “home” position while moving the
fingers individually toward the keys, whereas slower typists moved all fingers toward the target, thereby producing a slower performance. This was more pronounced for the right hand, with the left hand generally kept more static.

**Hand and finger alternation**

On average, bigrams typed with both hands were about 5–20 ms faster than those typed by only one hand. Interestingly, this benefit was smaller for faster typists (see Publication IV), and analyses in Publication III show a significant hand alternation benefit for only non-touch typists. The average benefit (of 17 ms) gained by touch typists was not significant. This is shown in Figure 6.3, which compares average IKI between touch and non-touch typists in the HOW-WE-TYPE dataset for bigrams typed with hand alternation, finger alternation, and the same finger.

This finding is surprising and contrasting to results from earlier research, with expert typists, which repeatedly found a clear hand alternation benefit of 30–60 ms [30, 53, 72, 131, 134, 156]. One likely factor is that the physical properties of modern keyboards better allows fingers of the same hand to prepare upcoming key presses, not just fingers of alternate hands. Accordingly, only bigrams typed by the same finger are typed considerably more slowly, by an average of 33–61 ms in the HOW-WE-TYPE dataset for touch and non-touch typists alike.

**Finger-to-key mappings**

The motion capture data allowed me to determine the finger-to-key mappings of non-touch typists – that is, which finger was used to press each
Observations of Modern Typing Behavior and Implications for Optimization

key. I used hierarchical clustering to identify similarities between these mappings across participants. The resulting clusters represent *typing strategies* employed by several users. I found four strategies for the left and six for the right hand, as depicted in figures 6.4 and 6.5. They range from using mainly one or two fingers per hand to using 4–5 fingers in line with the touch-typing system and variations thereof. Strategies varied more for the right hand, whereas half of the participants used a touch-typing strategy for the left hand. Interestingly, only 13 reported having taken a touch-typing course: some users had developed a similar strategy independently. That said, only seven of these participants fell within the touch-typing cluster for the right hand, with the rest deviating significantly from this strategy. Notably, participants typing at over 70 WPM could be found in almost all clusters, indicating that the number of fingers or the finger-to-key mapping is not decisive for high-performance input. More details on the clustering method are given in Publication III, along with analysis of the strategies.

*Key press consistency*

I found that, in general, faster typists were more consistent in the finger used to press a certain key than slower typists were. The entropy of the finger-to-key mapping (see Publication III) strongly correlated with the average IKI ($r = 0.45$). While this was found also for trained typists, whose finger usage sometimes deviated from the predefined touch-typing system, non-touch typists showed especially large variation in their finger-to-key mapping. That is, they used different fingers to press a certain key in different contexts.

Logan *et al.* [87] found a similar correlation between performance and consistent use of each finger. They argue that fast typists learn through practice to use the same finger for a given key and thereby reduce the reaction time to choose between the fingers (under Hick’s Law). This results in better performance.

### 6.3.3 Parallelization of input movements

I found evidence from both datasets that faster participants showed a more parallel (rather than serial) approach to input movements between fingers than slower ones do. For Publication III, I assessed the distance of a finger from its target at the time of the preceding key press and found a significant correlation with participants’ average IKI ($r = 0.5$). At the time
Figure 6.4. Strategies in typing with the left hand, as revealed from clustering of participants’ finger-to-key mappings. The figure presents example users from each cluster who achieved the highest performance rates. The bar plots indicate the relative number of keys pressed by each finger and the overall percentage typed by the corresponding hand. The circles on the keys indicate the relative frequency of using the corresponding finger to press the relevant key.
Figure 6.5. Typing strategies used with the right hand, as indicated by clustering of participants' finger-to-key mappings.
of a key press, the next finger is already moving toward the next target. This is handled more efficiently by faster typists.

Examining the 136M-KEYSTROKES dataset, I went one step further and analyzed when the next key was depressed. I found a large percentage of bigrams to be typed with rollover behavior. This concept refers to the technique of typing consecutive keys in an overlapping fashion: the second key is already pressed down before the first is released. This phenomenon is well known among keyboard manufacturers and e-sport practitioners, since it allows higher-performance input. However, it has gone unrecognized in text entry research. For Publication IV, I calculated the rollover ratio of participants as the number of keystrokes typed by rollover divided by the total number of keystrokes. On average, participants typed 25% of bigrams via rollover, and the rollover ratio was found to be highly correlated with performance ($r = 0.73$). Fast typists used rollover for 40–70% of key presses, which yields a large performance benefit. When rollover is used, keystrokes overlap by, on average, 30 ms and as much as 100 ms, which shows the potential benefit of this technique in terms of input performance.

Researchers studying expert typists already observed that text input is not a serial process – fingers move in parallel toward upcoming key targets, an approach that allows better input performance [110, 120, 131]. The motion capture data presented in Publication III allowed me to assess the extent of these movements explicitly and thereby quantitatively confirm these observations. It should be noted that carrying this behavior as far as rollover is a new phenomenon, arising in modern typing and impossible with traditional typewriters (it could jam the keys). Even today, most keyboards support only a small number of keys being pressed at the same time and not every user interface allows this technique.

### 6.3.4 Visual attention and gaze shifts

The analysis of the gaze data collected as part of the HOW-WE-TYPE dataset showed that, on average, faster typists spend less time looking at their fingers (visual attention is needed) and shift their gaze less between the screen and the keyboard. Another significant difference was seen too, between touch and non-touch typists. The latter were found to spend twice as much time looking at the keyboard (20% as opposed to 41%). On the other hand, the correlation with performance was stronger for touch typists than for non-touch typists ($r = 0.81$ and $r = 0.32$, respectively, for the number of gaze shifts and $r = 0.69$ vs. $r = 0.53$, correspondingly,
Observations of Modern Typing Behavior and Implications for Optimization

for visual attention). This demonstrates that the non-touch typists had adapted to their reliance on visual feedback and were able to maintain high input rates despite frequent gaze shifts.

These findings are in line with the self-reports from typists recently collected by Rieger and Bart [128], who found that non-touch typists relied more on visual attention paid to the fingers and keyboard.

6.4 Summary

The findings presented in this chapter provide novel insights into the typing behavior of modern computer users. While many cognitive and perceptual processes could be expected to be similar to those observed in prior work, I found several differences in movement behavior and in performance when comparing my results with commonly cited literature on expert typing, on which keyboard layout optimization is based. Moreover, by using motion capture data, I was able to uncover commonalities in the typing strategies employed by non-touch typists, showing that a wide spectrum of strategies can be found between touch typing and “hunt-and-peck”. These findings have important implications for text entry research and the optimization of text input methods (keyboards especially), as well as for typist-training programs and designers in general.

6.4.1 Implications for keyboard optimization

Even recent work in the field of text entry optimization has relied on heuristics that are based solely on the touch-typing system. These must be updated in accordance with the findings presented in this work. Let us revisit the traditional tenets presented at the beginning of this chapter:

1. Equalize the load of the hands: Modern typists generally use the left hand more skillfully than the right. They employ more fingers of the left and keep that hand more static than the right one; that is, they move the left’s fingers individually (rather than the whole hand) toward the keys, which was found to correlate with strong performance. However, in modern typing strategies, fewer keys are operated by the left hand and the right one covers a larger area of the keyboard, including movements toward the Backspace and the Enter key. In addition, I found that inputting bigrams typed by only the left hand was slightly slower than typing those produced by only
the right hand (by approx. 7–15 ms). Hence, the principle can be revised as follows:

*Position the most frequent letters for the left hand while avoiding the need for frequent bigrams to be typed by the left hand only.*

2. **Place frequent characters in the (fastest) middle row:** I could not find any significant differences between letters on the basis of whether they are found on the bottom, middle, or top row of the keyboard. Inspection of reference videos shows that modern typing strategies do not make use of the middle row as a home row where fingers rest between key presses. However, more research is necessary for investigating whether an alternative home position exists. In any case, this principle should be eliminated as a guideline:

*Place frequent characters in the middle row.*

3. **Maximize the frequency of bigrams typed by alternating hands:** The hand alternation benefit frequently found in prior work is not as distinct in typing on modern keyboards. Only small benefits (of 5–20 ms) were found, and bigrams typed by different fingers of the same hand benefit similarly. I conclude that it is generally beneficial to type bigrams with different fingers, rather than the same one, but it is less important that those fingers belong to different hands. Using different fingers allows parallelizing movements toward keys and enables the use of rollover. However, more research is necessary, for better understanding the biomechanical and physiological mechanisms that enable and facilitate the use of rollover. Informed by this knowledge, we could optimize a keyboard layout to allow maximal use of this technique. Until then, the principle should be reformulated thus:

*Maximize the frequency of bigrams typed by two fingers.*

Note that the various typing strategies presented in figures 6.4 and 6.5 yield different key combinations typed by the same finger. Therefore, a keyboard layout optimized for finger alternation also needs to represent the best tradeoff between relevant strategies. The HOW-WE-TYPE dataset can be used as a starting point, but more research is necessary, for thorough investigation of the various typing strategies that exist, along with their frequency.
4. Minimize the frequency of bigrams typed by the same finger: The principle of minimizing typing of bigrams with a single finger is consistent with the findings described above, wherein bigrams typed by the same finger are significantly slower to produce than those typed with finger or hand alternation. Hence, it can be kept as-is. However, note that this is redundant, since it is already covered by principle 3, above.

These new heuristics form a useful starting point for optimizing modern keyboard layouts but offer only an approximation and do not cover all aspects of typing. For finding the optimal physical keyboard layout, a quantitative model of general multi-finger typing would be needed that is able to predict the time between two key presses from the typing strategy of the user. The findings presented in publications III and IV are an important step toward such a model. The datasets collected have been published to encourage further research into this topic fundamental to human–computer interaction.

6.4.2 High-level implications for text entry

The new insights into the motor behavior of modern typists have further implications for text entry in general. For over a century, training in typing has been based mostly on teaching the touch-typing system. The findings on modern typing behavior show that factors additional to the number of fingers affect typing performance. Rather than ignore individual-specific ways of typing, training materials could explicitly teach rollover, reduce visual attention, and encourage less global hand movement. They could be personalized to the deficits of each typist and provide practice aimed at avoiding specific mistakes or at better utilizing hand and finger alternation. This would allow users to boost their typing speed without enforcing a new finger-to-key mapping (the latter often leads to an initial performance drop and is discouraging for users).

Designers and developers creating user interfaces that involve text input should be sensitive to the differences in how people type. This means that sensing pipelines, text input methods, and intelligent assistant methods (e.g., autocorrection and word completion) should be designed to optimally support various typing styles and their consequences, these in particular:

- Differences in the type of errors and in their quantity.
- The usage of rollover and parallel movements by fast typists.
• Reliance on visual attention to the keyboard among many self-taught typists.

• Slower typists showing less consistency in finger usage and more global hand movement.

For example, overlapping key presses (and input actions in general) should be possible with any new text input method, including interaction on multitouch surfaces, in mid-air, etc. Autocorrection techniques such as the ones employed by word processors, mobile keyboards, and search engines could take into account a user’s typing style and thereby better identify typing mistakes. Methods for post-editing could be designed for slower typists, who rely more on visual attention to the keyboard while inputting text and hence have less availability of cognitive resources to identify and correct typing mistakes. Machine learning techniques, such as the clustering method used for Publication IV, could aid in automatically identifying various typing styles, for purposes of adapting a user interface or text entry technique accordingly and also enhancing practical training in typing by making personalized exercises available.
7. Discussion and Conclusions

The design of text input methods is a highly complex task at the core of which is always the question of how to best assign symbols (characters, words, etc.) to the input actions performed by the user. In its simplest form, there are 26 letters of the alphabet that must be assigned to 26 input actions. This alone results in over $10^{26}$ possible designs. Often, there is no intuitive or natural assignment (e.g., for chording gestures). Instead, the criteria for the goodness of a design are more implicitly related to the actual assignment and often entail considering the speed or accuracy achievable with the input method. Empirically assessing these for such a large number of designs is not practical for real-world cases, so traditional design approaches are inapplicable.

In this thesis, I have focused on model-based interface optimization as a computational approach to the design of text input methods. This entailed three challenges:

1. Mathematical formulation of the design problem.
2. Collecting evaluative knowledge and capturing it in quantifiable objectives.
3. Using appropriate optimization techniques to systematically search the design space.

To address the first of these, I have relied on prior work [18, 120], which showed that designing a keyboard layout can be formulated as an instance of the (quadratic) assignment problem. In Chapter 2, I generalized this to the larger problem of assigning a symbol to any input action and showed that this can be modeled as a linear or quadratic letter-assignment problem. That work forms the foundation of the thesis project. The mathematical formulation reveals the internal structure of the problem, thereby allowing me to utilize existing optimization techniques to efficiently find good
The main contribution of this dissertation is with respect to the second challenge. Apart from Fitts’ Law, quantitative models for optimizing text input methods are not readily available. It is traditional for designers to rely on empirical observations, heuristics, and prior experience to assess the goodness of a design [129]. Automating this process requires quantitatively capturing psychological, behavioral, and/or cognitive aspects of the interaction in terms of an objective function. While much work has been devoted to quantifying the motor performance achievable with keyboards, less is known about their ergonomics or learnability, and studies of alternative text input methods are often absent altogether. In the course of my work, I have conducted extensive empirical studies to capture and understand core aspects of performance and usability of mid-air input and multi-finger typing on modern keyboards (see chapters 5 and 6). With Chapter 4, I showed how to mathematically formulate this knowledge in terms of objective functions that can be used to optimize diverse text input methods in line with criteria for performance, ergonomics, learnability, and input recognition. In Chapter 5, I demonstrated application of these objective functions to optimize three novel text input methods. These go beyond prior work by considering new interaction techniques, form factors, and objective criteria and thereby extending the space of optimizable text input methods.

### 7.1 Summary and Implications of the Findings

The main outcome of this thesis project is a set of objective functions for optimizing a wide range of text input methods operated by one or several fingers. These quantify various aspects of performance, ergonomics, learnability, and input recognition. They are based on experimental studies conducted in conjunction with the thesis, alongside models, theories, and empirical observations from prior work on motor control, visual search, human memory, and other factors. Depending on the data and parameters they are instantiated with, they can be applied to different input devices and movements.

I also included models used in prior work in the field of text entry optimization but formulated them in the same way as the other objective functions: as a letter-assignment problem for integer programming. This allows one to combine the functions easily and to draw on a large set of
heuristic and exact optimization methods to solve the problems efficiently. It also demonstrates that the letter-assignment problem is a suitable model to capture most elements that are important for finding the best assignment of symbols to input actions.

The main implication of my work is in these objectives greatly expanding the space of optimizable text input methods. State-of-the-art text entry optimization work has focused mainly on finding the best spatial organization of letters in a grid, considering only touch typing with physical keyboards, soft keyboards operated by one or two end-effectors, or ambiguous grouping keyboards. Although text input is a complex process involving perceptual, cognitive, and motor processes, optimization efforts have been largely limited to speed maximization, with only a few scholars considering multi-objective optimization that encompasses, in addition, a user’s familiarity with another layout or improving the input recognition through autocorrection and similar methods. Many aspects of ergonomics or learnability – among them strain, fatigue and anatomical limitations, memorability, motor learning, and intuitiveness – have never been considered before. Similarly, models of motor performance were limited to pointing and heuristic observations of touch typing.

The objective functions presented in this thesis widen the space of optimizable text input methods. Accordingly, I have been able to present three optimization cases never considered previously: novel optimization of an assignment for skill transfer, text input by mid-air chording gestures, and optimal input of a large set of special characters. For each case, I have applied the proposed objective functions in a multi-objective optimization model, formulated as an integer program. I gathered the corresponding input data to instantiate the respective parameters and solved the problem by using appropriate heuristic or exact methods. These applications clearly go beyond achievements of prior work in terms of the input devices and movements optimized, as well as the objective criteria, and these problems could not have been solved before the work presented here. Thereby, they exemplify well the potential for expanding the space of optimizable text input methods to novel text input methods and evaluation criteria.

My work contributes also to a better understanding of modern typing behavior. While prior work focused on optimizing keyboard layouts for touch typing, modern computer users employ a range of multi-finger strategies, emerging from differences in keyboarding tasks (online chat, gaming, use of keyboard shortcuts, etc.). There are significant differences from the
behavior of expert touch typists studied by researchers in the 1930s–1980s, with respect to typing errors, performance in the various rows, how the hands and fingers are used (in particular, the number of fingers and the hand alternation benefit achieved), parallelization of input actions (e.g., rollover), and visual attention. Nevertheless, even recent research on keyboard layout optimization has employed heuristics and models from the typewriter era. My extensive empirical studies of modern typing behavior allowed me to propose a new set of heuristics for optimizing typing performance with modern keyboards. However, they also indicate that more work is necessary for covering all aspects of multi-finger typing and hence enable designing the optimal keyboard layout.

Better understanding of modern typing behavior has implications beyond the realm of keyboard optimization. The findings presented here can, for example, inform the design of suitable typing-training methods. For over 100 years, typing courses have focused on teaching the touch-typing system. My work shows, however, that similar performance rates are achievable independently of the number of fingers and of typing strategy. Instead, other factors influence performance: visual attention, use of rollover, consistency in key presses, levels of global hand motion, and others. Training can address these aspects without any need to learn a new finger-to-key mapping such as that dictated by the touch-typing system, and exercises could be personalized to correct user-specific deficiencies.

The contributions made through this thesis have implications for UI design more broadly. The problem of manual design methods being insufficient to assess very large design spaces is not unique to the design of text input methods. It applies to many domains of HCI design, from development of graphical interfaces, such as simple menus or complex web layouts, to higher-level endeavors, such as determining which features to include in a software tool or how best to visualize data [113]. Many of these problems can be formulated as assignment problems [114], in which case similar objective functions to the ones presented here, might be applicable. For example, the objective functions developed for optimizing mid-air text input offer a systematic way to assess performance, ergonomics, and learnability for general input performed via chording gestures in mid-air. Rather than rely on elicitation studies or the like, designers could specify the characteristics for a design task and the presented objectives for automatically identifying optimal gesture sets. This is only one interesting application for making the design process more predictable and reliable.
7.2 Limitations of the Work

As with any research, this work has its limitations, which come with the choice of methods discussed at the beginning of the thesis.

The studies reported on in publications Publication I, II, and III were conducted in laboratory settings. While this may affect external validity, I went to considerable effort to minimize interference from external factors, since my goal was to collect precise measurements of performance and observe skilled behavior. For the same reason, my choice for all text-input-related studies was a transcription task with easy and memorable stimuli – this minimizes interference from creative and other cognitive processes involved in more realistic tasks, such as text generation.

I conducted two online studies wherein I strove to collect a large quantity of data to enable greater statistical power of the observations [124]. Although online studies have been shown to attract user groups that are more variety-rich than convenience samples for lab studies [40, 123], some self-selection bias may remain. For example, young females from the US accounted for a large percentage of the 136M-KEYSTROKES dataset. More than 70% reported having taken a typing course, in contrast to 43% in the HOW-WE-TYPE dataset. The dropout rate is much higher than that in lab studies but was similar to other web-based experiments’ [108, 124]. Notwithstanding this, the online study allowed us to assess a wide range of people, from eight to 60 years of age, from 200-plus countries, and who reported typing anywhere from less than one to more than 20 hours per day. The datasets have been published; therefore, anybody is free to retrieve suitable subsets of the data for addressing particular research questions.

On account of the demanding requirements imposed for participation in the second experiment (being a skilled pianist) and its long-term nature, the observations in Publication I are from single-subject studies. While prior studies in the literature have been conducted with a similarly small number of participants [144, 57], this approach still only allows point observations whose generalizability is unclear. However, it is justifiable when the goal is a proof of concept.

The objective functions presented in Chapter 4 are based on quantitative models that capture the users’ performance, anatomical constraints, cognitive ability, etc. The validity of these largely depends on the underlying data. The models in this thesis are based on data from my own experimental studies or on observations made in prior, peer-reviewed work. The
goal in using a model is to abstract from observations of specific users and generalize to a broader population. Mathematical models, such as the Fitts’ Law-based ones used in Publication II, describe the relationship between the observations and the independent variables in terms of mathematical equations. However, the form of the relationship or the factors it depends on are often unknown. In such cases, we can use, for example, lookup tables, as done in Section 5.3. In what is shown there, I explicitly store the average transition times between special-character keys and letter keys to use for optimization. A problem remains in that such approaches are prone to overfitting to the potentially noisy observations of particular users, rather than capturing a general relationship. To avoid this, I binned average times of users to 30 ms bins, smoothing out fluctuations; they were then normalized for users’ general typing speed; and, finally, they were averaged across users, with the exception of outliers. In cases wherein the exact relationship is unknown, we can use heuristics instead of numerical predictions. This was done, for example, to optimize the memorability and complexity of gesture sets as presented in Publication II. While heuristics are not subject to the problem of overfitting, they might instead overly simplify the empirically observed relationship.

The models and input data are the foundations of the optimization process, whose outcome is optimal only with respect to the given objective function. Poor input data, overfitting, and excessive generalization from empirical observations all could lead to optimal solutions that perform poorly in practice. Therefore, validating the models and testing the outcome is an important part of a design process using model-based optimization. The models that are used in the objective functions presented in Chapter 4 are based on data collected in peer-reviewed studies presented in the publications connected with this thesis. In publications I and II, I empirically evaluated the outcomes of the corresponding optimization cases. Publication I offers clear evidence that the objective criteria (skill transfer and input performance) were met by the resulting design. Performance in the work described in Publication II was limited by difficulties with the tracking technology, which made it impossible to observe expert performance. A limitation of this work can be found in the methods for the study: they did not allow evaluation of other objectives (motor learning and memorability). Further work is necessary to evaluate the success of the optimization approach with respect to these criteria. The outcomes from optimization in the special-character assignment case presented in
Chapter 5.3 were assessed by the expert committee, and the members’ feedback led to manual adjustments. Quantitative evaluation through empirical user studies will be part of my future work.

While I am able to show that the letter-assignment problem can generally be used to model the problem of assigning symbols to input actions with respect to different input methods and objective criteria, there are limitations to its modeling power. It relies on an objective function that summarizes the costs of individual symbol–action pairs (or of two pairs, in the quadratic formulation). If the cost of interaction depends on the assignment as a whole or on multiple symbol–action pairs, a different formulation is required (see Section 2.3 for examples). Such costs might be difficult to model mathematically and increase the computational complexity of solving the problem. Heuristic optimization methods do not rely on a mathematical formulation of the problem and might, accordingly, be easier to apply in such cases. However, they have other drawbacks, as discussed in Section 2.2.

7.3 Future Directions for Text Entry Optimization

The work presented in this thesis greatly advances the optimization of text input methods. I showed that the assignment problem can generally be used to model the problem of assigning symbols to input actions for different input methods and objective criteria. However, there are design cases in which the structure of the available cost functions renders the assignment problem inapplicable. For example, performance models of two-thumb text input [21, 91, 115] require simulating typing for the given text in order to assess movement performance. Models of gesture typing assess the input action for entering a full word [15, 121, 150]. Optimizing the assignment of thousands of words is too computationally complex for solving with the quadratic formulation of the letter-assignment problem. Such objectives require the development of alternative optimization models, ones that allow capturing higher-level structures of the cost function.

My work extends the space of optimizable text input methods by proposing novel objective functions that go beyond models used in prior work. These functions capture the performance of novel input methods, such as mid-air gestures; allow one to assess the strain, fatigue, and anatomical limitations associated with inputting text; and quantify such aspects of learnability as complexity, skill transfer, memorability, and intuitiveness,
which have never before been considered in the optimization of text input methods. However, this list of objectives is by no means exhaustive. The specific input method or user group at hand may necessitate other objectives if we are to design a text input method that performs well in practice. Therefore, text entry scholars might benefit from research in the disciplines of psychology, cognitive science, motor control studies, ergonomics, and other fields that capture relevant aspects of manual input.

While prior work has focused on the spatial organization of letters in a keyboard layout, the present work extends the definition of the design problem to more generally consider the assignment of symbols to input actions. This shifts the perspective on the problem from a device- to a user-centered one – rather than ask about the best way to arrange symbols in a grid, I set out to find the best organization of the input actions the user performs to enter text. However, the design cases considered in this thesis all involve an assumption that the input actions are uniquely defined and only the symbol assignment can be varied. This assumption is valid, for instance, in one-finger input via virtual keyboards or for mid-air chording gestures, but it does not apply to multi-finger typing on physical keyboards. Users can perform any of various input actions in order to press a certain key, as Publication III illustrates, and the findings presented in Chapter 6 indicate that the organization of movements has a substantial impact on the performance achievable. Reduced global hand movement, more parallelizing, and consistent finger use are essential foci of touch-typing training. Yet self-taught typists displayed the same skills while using fewer fingers, skills that simply emerged through computer use and did not require deliberate training. Such idiosyncratic strategies are similar in efficiency to touch typing but might be easier to learn. Future text entry research could explore the matter of finding an optimal input strategy that is easier to learn or faster than the touch-typing system. This is yet another instance of the assignment problem. For a large family of problems, including this one, the methods presented in the thesis can serve as a useful starting point.

The goodness of a text input method, and user interfaces in general is often multifaceted. Optimization can be a powerful tool to explore different tradeoffs with respect to objective criteria, as well as user groups, tasks, and interests of stakeholders. The optimization of the French keyboard presented in Section 5.3 illustrates the importance of this exploration in real-life design processes. As presented in Chapter 2, I used a weighted
sum approach to combine multiple objectives into one objective function, where the weights denote the importance of each objective. This function is intuitive to use, and it is particularly suitable for larger instances of the NP-hard quadratic assignment problem since it does not further increase the computational complexity. However, there are several drawbacks to such scalarization methods. The importance of each objective must be determined up front, which impedes the exploration of different tradeoffs. Also, they typically ignore the shape of the Pareto set, and small changes in the weights can have a large impact on the optimized design. However, given the complexity of the quadratic assignment problem, it is challenging to apply methods with *a posteriori* articulation of preferences. Further research is necessary to develop efficient tools for better exploring the solution spaces and the relation among the various objectives.

The optimization cases presented here and those considered by prior work were dealt with primarily through a one-shot optimization process: firstly, the design problem is fully defined, and then instances are generated and solved. However, the case of the French keyboard showed that, similarly to a traditional design process, real-world optimization cases are iterative processes. The input of stakeholders was used to refine the optimization model and the instances for optimization. After that, the objective functions were employed as an evaluation tool to assess the effect of manual changes and thus locally explore different tradeoffs. Recent work in user interface optimization has developed tools for interactive optimization, which supports the designer’s work to define the problem, exploring the design space, and design good solutions in a manner that allows the system to apply any of various levels of proactivity [8, 112, 157]. Systems of this sort would be useful for the optimization of text input methods; however, the complexity of the quadratic assignment problem does not allow real-time optimization and thus poses a serious challenge for future work on developing interactive tools for text entry optimization.

### 7.4 Conclusion

The ultimate goal for this thesis is to enable the design of better text input methods. At a time where textual communication is one of the main activities carried out with computing devices, providing an interface that allows text to be entered efficiently and effortlessly is key to an excellent user experience. A complex but fundamental question in the design of
Discussion and Conclusions

any text input method is how to best assign symbols to the input actions performed by the user. The number of possible designs is very large, and their evaluation is complex, often requiring the designer to trade off among numerous criteria. Therefore, traditional design methods cannot be expected to yield good results.

In this thesis, I have shown how we can utilize methods of computational optimization for automatically generating and evaluating text input methods and thus find the best one with regard to various criteria. In this, the letter-assignment problem serves as a mathematical foundation. I have shown that it is able to capture the essential aspects of designing text input methods of various types. Proceeding from my experimental studies in combination with earlier work’s empirical observations and theories, I developed objective functions for optimizing the letter-assignment problem, which quantify aspects of performance, ergonomics, learnability, and the recognition of users’ input. This allowed me to optimize novel text input methods for criteria that have never been considered before in text entry optimization work. In addition, I obtained empirical evidence that the previous understanding of typing on keyboards is insufficient. Based solely on observations of touch typists, the literature ignores the large variations in typing behavior that have emerged with modern computer use. My findings characterize modern typing behavior and have important implications for keyboard optimization and text entry in general.

In conclusion, this thesis advances the field of text entry research through the following contributions:

1. Showing that the letter-assignment problem can be used to model the general assignment of symbols to any input actions for a wider range of text entry methods, not limited to keyboard layouts.

2. Mathematically formulating objective functions for the letter-assignment problem to optimize a text input method for several aspects of performance, ergonomics, learnability, and input recognition.

3. Exemplifying the application of these objectives in the optimization of three novel text input methods, thereby proving the objectives to extend the space of optimizable text entry methods.

4. Advancing our understanding of modern typing and proposing new heuristics for optimizing keyboard layouts that take into account the great variations in typing behavior.
References


the 26th Annual ACM Symposium on User Interface Software and Technology (UIST ‘13). ACM, New York, NY, USA, 331–342. DOI:http://dx.doi.org/10.1145/2581988.2502024


163


14) ACM, New York, NY, USA, 621–626. DOI:http://dx.doi.org/10.1145/2642918.2647382


170


[142] Andrew Sears, Julie A. Jacko, Josey Chu, and Francisco Moro. 2001. The role of visual search in the design of effective soft keyboards. *Behaviour & Information Technology* 20, 3 (Jan. 2001), 159–166. DOI:http://dx.doi.org/10.1080/0144929010049790


175


Assignment Problems for Optimizing Text Input

Anna Maria Feit