ABSTRACT

MCINTEE, SHEA STIRLING. A Task Model of Free-Space Movement-Based Gestures. (Under the direction of Robert St. Amant.)

One of the most useful tools in human-computer interaction is the task model, a structured decomposition of user actions toward a goal. Task models can predict average human performance on user interfaces, even at the design stage, before implementation. Existing task models describe and predict behavior ranging from selecting an icon with a mouse to typing on a keyboard.

Novel forms of interaction, however, often lag behind traditional modes of input such as keyboard and mice in their incorporation into task models; one of these modes is gesturing. With the current development of hardware platforms for augmented and virtual reality applications, input modes such as speech and gesture may come into high demand, leading to the need for task models of performance. This dissertation addresses this challenge with a simple but flexible representation and task model of gesture performance, called Pantomime.

Pantomime breaks down a one-handed free-space gesture into discrete components that together form an abstract spatial representation of the gesture. The components are parameterized with respect to duration, to support a priori predictions of the entire gesture's duration.

Experimental results show that the task model performs well both in fitting the performance of gestures and in predicting the average performance of new gestures. The model passes the heuristic engineering threshold of "80% accuracy with no more than 20% effort," which means that it is suitable for inclusion as an operator in frameworks such as the Keystroke Level Model or hierarchical task modeling frameworks. The Pantomime representation and its task model extend the scope of task modeling to gesture-controlled interfaces and establish a base for future research into gesture performance.

A Task Model of Free-Space Movement-Based Gestures

by
Shea Stirling McIntee

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2016

APPROVED BY:

Benjamin Watson	Christopher Healey
James Lester II	Robert St. Amant Chair of Advisory Committee

DEDICATION

To my mother.

BIOGRAPHY

Shea S. McIntee was born October 19th, 1982 in Worcester, Massachusetts. He attended East Chapel Hill High School in North Carolina and Durham Technical Community College before transferring to Appalachian State University, where he graduated with honors with a Bachelor of Science in Computer Science and a minor in Anthropology in May of 2006.

He was then invited into the graduate program at the Department of Computer Science at North Carolina State University, where he worked as a teaching assistant while studying the field of human-computer interactions, receiving his Master of Science in May of 2013 and his Doctor of Philosophy in Computer Science in December of 2016.

ACKNOWLEDGEMENTS

Many people have contributed to this thesis. My parents, Mark and Sally McIntee, who showed at times endless patience with my enthusiasm, my anxiety and my desire to talk for hours about my ideas and concerns. My brother, Tomas McIntee, who gave advice and feedback from the perspective of a different discipline.

My grandmother, Georgia Smallman and the folk dancers of Raleigh and Chapel Hill International Folk Dancing, who got me interested in gestures and movement and were willing to let me bend their ears about my research and life concerns.

My advisor, Robert St. Amant, who always had feedback for me and believed I could do it even when I myself didn't. It's been a long road from the semester I asked him to be my advisor, and he's been there the whole way with honest advice on everything from experimental protocol to writing to life concerns.

The faculty of the Computer Science department, with whom I've shared many semesters as both a student and a teaching assistant—in particular, the members of my committee, who saw me grow from a student who knew nothing to one that knew a bit more than nothing, and helped guide my work from a crude proposal into a viable thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

Until the mid-2000s, everyday interaction with a computer meant using a keyboard and mouse as input devices. Today, thanks to advances in software and hardware technology, we have touch-sensitive surfaces, cameras, accelerometers, speech systems, eye trackers and even brain-computer interfaces—input to computers has expanded enormously in scope and variety.

Gestures are one of these alternative forms of input. With the emergence of commercial camera-based devices such as the Microsoft Kinect and Leap Motion, a wide range of new applications can be seen that exploit gestures for input, from gaming to augmented and virtual reality. From their constrained and fragile beginnings decades ago, with requirements of a data glove [SZ94] or a room-sized environment [Bol80; Zim87], gesture-based systems are starting to move into the realm of common usage.

What is a gesture-based system? In the literature of human-computer interaction (HCI), the general term “gesture” encompasses a broad range of different types of input. The fundamental domain is that of movement or position of the fingers, hands and arms; other body parts may also

gesture [Vel15], but rarely do. Gestures can be performed in three dimensions or against a surface; they can be performed with the body alone or using a device such as a mouse or a touchpad. The movements can be *manipulative*, in which the movement itself accomplishes a task, such as in the working of a Rubik's Cube, or accomplishing the task with a virtual analog to a physical object. They can consist of *poses*, in which the arrangement of fingers or arms holds meaning, or the movement itself is a channel for communication. There are ways to categorize non-manipulative gesturing based on informational content; one way (into *symbolic*, *relational* and *communicative* categories) is discussed in a later chapter.

Billinghurst [Bil02] notes an important quality of gestures, to distinguish them from movements in general in HCI: a gesture communicates information through the movement itself, rather than through the destination or a result of the movement. For example, pointing to a panel with a finger is a gesture, while using the same finger to press a button on the panel is not.

Research on gestures has a broad base in HCI. Designers and those interested in gestures for communication or control explore and expand classifications such as that outlined above (e.g. [Que96]), which can lead to guidelines for the use of gestures in implemented systems (e.g. [Saf08]). Gesture recognition researchers work on capturing and interpreting the information content of gestures. This can be challenging, given the qualitative differences between sensing devices: gestures may be recognized by analysis of video data from a camera [Dha06], or of accelerometer or gyroscope data from a handheld device [Pan10], or electromyography data from an armband [Zha09], or other sources [HH09]. Yet other researchers focus on the human performance of gestures. Given a system that captures and recognizes gestures, what can be said about the efficiency or effectiveness with which people can carry out those gestures?

This last approach falls within the domain of *task modeling*. A task model "describes a set of activities which the user can perform to reach goals in a specific application domain" [Pat97]. Said differently, a task model is a structured description of observed and inferred actions carried out by a human user or operator of a machine in some environment to achieve goals.

Task models are of value in HCI for two main reasons. First, they can give designers insight into the structure of a task carried out by users interacting with some computer system. Even a coarse model of conventional computer use might show, for example, separate clusters of key presses and of mouse movement, or perhaps their interleaving for some tasks. Such insights can suggest refinements or improvements to a given interface design. Second, task models can provide predictions of human performance, often in advance of the implementation of an interactive system. John [JK94] writes that "engineering models in HCI should strive to predict the average user behavior to within 80% accuracy with less than 20% of the effort of prototyping and testing... Engineering models must be detailed enough to attain this level of quantitative prediction." Task modeling has demonstrated a strong predictive ability for human interaction using a range of commonly-used devices such as keyboards and mice [Mac91], and has on occasion led to significant cost savings [Gra92]. Task modeling has recently made forays into the other input modalities mentioned above [SW13], but gestures are a relatively new area.

The whole domain of gestures is very broad; a task model that covers (as per one of John's optimization criteria for engineering models) the whole of the gesture space would likely need to be complex and incorporate several sub-models. Therefore, for the purposes of this work, the gesture domain space is narrowed to much smaller domain, that of *movement-based*, *symbolic*, *free-space* gestures. That is, gestures that have a fixed shape (and no pose element) performed in empty air.

The movement-based constraint removes the need to model pose. While poses are able to communicate information as well, there's little evidence of much difference in performance between poses; moreover, transitions between poses are 'functional' and recognition of hand pose is more difficult than tracking a hand. Non-free space gestures, those performed against a surface (such as a PDA touchpad or Pocket PC [Wob07], often use a stylus and the scale of movement is small, sometimes only including significant movement by the fingers. In addition, the surface may act as a 'guide'; free-space gestures, then, would be the 'purest' form of gesture. Symbolic gestures, such as drawing a triangle in the air, have a fixed path—unlike conversational, relational or manipulative

gestures, which have a less fixed shape—the ability to distinguish fixed paths allows distinguishing gestures and both classification and comparison (explored later, in Chapter 3). One last qualification is that the gestures are *unpaced*, not carried out under time pressure.

1.1 Thesis Statement and Contributions

While there is extensive literature concerning gesture recognition systems, relatively little research has been done on task models of gesture performance. While some studies of gesture recognition systems [Wob07] report mean gesture performance times, they go no further. There are some task models of drawing, such as the highly restricted Drawing operator of the Keystroke-Level Model [Car80], the straight line reduction of Isokoski [Iso01], and the Corners, Lines and Curves model of Cao & Zhai [CZ07]. There are extensions of task modeling to gesture in the form of a constant-time operator, such as the Gesture operator (modeling the shaking or movement of an accelerometer-equipped phone) of Holleis et al's application of the KLM to mobile input [Hol07]. But there are few other task models attempting to tackle the domain of gesture more broadly, let alone the free-space movement-based gesturing under discussion.

A task model of gestures will provide value in three ways. Most directly, it can be used to make predictions about the performance of tasks carried out in a gesture-based interface, to inform interface design, much as GOMS has been used to assist the design of a number of commercial and non-commercial systems [JK96a]. An accurate task model can be used indirectly to examine some aspect of human physical or mental behavior. Finally, a task model of gestures provides a baseline to quantify the impact of qualities such as age [Hou04] or limited sensory feedback [Cao08] upon gesture-based interfaces.

Within these ways there's a number of specific benefits a task model of gestures could provide. For example, given a set of gestures and a set of actions, an *optimal mapping* could be developed based on the predicted performance of the gestures and the frequency of the actions in application use-cases. Quantitative predictions of gesture performance can allow a gesture interface to be

compared to an alternative non-gesture interface, such as a touchpad. A prediction of gesture duration could be used to determine how long a system would be waiting for human input - and thus lead to a time-efficient arrangement of delayed calculations. Or a qualitative task model may highlight possible points of ambiguity in a gesture set—pairs of gestures that could be misrecognized and lead to errors.

The thesis of this research is as follows:

> *Task models can represent a user's execution of free-space gestures to make predictions of duration at a level of granularity and accuracy sufficient to act as engineering models of performance.*

The contributions of this dissertation are as follows:

- *A simple model of movement-based gestures called Pantomime.*

  The Pantomime representation discretizes the path of a gesture into fixed pieces, called components. The sequence of those components composes the gesture representation. Pantomime then reduces the representation into a task model that can be used *a priori* to predict one-handed gesture performance.

- *Identification of dimensions along which a representation of gestures can be evaluated for the purposes of task modeling.*

  Evaluating representations is difficult: usually they are only judged indirectly, by answering the question, "What can the representations be used for?" and considering the value of the answer. In modeling literature, a representation similarly is judged through the task models it enables; a useful task model is considered to demonstrate the value of the representation it came from. By filtering the requirements of the representation's domain through a set of categories of usage (communication, classification, comparison and calculation), a representation can be evaluated directly.

- *Experimental validation of the Pantomime task model, demonstrating that it fits the "80%/20%" benchmark for engineering models proposed by Bonnie E. John.*

  Experimental results demonstrate that the proposed component count model, with a prediction error of 20.2%, fulfills John's accuracy requirement for a useful engineering model. When tested against two competing models, the mean-value model and the undifferentiated stroke count model, the component count model showed a superior fit, with an RMSE 14% lower then either of the others.

1.2 Overview

This dissertation contains five chapters, which outline the background of task models relevant for the developing of models of gesture, develop a representation (Pantomime), transform the representation into a task model, and describe a series of experiments used to define and validate that task model. The dissertation ends with an overview and descriptions of directions for future work.

Chapter 2 introduces task models on different scales. On the lowest end, Fitts' Law is a task model that relates the time to perform a selection movement to two variables—the one-dimensional width of the target to select, and the distance to the center of the target. Moving up in complexity, collections of task models and a set of rules to apply them allow the Keystroke-Level Model and other parts of the GOMS family of techniques to model more complex behavior, up to and including goal-directed tasks such as "delete a file" or "find a web page".

Chapter 3 describes the development of a representation of movement-based gestures, Pantomime. After covering a classification of gestures by context, the chapter discusses four categories of usage relevant to the domain of gestures, analyzing how existing models and representations implement each. The second part of the chapter is concerned with Pantomime, a novel representation of movement-based gestures, ending in the formulation of a task model equation, the component

count model. The chapter finishes with a discussion of future directions Pantomime can go in and how it implements each of the categories of usage previously-described.

Chapter 4 describes three experiments, one using the Leap Motion camera-based device and two using the Microsoft Kinect camera-based device to capture hand movement. The Pantomime task model and two competing models, a constant mean-value model and an undifferentiated stroke count model, are brought over from the previous chapter to be tested against the data. The first experiment, a pilot study with the Leap Motion, is used to examine the different approaches, while the second experiment, with the Kinect, is used to inform the three models. The final experiment, using the same device, environment, and testing procedures as the second experiment, is used to compare all three informed models against new data to test the utility of the component-based approach of the Pantomime representation.

Chapter 5 summarizes the work, placing the representation and task model within the context of the existing body of modeling. Directions for future work are discussed, including the extension of the representation and its task models beyond the existing boundaries and in future studies that use the results of this work as a baseline to further explore human gesture input.

CHAPTER

2

TASK MODELS

Task models provide a structured description of the way that humans perform complex goal-oriented tasks. Different types of task models include different sets of primitive operators, as well as imposing different constraints on the ways that operators can be combined, but modeling frameworks used in HCI typically include basic physical actions carried out by computer users: pressing keys on the keyboard, moving the hands between the keyboard and a pointing device, moving the pointing device to select targets, and so forth.

Some task models include little more than this: The Keystroke-Level Model (discussed below) simply has a mental preparation operator to account for "thinking time". Other task models include some abstraction of the planning and other cognitive activity that is part of the task—for example, GOMS (also discussed below) implements a system of goals, subgoals and selection rules to organize and direct the model's behavior.

Task models offer a number of benefits to HCI [ROO90; JK96b]: They can be used to identify constraints on the interface design space; they support specific design decisions; they can estimate performance time and performing analysis; they can guide training processes and help systems; and they can direct research towards the parts of performance that require the most scrutiny. Understanding initial constraints on the design space for interfaces lessens the time wasted on interface designs that are unlikely to be successful, while the ability to decide between versions of an interface without needing to implement both and conduct a study saves time in the design process. Both roles of modeling are valuable in the overall design process. Performance time estimates are useful not only for design decisions but for usage decisions—Olson mentions determining the number of workers needed, but it also is useful for determining necessary system responsiveness and other criteria for the overall computer system. Training needs are likewise important for usage decisions, albeit in a different fashion.

This chapter begins with a modeling operator incorporated into most task models in HCI: selection of a target with a pointing device. The sections that follow describe the three task modeling approaches most commonly used in HCI: the Model Human Processor, the Keystroke Level Model, and GOMS. The chapter ends with a discussion of how gestures can be integrated into such modeling approaches.

2.1 Fitts' Law

A predictive model of gesture performance should follow in line to similar successful past models. Towards this end, this section covers perhaps the most-studied model of human performance in HCI, Fitts' Law. The classic Fitts' Law explains the connection between the time to perform a physical targeting movement such as moving a cursor to a destination, the size of the target and the distance from the starting point to the center of the target.

In the decades since it was first proposed, it has been validated for a range of similar motor tasks, used to explain variations in human performance and applied to design decisions and the creation

of later physical action models in the field of cognitive modeling.

In particular, a variation of Fitts' Law called the Accot-Zhai Steering Law [AZ97] demonstrates how the original formulation, which describes a very goal-based motion, can be applied to a movement in which motion is not necessary for functionality. This suggests a possible progression from the Information Theory base of Fitts' Law through pointing, pathing and finally to movement-based gestures. Unfortunately, additional characteristics of gestures make a direct application infeasible; these include a lack of defined characteristics such as target location and path width (or target size and width for classic Fitts' Law) along with the Steering Law's weakness to angles close to 90°.

Developed in the early 1950s, Fitts' Law [Fit54] can be considered the gold standard for human performance models; although Fitts' Law itself is narrowly-scoped (to "selection tasks") and defines the relationship of three variables (task duration, target width and distance to target), its simple structure hasn't stopped it from a high degree of accuracy and broad usage beyond the original tasks used to develop its parameters.

The fundamental nature of Fitts' Law is the observation that in an one-dimensional pointing task, performance time is a result of the distance to the target and the inverse of the width of the target; that is, the farther away or smaller the target is, the longer it takes to select. Another constraint is that the movement is controlled through the whole process, as opposed to a ballistic movement in which there is no control aside from the initial impulse; what is sometimes called a "closed loop" movement [WG08] . It's usually expressed using 'Shannon's Formulation' [Mac92], which has an almost perfect positive correlation (0.95) coefficient with experimental observation in study after study.

The name for "Shannon's Formulation" comes from the similarity to an equation in Shannon & Weaver's book "The Mathematical Theory of Communication" [SW49], the Shannon-Hartley theorem, which describes the theoretical maximum amount of information that a channel (originally applied to electrical signals such as telegraph and radio) can communicate with an insignificant level of error. In it, C is the capacity of the channel, B is the bandwidth, S is the signal power and N is

the noise power.

$$C = B \log_2((S+N)/N)) \tag{2.1}$$

The Shannon Formulation of Fitts' Law has a similar structure and 'inherits', conceptually, the idea behind the Shannon-Hartley theorem of there being a logarithmic relationship between an amplitude (signal power for Shannon-Hartley and the distance to target for Fitts' Law), a constraining factor (noise power for Shannon-Hartley and the size of the target for Fitts' Law) and the resulting performance. The capacity of the channel becomes, in Fitts' work, the "index of performance", the bandwidth is inverted (due to the inversion of the meaning of amplitude and signal power) and becomes "movement time" and the logarithmic term that uses signal and noise power becomes the "index of difficulty" (and thus IP = ID/MT).

$$I_p = -1/t \log_2 W_a/2A \tag{2.2}$$

Since Fitts, the Shannon Formulation has undergone revision [Wel60; Mac91]; now the equation solves for movement time and the doubling of the distance to prevent a negative index of difficulty has been replaced in modern usage by adding a constant.

$$MT = a + b \log_2((A+W)/W) = a + b \log_2(A/W+1) \tag{2.3}$$

Some have replaced the width with "effective width" [Jud16], with the target resized after the experiment to have a standardized 4% error rate, while others have experimented with splitting the single index of difficulty into two terms, each dependent upon one of the two variables, to model the targeting task as a "two impulse" (movement and homing) action [Wel69].

2.1.1 Fitts' Experiments

Fitts ran a study for each of three experimental tasks - reciprocal tapping, disc transfer, and pin transfer. Reciprocal tapping involved the participant tapping alternate metal plates with a stylus repeatedly. Two versions of the experiment were run, one using a single ounce stylus and the other with a one-pound stylus. The experiment varied the width of the plates by powers of two (2 inches, 1 inch, 1/2 inch and 1/4 inch), with a fixed second dimension of 6 inches, while the distance (amplitude) between the targets was similarly tested at 2, 4, 8 and 16 inches. Errors were caught with additional metal plates in front and behind each of the target plates. This task formed the basis of most future experiments testing extensions and modifications of Fitts' Law.

Of the other two, disc transfer involved the movement of plastic washers from one post to another, not unlike the steps of a classic Tower of Hanoi. Here the variance was across the size of the holes in each washer and the distance between the pair of pins. Pin transfer was an inversion of the disc transfer task; pins were transferred from one hole to another, with the variance being in the size of the pins (the holes were always twice the size of the pins) and the distance between the columns of holes.

The two domains compared by Fitts, electrical signals (Shannon-Hartley) and the human nervous system (his own work), parallel conceptually as well as mathematically. When examining human motor performance, the amplitude of the muscle contraction signal that passes through the nerves is analogous to the signal power of a communication. Meanwhile, a nerve signal passing from cell to cell is subject to what's called "neuronal noise", which is a variance in the strength of the signal based upon the strength of the signal (studies of this noise also refer back to Shannon's work [Ste05]). This neuronal noise isn't measured directly in Fitts' original formula (or variations that appears afterwards), but it results in a lack of accuracy in targeting that impacts whether a given strength of nerve impulse results in the focus successfully contacting the target.

The human brain has had years of practice in achieving high accuracy in pointing-type tasks, and thus chooses an amplitude of nerve signal that would result in an accurate landing; thus neuronal

noise shows up indirectly through the compensation the human brain applies to the speed of the selection task performance. The contrast to a robotic performance of a motor task such as bouncing ping-pong balls on a paddle can be seen; the robotic arm, using straight electrical signals, can move at maximum speed while the human arm, dealing with additional noise, cannot.

The criteria of "expert performance" that underlies not only Fitts' Law but many later models such as the Keystroke-Level Model, comes from greater consistency of performance of practiced users - this in turn may call back to the lack of monitoring over actions with an assured result mentioned by Welford [Wel60] in the context of the single-channel hypothesis; that is, that since an expert performer knows the outcome of a given muscle impulse (or set of impulses), there's no delay to process sensory data to check for success.

Fitts' hypothesis was that movement time would vary with task difficulty (which is based upon the distance to and width of the target) such that the index of performance (I_p) would remain constant. As referenced in his tables, for each of his three experiments the I_p varied to some extent (within about 10%) with a single significant outlier in the reciprocal tapping and disc-transfer tasks (50% and 25% respectively). He also found that about 3% of the task attempts resulted in error (missing the target).

2.1.2 Later Fitts' Law Research

Since the original experiments, Fitts' Law has been applied to a wide range of pointing tasks, from area cursors [KB95] to click-and-drag [Mac91] to scrolling [Hin02]. It has formed the foundation for some of the KLM operators later discussed [Car80] and has been shown to apply to selecting cell phone buttons [Sil00] and head control interfaces [LoP00], as well as comparing different devices or interfaces [MZ99; SW13]. Many recent studies have turned to using Fitts Law or the later Steering Law as a way to quantify the performance impact of human variation, from visual limitations or distortions [Cao08; TK08] to handedness or motor impairment [WG08; Jud14] to the impact of age on performance [Hou04; Car15].

When conducting Fitts' Law experiments, the particular constants (a and b in the equation) are determined through regression, and often the suitability of the equation is demonstrated with a coefficient of correlation (r) [KB95] or of determination (R^2) [Mot01]. Across Fitts' Law studies, coefficient of correlation values for the model against the data tend to be above .9000, and Fitts' own data offered $r = .9831$ when later analyzed [Mac92]. Standard error of the slope (b in the MacKenzie variation of the Shannon Formulation) has also been used to consider the fit of the equation to a particular data set, and often ends up being around 10% to 15% of the value [Mot01; Pas06].

However, MacKenzie [Mac92] notes that while in theory an index of performance ($1/b$) should be constant for a given device, it actuality it can fluctuate significantly from experiment to experiment. Some of this variance can be removed by in-study comparison of devices (as a similar ratio of IP exists across studies), while the amount of movement by larger muscle groups in any given experimental protocol can influence performance (as smaller limb group usage tends to show a higher IP). The latter may be partially the cause as well of the low fit ($R^2 = .51$) reported for 2D and 3D pointing with the Kinect [Pin13].

2.2 The Model Human Processor

The Model Human Processor (MHP), developed by Card and Moran to model a human brain's characteristics to inform good design decisions [CM88], is a "simplified architecture of the user" inspired by the work of Allen Newell that contained interacting modules that gave rise to conclusions about how a human being would react to events occurring during task performance.

The model of the human (Fig. 2.1) devised by the pair contains both short- and long-term memory, processors for each of cognition, perception and motor as well as image stores in short-term memory for auditory and visual data. Information comes in through the perceptual processor, travels through the short-term memory stores for sensory information, the human user acts upon the information, pulling additional information from long-term memory and performing calculation on the contents of the short-term memory. When a conclusion is reached, the motor processor

Figure 2.1 Figure 3 from *User Technology: From Pointing to Pondering*, Card and Moran, 1988

carries out the action planned in short-term memory.

Each of the modules in this model are given cycling speeds—how fast each module will process to the next step. Other constants, such as the number of things that can be stored in short-term memory, are also included in this model. The goal of the system is to make predictions not about motor tasks as Fitts' Law does but for the cognitive side such as reading and performing calculations.

2.2.1 TYPIST

While the Model Human Processor's focus is cognitive and not motor activity as models like Fitts' Law cover, it can still be used to make meaningful physical task performance predictions. In TYPIST [Joh96], Bonnie E. John creates a model of expert-transcription typing within the MHP's framework and tests it against a set of transcription tasks. Assumptions brought in from existing theories of typing, such as the chunking of letters during reading and the same-hand constraint are matched with those from the MHP such as the serial/parallel processing of the three processors. TYPIST takes the transcription task and places the operators in a schedule chart using the three processors to determine limitations and bottlenecks; the total task performance is thus dependent upon the critical path determined by the scheduling.

When applied to the same transcription tasks and compared to experimental data, TYPIST performs quite well. In the case of restricted preview (when the typist cannot look ahead), for example, TYPIST has prediction error against the data of 15%. The "stopping span", the amount of typing done when a task is abruptly halted, had a prediction error of 9.9% (for a word) and 3.7% (for a sentence); overall prediction error for task manipulation as a whole was 8.6%. Over all the tests, TYPIST predicted within 20% of the measured values for an average-speed typist and predicted 9 of 12 pre-identified phenomena of typing.

2.3 The Keystroke Model

In 1980, Card, Moran and Newell introduced the Keystroke-Level Model [Car80], a simplification of their ongoing work with the MHP and GOMS. Designed as a simple task model to make predictions for user performance times arising out of Card and Moran's work in the '70s at PARC [CM88], the KLM divided up user tasks into simpler task models, called operators, and gave an experimentally-determined value to each. The task model then calculates the total values across all operators and uses that as a prediction. Limitations of the model are similar to those found in other models, such as Fitts' Law; task performers are assumed to be practiced with the system and the task ('experts'), the tasks are defined as being routine and predictions are limited to the particular scope defined by the operators. Errors are not modeled in KLM due to the additional level of complication the attempt would introduce; KLM's introduction states that the authors were unable to predict when and how often errors occur, although they did note that in the case of an error, KLM could predict the time to correct. Learning and other complex mental activities are not modelable under KLM, as it only contains a single operator to cover mental activity.

Testing of the KLM model against a set of benchmark tasks revealed a prediction error ranging from 0% to 26% for text editor systems and 1% to 40% for graphics-based systems. Operator-specific variances were not reported, other than for the Mental operator (1.35 seconds mean, estimated standard deviation of 1.1 seconds).

The KLM contains two levels of abstraction—the *method* and the previously-mentioned *operator*. The operator is one of six possible atomic actions (keystroke/button press, pointing to a target with a mouse, homing a hand onto the keyboard or mouse, drawing straight line segments with a mouse, mental preparation for a physical action and system reaction time), while the method is a sequence of operators that together accomplish the target task. The KLM's predictions are made with a simple equation—total task performance time is the sum of the performance of each of the sequence of operators the task is composed of. The target task is often called the "unit task" and is specified as a

small “cognitively-manageable, quasi-independent” task. Larger-scale modeling, such as the Goals (methods writ large) and Selection Rules (to choose among methods) of the GOMS architecture developed at about the same time by Card and Moran (along with Allen Newell), are not within the KLM’s scope.

Card and Moran’s introduction to the KLM is very careful to mention that of the multiple possible ways to measure performance (those mentioned include performance time, error rate, learning time, functionality, recall, concentration, fatigue and user subjective evaluation), the model only claims to evaluate a system based on performance time; most other cognitive techniques and architectures, from Card, Moran and Newell’s own GOMS (and later GOMS variants) to Anderson’s ACT-R to Sun’s CLARION likewise usually makes performance predictions on performance time, although those with a robust mental component, such as NGOMSL (using Cognitive Complexity Theory), ACT-R and CLARION (which are full-blown cognitive architectures) are able to make predictions for other metrics.

Another weakness in KLM as well as earlier GOMS is the restriction to a singular “track”; it isn’t until later GOMS variants such as Bonnie E John’s CPM-GOMS (standing for both “cognitive-perceptual-motor” or “critical-path-method”) that human multi-tasking is modeled.

2.3.1 Physical Action Operators

The Keystroke operator, the namesake of the task model, is the most straightforward to use and calculate. Each press of a button or key has a value determined by practiced average typing rates (0.2 seconds for a “skilled typist”, for example). Special cases, such as random letters, complex codes or unfamiliarity substitute their own values. For a string of characters, the number of keys pressed are added up and multiplied by this value. The Homing operator is, like the Keystroke operator, a constant value, representing the average time to move a hand between devices (the keyboard and the mouse), given as 0.4 seconds.

The Pointing operator has two values that can be used, depending on how close to ‘typical’ of

a mouse pointing task a given usage is. An averaged constant value, 1.1 seconds, is used in most circumstances for convenience, although the underlying calculations derive from Fitts' Law, giving a claimed range for mouse usage between 0.8 and 1.5 seconds. The formulation used for measurement is the Welford variation of Fitts' Law, which adds 0.5 to the inside of the logarithmic term. Explicitly this operator is only for the pointing task itself; pressing a mouse button is covered by the K operator. Fitts' Law may also be applicable to a customized Homing operator, as that too is a form of pointing task.

The Drawing operator is the least used in the literature discussing KLM, and is even replaced or dismissed in several adaptations to other hardware environments [Hol11]. However, for the purposes of gesturing it is perhaps the most interesting of the original six. Specified as a "very restricted" operator, it models the drawing of a sequence of straight line segments on a 0.5 cm grid on a screen with a mouse. The average performance value is derived from both the number of segments and the total length (0.9n + 0.16d, where 'n' is the segment count and 'd' is the distance).

2.3.2 Mental Action and Response Operators

One of the key elements of the KLM is the 'M', or Mental operator. Unlike the four physical operators, it is not placed in response to a necessary physical interaction in a method but inserted into a KLM sequence according to a set of rules after all physical interaction has been determined.

The first rule inserts an M in front of Ks and Ps. Then a set of four rules are used to remove most Ms that are for Ks and Ps whose preparation would be done while the previous operator was acted upon or where the next operator was chunked with the previous (for example, a string of keys would only have an M operator in front of the first keystroke).

The end result is meant to model the points at which the method performer would take some time to position or mentally retrieve the next step of the plan. It is important to remember the limitations of KLM previously specified–the task must be routine and the user expert. Long pauses to consult instruction sheets or reason through a mental heuristic to determine the specifics of the

next action are not covered by the Mental operator and its constant 1.35 second duration.

The last operator, 'R' or Response time, is deliberately kept variable, since it depends upon the processing performed by a particular system and not the method performer. The performance value for R is skipped when it's covered by a Mental operator or is otherwise not resulting in an actual wait time.

The R operator itself provides value to a designer in conjunction with the M operator, in that if a system designer can predict when there's mental preparation they might be able to slip a second or so of system calculations in without actually impacting the user's workflow or the responsiveness of the system during a task.

2.3.3 Operator Refinement

The first way in which the Keystroke-Level Model can be refined is validating, replacing or specializing the six operators the basic model specifies. Sometimes this operator refinement occurs on the keyboard-and-mouse interface model of the original study, and sometimes it occurs in the transition to a different interface model.

While the basic keystroke operator's range of constant values (from 80 milliseconds for 'best' typists to 280 for 'average non-secretary' to 1200ms for 'worst' typists) has been confirmed in later literature, refinements for specialized situations have been offered. Olson and Nilson [RON89], for example, give single-constant values for formula entry in specific applications (Lotus and Multiplan). Similarly, Olson and Olson mention a specialization of a specific form of the Pointing operator in navigating a drop-down menu done in an unpublished manuscript by Walker, Smelcer and Nilsen.

One of the most-examined aspects of the KLM is the placement of Mental operators. Lane et al [Lan93] uses a common combination of menu selections in Lotus 1-2-3 to argue that while classic KLM may require (depending on the interpretation of "if the keystroke is fully anticipated") a new Mental operator for each of the three selections to complete a single command, the actual process for a practiced user of the application suggests that a single Mental operator for multiple selections

in a single menu hierarchy is more appropriate. The difference in conclusion from an earlier work for the same combination in the same application elsewhere in the literature [ROO90] was the more extensive experience with the application as a whole of their subjects in comparison to the earlier Olson study.

A comparison of KLM predictions to data using a handheld device [TJ06] sought to determine over- or under-estimation of separate operators *in situ*, and found that the Mental operator value initially used (1.2 seconds) resulted in a less accurate result in cases where the system response time value was under the Mental operator constant value; a decreased in the Mental operator constant to 940ms eliminated the error almost altogether. However, there was no follow-up experiment mention to validate the new value or to determine under what circumstances the smaller value may apply.

2.3.4 Additional Operators and Domains

An advantage of the discrete operators of KLM is the ease of expansion to other domains. The procedure is simple—existing applicable operators are tested for an altered constant value and new operators added in a similar fashion as they arise. Thus KLM has been applied to domains beyond the original keyboard-and-mouse arrangement. This often gives rise to the second way in which KLM can be refined—the addition of new operators.

Early mobile phones were popular for expansion given the small devices' need for text input on either a shrunken keyboard or alternative methods. In a 2004 study on Korean text entry methods [Myu04], Rohae Myung applied two of the six operators to the task—Keystroke, which as per the recommendations of Card had a value derived from Fitts' Law, and Mental, which was split into multiple sub-operators on the specific mental tasks. These sub-operators, recognizing an intended character, recalling the production method and confirmation of the correct result, are given different values. The summation of these values is still under the standard value for keyboard-and-mouse; it is argued that the shorter saccadic distance for a small device is responsible. The study not only confirmed the fitness of the KLM adaptation but also allowed a design choice between three prospective

text entry methods by comparing both model and experimental data for each.

In addition to a differing layout of keys and buttons, with mobile devices came touch-sensitive surfaces and styluses. Luo and John [LJ05] applied KLM to stylus-based devices, particularly the Palm PDA, with strong accuracy (3.7% average error between prediction and experimental data). Four different methods to complete navigation task were tested—tapping icons on the screen, a virtual keyboard, the graffiti character input method and a scroll bar. The KLM models for the task were produced using John's CogTool application and were converted to a different more complete architecture, ACT-R through an intermediate form called ACT-Simple. The Graffiti input method used a new specialized operator with a static value of 580ms.

With the embedding of movement sensors in mobile devices comes movement-based commands such as gestures. Initial forays into the adaptation of KLM to gesture-enabled devices as done by Holleis et al [Hol07] serve to integrate the additional command mode in to the existing operators seen elsewhere in KLM literature. The Drawing operator of the original six was removed and several additional operators were added for this model—Gesture (a constant value of 0.8 for movement-based commands), Finger movement (a form of micro-Homing operator), Initial act (retrieving the phone), a trio of focus-based operators (Macro Attention Shift, Micro Attention Shift and Distraction) and an Action operator that acted as a catch-all for reading RFID tags and other unique actions unable to be broken down. Pointing is adapted to physically moving the phone to a desired absolute location and Homing to moving the phone between by the ear and out in front of the body. Prediction error was not given per operator, but overall task prediction error per data sample ranged from -15% to +8%.

The addition of many new operators, particularly designed to model environmental impact upon the task, suggests that KLM may not have been the best architecture to use for this study. While the prediction of performance was quite close to the experimental performance in this case, the additional Distraction operator, in particular, was set as a multiplicative factor rather than an additive one and required a subjective evaluation for "slight" or "significant" levels of distraction.

2.3.5 Expansions of the KLM

KLM lends itself well to automated tools because of its simple linear structure. The CRITIQUE tool [Hud99], for example, logs an operators example performance, generates a Keystroke model and then converts that model (with some operator assistance) to a more complex GOMS model. The Mental operator placement is simplified, with insertion occurring as in Kieras' adjustment to Card's original "add-and-then-remove" process and only two conditions—when interaction switches to a new object or when the type of interaction (point-and-click, typing) changes; unspecified exceptions are made to this pair of rules. The Homing operator is also inserted as interaction changes from mouse to keyboard. Finally, the K operator is split into Keystroke and Button operators with refined constants acquired from Kieras. The model generated by the tool was equivalent in prediction to a hand-generated model in spite of the difference in Mental operator placement.

2.3.6 Application of the KLM

The Keystroke-Level Model was validated upon text editors [CM88], as that was the focus of Card and Moran at the time, and demonstrated a strong correlation between prediction and observed performance across several text editors and graphics editors.

One of its first major applications was the Xerox Star mouse—in particular, determining the number of buttons the mouse should have. As the product was still in development, and there were a variety of different proposed methods to control the interface using the mouse, there were no expert users to run a study across the different possible schemes. Thus KLM became instrumental in the two-button mouse arrangement that still dominates most mouse usage interfaces to this day.

John and Kieras, in a survey of GOMS methods and uses [JK96b] mention several other times the Keystroke-Level Model was used to assist serious large-scale design processes. Bell South's telephone operator protocol was changed from keying in only a few letters and visually scanning many results to keying in more letters and visually scanning fewer results through the demonstration of the improved performance of the latter; current search practice at NYNEX follows this recommendation.

A redesign of a space operations command and control system was able to run through many iterations of testing using KLMs to allow early design decisions that didn't need revisiting after user-stage testing. And a port of a CAD software package to Mac was able to be significantly streamlined due to KLMs being used to model and test variants of the current interface.

Each of these successes led to future iterations containing the same KLM-driven design decisions used up until the publication of the survey and even further into today. It's clear that cognitive modeling in general, and Keystroke-Level Models in particular are useable and effective for application design and evaluation.

2.4 GOMS

The Keystroke-Level Model is in some ways half-way between a physical task model such as Fitts' Law and one that includes a cognitive model such as GOMS. It describes a whole range of human activity rather than a narrow specific task and has rules for development of models beyond an equation. However, the emphasis still is entirely upon actions—even the Mental operator (and other cognitive operators developed in later extensions) is treated as a piece in a jigsaw puzzle and cannot be used to in a broader context.

Because of this, KLM is usually used either as a foundation for a tool that converts a KLM model to one in another architecture or to answer narrow design decision questions and provide performance estimates; two of the five uses described by Olson and Olson mentioned earlier. However, applying the operators and revisions to a Keystroke model is not difficult due to the concurrent development (by the trio of Card, Moran and Newell) of GOMS [Car83], a more flexible and broader model based on the MHP that also incorporates KLM's operator-based structure.

GOMS is an acronym, standing for "Goals, Operators, Methods and Selection rules", the four parts of a GOMS model. Operators are just as described by the Keystroke-Level Model, while methods are similarly described in KLM. Goals can be considered to be methods writ large; a goal is the desired state the user wishes to reach. Goals can in turn be divided into a hierarchy of sub-goals,

descending down to the methods which encapsulate the steps to take to achieve the lowest-level goals. Selection rules are the control structure of GOMS, describing how a user decides which of multiple alternative methods to perform based on inclination, environmental conditions or current arrangement of hands and devices.

As a note, the original GOMS framework developed by Card, Moran and Newell is often referred to as "CMN-GOM" to distinguish it from other members of the "GOMS family".

2.4.1 NGOMSL

David Kieras' NGOMSL (Natural GOMS Language), is a notation for and process to create GOMS models [JK96a]. It included elements borrowed from the cognitive complexity theory architecture developed by Bovair and Kieras, and sought to apply a classification of simplicity or complexity to the methods defined in a GOMS model. Methods are divided into production rules; each production rule is invoked by and then acts upon working memory and is composed of either an "external" or "internal" operator. The former are GOMS operators, while the latter are CCT operators that manipulate working memory, retrieve information or establish sub-goals. The Mental operator seen in the KLM is dissipated among all operators and is otherwise mostly handled by the more specific CCT-derived production rules.

One of the major assets to the combination of the two theories is bringing the predictive power of GOMS to bear upon not just performance but also learning times, through a simple count of NGOMSL statements and the amount of declarative information that needs to be memorized. By comparing the new method to existing known methods, the amount of learning time that can be cut short through knowledge transfer can also be estimated. The trade-off is a very strict single-track action sequence; cognitive parallelism is not possible within the constraints of cognitive complexity theory, and neither are methods that are 'broken up' or can be rearranged in multiple ways.

2.4.2 CPM-GOMS

Bonnie E. John's CPM-GOMS (Cognitive-Perceptual-Motor GOMS, or alternatively Critical Path Method) is much like NGOMSL in that it seeks to merge GOMS with another architecture—in this case a schedule chart from which "Critical Path Method" derives. Performance is split among three 'tracks' pulled from the Model Human Processor (cognition, perception and motor action); the schedule chart is used to place constraints upon the scheduling of operators in each track based upon dependencies.

The addition of parallelism is powerful and the additional detail suggests that CPM-GOMS might approach the precision of other architectures such as ACT-R, particularly in more cognitive-based tasks. Parallelism can be spread among multiple motor tracks, with the left hand using the keyboard and the right using the mouse simultaneously. A CMN-GOMS model can be mapped to a CPM-GOMS model and then 'scheduled'. Multiple-goal situations are also claimed to be addressable [JK96a] due to the additional flexibility—only the cognitive architecture CLARION makes a similar explicit claim.

However, the increased complexity and the constraint-based approach of arranging operators on each track by dependencies results in a higher learning cap for designers and a decreased ability to automate the model-production process. Operators must act along the "clock speeds" of each track as per outlined in the MHP. The standard "expert level user" assumption is even stricter than in other members of the GOMS family, as the user is assumed to have come to the 'optimal schedule' determined by CPM-GOMS through sheer practice (although that assumption allows the pruning of some perceptual and cognitive operators).

2.5 A Place for a Representation of Gestures

A representation of gestures potentially is useful for each of these modeling techniques and frameworks. Particularly in the domains of mobile devices (equipped with accelerometers) and augmented

reality devices such as camera-equipped video glasses, gesture as an input method provides a way for users to communicate with their devices. As applications proliferate, predictive analysis of performance needs to keep pace—and in order to do so, task models of gesturing need to be developed for the existing cognitive model creation processes. Towards that end, a representation of gestures that allows the translation of analog input into arrangements of discrete components provides the first step towards gesture task models. The component-based approach already provides a layer of simplification without losing, it is hoped, essential performance data that might be needed for a task model accurate enough to stand up to the other task models modeling techniques such as the KLM or cognitive architectures such as ACT-R incorporate in their operators and modules.

In the case of the Keystroke-Level Model, a gesture task model (packaged as an operator) could, as an example be useful in a multi-modal application that uses the Leap Motion as both a three-dimensional mouse and a gesture recognizer alongside a standard keyboard. Gestures such as a circle or a zig-zag would be evaluated based on their characteristics alone, allowing choice between gestures during interface design. In addition to the gesture operator, the homing operator may need to be revised for "keyboard-to-open-space" movement; likewise, as the mouse is replaced, a new pointing operator may need to be evaluated for both the new input device (the Leap Motion) and the addition of a third dimension.

An NGOMSL gesture operator may need to incorporate different parts of the Pantomime representation to properly represent the complexity of any given gesture to allow predictions of learning times, although Isokoski's straight-line reduction model of Unistroke writing [Iso01] defined complexity by a simple stroke count. But with the same assumptions of revised operators, an NGOMSL model that includes such an operator might be able to determine training periods for using the Leap Motion to perform gestures.

The process for developing a task model along the lines of those discussed in this chapter is straightforward - once the form of the model's equation has been determined (such as the Shannon Formulation, or a proposed variant with additional terms for many Fitts' Law studies), experimental

data of the task to be modeled—gesturing in this case—is gathered. The equation is then fit to the data through regression to determine the constants that best suit the data. Fitting should be done through a statistical measure based on the residuals between predicted and actual values for each data record.

Most task model studies evaluate models and compare them to one another using either coefficients of correlation or determination (measuring fitness against the data) or using prediction error (measuring how far off categorical means the model estimate is). In Fitts' Law studies, for example, MacKenzie suggests that coefficients of correlation are usually expected to be at or above .9000, while in GOMS models Bonnie E. John's "80% accuracy with 20% effort" heuristic is often cited, respectively, for the two evaluative metrics.

For this work, the constants for the task models will be determined using the root mean square error, which is the square root of the average squared residuals between a model's prediction and the actual value. While the RMSE shares the same vulnerability to outliers as r or R^2, it has the advantage of being in the same units as the regression coefficients. Comparison between models will be done using the same statistic, and the utility of the task model will be evaluated using the prediction error against John's goal of 80% accuracy (or 20% prediction error).

Demonstrating that Pantomime fits the requirements of a representation of gestures is the task of the next chapter, in which it is translated into a task model designed to fit with the Keystroke-Level Model (and so fit the other techniques of the GOMS family) as a "Gesture Operator". Determining the task model's constants, comparing its performance to a constant-value model and evaluating its prediction error is the task of Chapter 4.

CHAPTER

3

PANTOMIME: A REPRESENTATION OF GESTURES

A task model makes estimates or predictions; supply the appropriate variables and the model will produce a target value for use. Fitts' Law produces a task's "index of performance" given a distance to target, the target's size and how long it took to perform the pointing task; alternatively, given an index of performance, distance to target and target size it will produce a prediction of duration for the same task. The Keystroke-Level Model produces a duration prediction for a task given some sequence of actions needed to complete it. A task model of gestures will have a similar form; given a gesture, it will need to produce a prediction of human performance of that gesture.

While some task models, such as Fitts' Law, are no more than a equation, more complex models

have a representation that defines the modeled behavior. This could be a simple sequence of operators contained in a method, such as seen in the KLM, or could be a set of productions, selection rules and a hierarchy of goals found elsewhere in the GOMS family of modeling techniques.

In this chapter, the Pantomime representation and task model will be introduced. First, the domain and requirements of the representation are defined, accompanied by a discussion of the different contexts or contents of gestures. There are a number of ways to to categorize gestures; for example, Billinghurst [Bil02] describes a spectrum (symbolic, deictic, iconic, pantomic, beat, and cohesive) ranging from the communication of discrete concepts to the emphasizing of some part of verbal speech. This chapter suggests a similar division, but into three categories—symbolic, relational and conversational. Symbolic gestures, what Quek [Que96] called "emblems" are akin to a word—they hold a single static meaning; for example, waving "hello" or circling a finger to tell someone to get to the point. Relational gestures scale and are often spatial in nature or depend on context with objects or directions for meaning, and include what Billinghurst called 'deictic', 'iconic' and 'pantomic'; indicating size, location, orientation or other qualities can be considered relational gestures. Communicative are what Billinghurst called "beat" or "cohesive" gestures, and what Quek called "gesticulative" gestures; these are used for conversational assistance or emphasis and often rely on words to provide the base meaning.

Second, this chapter will discuss four categories in which a representation, either of gestures or in general, can be used. These categories, *communication*, *classification*, *comparison* and *calculation*, while not claimed to be comprehensive, cover the range of tasks that a representation of gestures might be used for.

Communication, the transmission of information can refer to both human-to-human communication and between human and computer absent visual demonstration capabilities. In write-ups for gesture recognition experiments, gestures are communicated in one of three ways. The first is to use shapes that are familiar to the audience, such as letters or numbers [Alo09], and the second is to draw the paths in a table or image, often with an indicator for the starting point [Kal03; Oka02] or to

use drawings of hands or arms for the same purpose [Que96]. The third is to not describe the gestures at all [Gao04]. Meanwhile, Baudel's Charade system [BBL93] offers a Stokoe-like pictographic representation; while it accomplishes the communicative goal, the series of symbols are not defined generally—a string-based encoding would be ideal.

The second, classification, allows for the grouping of gestures by similarity or by common characteristic. Classifications can be mapped to one another to form relationships that are informative—for example, Vatavu [VZ13] applies a taxonomy of hand posture to a system that maps those hand postures to the size and shape of a grasped and manipulated object. Card and Moran's Keystroke-Level Model [Car80] functions by classifying atomic actions or operators and breaking the whole of a user's task into those operators; the representation presented in this chapter only has two 'classes' of gesture components, although features of a gesture are also useful for classification purposes.

The third, comparison, refers to the relative evaluation of two gestures for suitability for a particular mapping, a form of qualitative analysis. Comparison is a key goal of many representations across the domain of task modeling; for example, Card, English and Burr [Car78] compare different input devices not just experimentally, but by comparing the Fitts' Law regressions of each (in particular, the index of performance for each device). MacKenzie [Mac91] also tested a range of devices (mouse, tablet and trackball) on both dragging and clicking tasks and used the index of performance to compare the performance of each. Kurtenbach and Buxton [KB93] compared not just input devices (a mouse and a pen using a "marking menu" method) but compared two 'classes' of marking menu input (those 'on' and 'off' the horizontal and vertical axes). The gesture domain currently lacks models that allow comparison between gestures beyond experimental performance results for specific gestures.

Comparison along with classification also leads to the ability to identify points of potential ambiguity in a mapping; the "subgesture problem" referred to by Alon in 2009 [Alo09]. The subgesture problem, a unique (in comparison to other modeled input methods) difficulty for gesture recognition, is the case where all of the movement of one gesture can also be found, in the same sequence, in

another—potentially leading to systematic error.

The last of the usage categories, calculation, refers to the ability to make performance predictions for task performance for a gesture-based interface; the domain of quantitative predictions. Most gesture literature concerns gesture recognition; performance time of gestures is often not recorded in favor of recognition accuracy of a tested system [Par08; Que96]. The approach to calculation seen in the KLM [Car80]—of taking the expert performance of a whole being the time it takes to perform each piece—is simple and robust, and will inform the approach taken by the representation presented in this chapter. In particular, the Drawing operator which, although specified as applying only to using a mouse on a grid, can be seen as the beginning of a stroke-count model. Holleis *et al.*'s solution to the need to model gestures in a cellphone interface [Hol07] is a singular operator. The 'G' operator has a static value, 0.8 seconds, and applies equally to rotating, shaking and drawing shapes in the air. It is unknown how much variety in gesture was tested, as the focus of the paper was the whole of a "KLM for cellphones" model, rather than dynamic gesturing specifically.

Of these four types of usage, calculation is the primary goal of modeling, specifically what John [JK96b] calls "engineering models". These models provide prediction of human task performance, which in turn brings quantifiability to comparison efforts and gives answers to critical design decisions; for example, John cites as the first known example of using the KLM a case at Xerox described by Card and Moran [CM88], in which the model was applied to mouse design. Performance predictions can accompany classification of gestures, giving rise to rules of thumb that allow some of those design decisions to be made rapidly. Communication is a secondary concern; due to the visual rather than linguistic mode, abridging the need for demonstration and video capability is of great use. Comparison and classification are also of secondary importance, offering structure and additional utility, particularly in regards to how certain gesture characteristics may fit a performance environment or in 'cleaning' a gesture set of subgestures and other sources of error.

With the domain, context and usage goals established, the chapter builds the Pantomime representation and offers a pair of simple task models, the stroke count model and the component count

model, to be tested in the experiments covered in the next chapter. With the task model established, attention turns to ways in which Pantomime could be extended and how Pantomime implements each of the four usage categories already discussed.

3.1 Domain & Requirements

The target domain of the representation and task model is, as mentioned previous, that of free-space movement-based gestures. That is, gestures performed by the movement of the hands and arms against no surface and without meaning encoded in the position of the fingers, hands and arms. A suitable representation is likely to be able to be applied to movement in general, although unlikely to find use outside of the hands and arms due to joint constraints and the rarity of interfaces for heads, legs or other body parts.

In the case of a gesture that encompasses both movement and pose, as per Quek's preparation-stroke-retraction model [Que96], the representation would need to be paired with an encoding of hand posture as seen in the Charade system [BBL93]. It would also be a representation of the movement of a single focus; fine-grained movement such as flicking or rubbing fingers together covered in some recognition systems [Ame02] are better modeled as a series of static poses rather than a dynamic movement. As in the broader modeling literature, the representation will need to be paired with experimentally-determined constants to make task duration predictions.

The representation must be capable of formulating a task model for the prediction of a gesture's performance, as that is the primary goal of this work. It should allow interface designers to map the most suitable gestures to the interface's actions—either through task model predictions or through a comparison of gesture characteristics to environmental constraints. As part of this assistance, it should be usable to determine gestures that would cause errors through ambiguity—either through similarity to one another or because one gesture is a sub-gesture of another.

In both common usage and literature, gestures are often communicated through pictures or diagrams or through the use of labels; a representation of gestures should allow for an encoding of a

gesture that can be communicated to either human or machine without demonstration or other visual depiction.

Each of the mentioned capabilities are those that existing task models can allow for their domains, whether that is the selection of a block of text [MB92], the usage of an interface with keyboard [Kie95] or a pair of people bringing two items together [Mot01]. A "gesture module" added to these task models shouldn't restrict the full task model's functionality.

3.2 Categories of Gesture

Gestures have been categorized and classified in many ways, ranging from the usage of features to the context in which they're used. However, the type of information conveyed through gesture has been categorized along a spectrum that ranges from a static iconic meaning through context-driven to emphasizing verbal communication in gesture recognition literature.

It's referred to as a spectrum because of two 'poles' that exist in how the brain selects, controls and uses gestures based on context and content. In formal sign language, for example, the brain repurposes the language center of the brain, using it the same way as a verbal language—even to the point that routines that control the lips and tongue in speech are used for the nuance in signs. The other 'pole' of gesturing context, the improvised gestures used for emphasis, use other parts of the brain. The impact on performance is unlikely to shift much outside of those with particular neurological weaknesses or strengths, but it suggests that considering gestures to exist along a spectrum of context is beneficial.

How many categories this spectrum has been divided into and the terms used have varied from paper to paper. For example, Wobbrock's "User-Defined Gestures for Surface Computing" [Wob09] cites two five-category listings—Kendon's set (sign languages, emblems, pantomimes, language-like gestures and gesticulations) and Efron's set (physiographics, kinetographics, ideographics, deictics and batons). He also mentions McNeill's set of four—"iconics, metaphorics, deictics and beats".

Robert Wilson [Wil96] notes that gestures can often be bi-phasic (movement-rest) or tri-phasic

(transition-movement-rest), allowing separation of gestures through detection of rest-states, although Quek *et al.* [Que02] suggest that phases from one gesture can blend into those of another.

To a certain extent, any division of gesture by the type of information is going to be arbitrary; gestures can have layers of meaning and often visual communication comes with multiple modes and emphasis. Gestures can also be involuntary, adding additional layers of non-deliberate meaning. In neurological studies of sign language and non-sign language gesturing there are different areas of the brain that may be activated.

Here the spectrum is divided more coarsely to reflect not just the type of information and the level of formalism but the context as well—*symbolic*, *relational* and *conversational*. The scope of this categorization are *movement-based* gestures; pose-based gestures (which rely on hand-shape to communicate meaning) are a separate domain. Similarly, what Quek [Que02] calls *manipulative* gestures (functional movements that manipulate an object) and their virtual equivalents such as using a cursor to select objects or navigate a menu, are also beyond the scope of these categories.

3.2.1 Symbolic Gestures

The category of 'symbolic' gestures, also referred to by gesture researcher Adam Kendon as *quotable gestures* and by Goldin-Meadows as *emblems* [GM99] includes a range of iconic, metaphorical and emblematic gestures; the criteria is that the gesture must communicate without any context. Most gesture recognition systems function upon symbolic gestures; whether the gesture is a hand-pose or an arm-movement it is isolated and interpreted without conversational or environmental cues.

Because of their discrete nature, symbolic gestures are the ones best served by formal representations and the most commonly used in human-computer communications. In the simplest form, often used in gesture recognition literature for identification purposes [Glo12], a label on a symbolic gesture allows it to be taught and associated with the label and from this whatever meaning is assigned to the label is communicated. While effective for simple movements, more complex gestures increasingly rely on a common existing reference and label system (such as "A-Ok – the

common ‘okay’ gesture” or “Cutthroat – the common taunting gesture”).

A common alternative to relying on common labels but equally as simple in gesture recognition is to offer a photograph or drawing indicating the gesture in question, as in Fang *et al.* [Fan07] for hand pose and Kratz and Ballagas [KB09] for movement. Indeed, the latter only describes the gestures recognized in a visual form, while the former is more traditional in that it describes gestures with common labels and accompanies them with example images.

Functional movements can become symbolic gestures, as seen in Marking menu systems. As exemplified in the Flower menu [Bai08] variation of Marking menus, a radial menu with fixed item arrangement is navigated with a stylus or finger. A traditional Marking menu uses linear movements to navigate a menu structure and select a command while the Flower menu includes distinguishable curves to allow a broader menu tree, but in either case the process results in the shape of the path needed to get to a particular item being practiced until the menu no longer needs to appear on the screen as a guide. At this point a symbolic gesture is, for all intents and purposes, created. This kind of “half-gesture” approach can also be seen in the Shark2 typing system [KZ04], an approach also popularly used in modern smartphones under the label “Swype” where the letters of each word on a keypad connect to form a diagram and the end of the word is signaled with the lifting of the pen or stylus. As demonstrated in the Shark2 study, with experience the keyboard is no longer necessary to input that word.

3.2.2 Relational Gestures

The class of ’relational’ gestures contains a much more variable grouping than symbolic gestures; this is because part of the message embedded in a relational gesture is the magnitude of some variable. This variable can be directly spatial, such as position, orientation, scale or velocity, or it can be a spatial representation of a non-spatial variable, such as temporal duration or relative occurrence or decibels of sound.

While the magnitude of any metric can be communicated through relational gestures, almost

all such gestures are spatial-temporal in nature. One benefit of this relationship is the natural usage of affordances learned very early in life through the manipulation of physical objects or directions in human-to-human communications.

In fact, while symbolic gestures are more popular in the gesture recognition literature, relational and manipulative gestures are popular in gesture-control systems and virtual reality as implemented for common use. Virtual objects are pointed to, scaled and rotated, while more abstract actions are tied to the interaction with these objects.

Relational gestural representations, being less fixed and rigid, are less commonly represented and representations like Etchler *et al.*'s Gesture Definition Language [Ech10] rely often on constraint relationships with environmental objects or spaces. While Pantomime could have an indicator for variable movement and thus act act as a representation for relational gestures, it is not particularly suited for the task.

3.2.3 Conversational Gestures

Conversational gestures are the most informal and context-sensitive of the three categories. Whereas symbolic gestures communicate a fixed idea and relational gestures communicate a quantity, conversational gestures communicate emphasis, subjects, conversational rhythm or emotional state. They encompass both deictics and beats in McNeill's classification, and gesticulation in Kendon's. While there is research into interpreting conversational gestures and emotional state [Wex95], the impromptu nature of the movements and their dependence upon other modalities such as speech make representing them of little use.

For example, in Richard Bolt's "Put-That-There" interface [Bol80], finger pointing is used along with speech to indicate either an empty location to 'create' a virtual object, an existing object to be moved or changed or a destination for an object to be moved to. In this case, the meaning of the pointing is heavily contextual on the recognized word spoken while the gesture occurs.

Quek's FingerMouse [Que96], which uses a pointed finger as a three-dimensional cursor, similarly

recognizes both a hand pose (the pointed finger) and gross hand movement. However, as the target location or object is unimportant, this is not a deictic gesture system so much as a symbolic hand pose triggering a (virtual) manipulative movement.

Modeling conversational gestures in a simple representation such as Pantomime would be difficult due to several factors. The idiosyncratic nature of the gestures and the frequent pairing with speech to provide meaning and context make task modeling of gesture alone unsuitable; moreover, while symbolic and relational gestures tend to involve only the hands and arms, conversational gestures, as noted in Wexelblat's analysis, often will include a wider range of body parts and include actions such as rolling shoulders and tilting heads, which involve changing the shape of a body part rather than gross movement. Finally, the movement component of conversational gestures is often subservient to the pose component, meaning that movements will often be simple and functional.

3.3 The Utility of Representations

This chapter offers a simple representation of movement-based gesture. But what distinguishes a good representation from a poor one? A formalized representation, whether a taxonomy or a language, whether a calculus or a grammar, services a wide range of uses. Which usages are the focus of a particular representation vary—for example, the well-known Linnaean taxonomy of animals is ideal for classification, but offers very little capability to compare species and "calculation" isn't even relevant to the domain. A formal grammar is specialized for production of strings that exist a formal language, but does not lend itself well to communication.

This section describes four types of usage that are valuable for modeling—communication, classification, comparison and calculation—and outlines the importance of each for a representation of gesture. Of these four categories, the most important for an engineering model is perhaps calculation; comparison can either be done through the results of calculation or without for non-quantified contexts. Communication and classification are useful for design purposes, but are less important than the other two.

3.3.1 Communication

Much of gesture's use *in* communication is as an adjunct for verbal communication (often to indicate emphasis, relative quantities or a spatial relationship or location), when verbal communication is difficult or impossible due to distance or other constraints or when attempting to attract attention to oneself or some other location. It is an intrinsic part of human communication, developing in early childhood alongside verbal communication skills, but is not typically used as a replacement for them. These adjunctive gestures are currently outside the scope of the proposed representation due to their undefined and individualistic nature, but could be brought in with the right definition.

3.3.1.1 Communicating Gestures

Gestures are difficult to communicate without demonstration; in research literature, gestures are usually communicated in one of three ways. The first is by using pre-existing labels—for example, the set of roman numerals or English alphabet betters [Alo09; Wil09; Yoo01]. By relying upon the pre-existing knowledge of the readers, there is no need for illustrations or descriptions.

The second way of communicating gestures in the literature is to draw the gesture in a diagram or table. In some cases this is because the shape of the gesture is actually determined dynamically, as is seen in forms of marking menus such as Flower Menus [Bai08] or FlowMenus [GW00]; that is, each gesture is built by the need to touch a sequence of targets in a relative relationship. Once the menu is removed after the gesture is "trained", the shape alone is sufficient to trigger the desired command; while Fitts Law could be used to to model the movement with little difficult before that point, there's uncertainly how close the user will adhere to the scale and target locations without visual reference over time.

In others its because the movement focus is a wand or other hand-held tool [CB03]. The gestures might be performed in relation to an object, such as the Abracadabra system that uses polar gestures around a watch [HH09]. Most such papers have no particular reason mentioned or implied [CB05; Kal03; Kel06; Oka02]. In a similar circumstance, a photograph of someone performing the gesture

with the previous movement traced over is sometimes used to show an example from the gesture set [Alo09; Kim07; Suk10], particularly if the gestures contain pose and movement elements or are difficult to describe [Cha07; Dha06].

These two methods can be combined as well, with a drawing or pictograph accompanying a common label [Que96; ST07; Wob07].

Several papers never communicate the specifics of the gestures tested for a recognition system; the specifics of the gestures themselves were irrelevant. This happens particularly in cases where the gestures are an existing set such as a sign language or the set of gestures used is so large that listing is unfeasible [Gao04] and the specifics of the gestures are not relevant to the work.

Occasionally a paper will attempt to describe the gesture linguistically. These labels-with-description, such as seen in Ramamoorthy *et al.*'s "Recognition of Dynamic Hand Gestures" [Ram03] are usually enough for replication, but can be long and awkward to communicate.

3.3.1.2 Charade, GISpL and QualGest

Baudel's "Charade" system [BBL93] uses a three-stage pictograph to represent a gesture. The first is an abstraction of the hand pose, with a line indicating the palm and more lines or dots branching away from it indicating the thumb and fingers in either an open or closed position; all lines are rotated to show hand rotation. The second stage indicates the direction of movement; one gesture from the displayed set had a two-vector movement and the rest had only one. The third stage is, like the first, a 'pose' stage. The second stage has some additional symbols for repetition. This representation is intuitive and covers pose and movement. However, strip away the first and third stage (focusing on the movement) and it's little different from the arrows seen in gesture recognition work elsewhere. Moreover, it does not translate well to a computer's understanding, as it lacks a string form.

Echtler, Klinker and Butz [Ech10] developed a "language of gesture", later refined into GISpL [EB12] aimed at simplifying gesture recognition code. Their goal was an extensible system that would give

rise to a generic recognition engine that could be easily customized to allow rapid development of gesture-controlled applications. Their system has three parts—*region*, *gesture* and *feature*. The region is the location in which one or more gestures are eligible, the gesture is the activity and features are atomic (and recognizable) constraint-based properties of a gesture. Features can range from motion or rotation characteristics of an object (such as a hand, pen or block) in a region to the identity or count of an object type (such as fingers). This representation is certainly complete across gesture movement (and more), but for communication utility its specialized to the computer; while a human being can understand a description in Echtler's representation and it has a string form, the length and level of granularity are undesirable for concise communication.

Gibet and Lebourque [Gib01] developed a representation of communicative gestures, QualGest, for the purpose of computer reproduction (as opposed to GISpL or Charade, which were meant for recognition). Emphasis was made on the need to create new movement 'pieces' to build more complex movements and on the natural appearance of the animation. The result is meant to out-compete approaches such as inverse kinematics + key frames or theories of control and seeks to do so by the accumulation of 'building blocks' that can be turned into complex communicative movements. The source of the gestures used (and the immediate context of use) was French Sign Language—the broad range of types of movement and the existing research and grammatical infrastructure making it appealing to the developers of QualGest. Gestures are decomposed into five parameters—hand shape, hand orientation, arm and hand movement, location and facial expression. The goal of a complete representative system is similar to that of the later GISpL, but the addition of facial expression is of note. The communication of each gesture is done with a string rather than the pictographs of Charade. Each parameter can be described and the concatenation of those descriptions can be considered a communication of a whole gesture.

3.3.1.3 Needs of Communication

Being able to produce a gesture from a representation and produce the representation from a gesture is key—thus a good representation of gestures should be easy to understand by human beings and by computers. Simple and short is ideal, similar to the KLM or Charade; while more complex representations like GISpL or QualGest grant power and precision useful for gesture recognition or gesture animation across multiple domains of gesture, the goal is modeling and prediction and the domain is movement-based gestures alone. A good representation, as it should be able to be processed by a computer, needs to be compact, perhaps string-based. While a "display form" that interprets the representation in pictographs (as in Charade) is useful for human communication, its not relevant at this point.

3.3.2 Classification

Beyond the communication needs fulfilled by language, representations bring the ability to classify. Taxonomies, in fact, do almost nothing but classify, while underlying languages and calculi are systems of classification. Classification is a way to organize a subject, to make sense of similarities and differences across a category. The user of a classification system can make judgments on novel instances using their similarity to already-known instances.

A task modeling system like the KLM [Car80] takes the broad space of human task performance in a domain and splits it into smaller pieces. Those smaller pieces are grouped under categories and called "operators"; these atomic actions, such as pressing a key, moving a mouse or engaging in the perception-reaction loop, are defined and can be used as building blocks for future human task performance. If there's some activity that isn't covered by the existing operators, either because of the introduction of a new input device or because of the extension of the domain, then a new operator can be defined and added to the existing set. The decomposition of task performance and the treatment of each piece (the operators) can be thought of as an example of a 'path'-based approach to classification, where the path of an action is split into pieces which are then classified

and information rises from the characteristics of the pieces' categories.

A classification system can be mapped to a grouping of some kind (whether a similar classification system or not). Vatavu [VZ13] uses a developed taxonomy of hand pose to make a determination of the shape of what the hand is holding without visual inspection of the object itself. A classification of six 'grasping' hand postures (cyclindrical, spherical, tip, hook, palmar and lateral) was used for their experimental design, but the focus was on a fourteen feature definition of posture, with each feature being the measurement of the flexture or abduction distance measured by sensors on a worn glove. Data was used to train a nearest-neighbor-based matching algorithm on the size and shape of grasped objects. Vatavu's classification system can be considered to be using a 'feature'-based approach, where certain features (in this can the measurement of joints) of hand posture can be interpreted to mean that the hand is holding an object of a derived shape.

3.3.2.1 Classifying Gestures

The most straightforward way to classify gestures is by the path of movement. Gestures consisting of only linear movements in sequence could be grouped by the number of such movements; non-linear movements, with no obvious end-points would need to be segmented according to a set of rules, however. Gestures could also be classified by patterns or features—a gesture could be classified as 'closed', for example, if the gesture is complete at the same spatial point it started, and 'open' if the beginning and end are different spatially. The usage of straight lines, curves or both could be separate categories, as could the existence of patterns such as 'intersection' (the path crosses the same space at least once), or radial or linear symmetry.

Baudel's "Charade" classifies movement by direction with arrows, and has room to indicate either a repeat of the movement or a second movement; the examples shown show no non-linear movements such as circles. The beginning and ending pose pictographs indicate the status of each finger (extended or retracted) and the orientation of the hand. There was no displayed way of determining an open or closed state or other features of a movement suggested, perhaps because

Charade is aimed toward human reproduction rather than analysis, but gestures in Charade can be grouped by the basics of position or movement reasonably well.

Gestures could even be modeled 'from the top down' with constraints, allowing flexibility in performance such as a circle drawn clockwise being the same as one drawn counter-clockwise. The versatility of constraints allows for a very large gesture-set with precise definitions separating them from one another or a very small gesture-set that allows a variety of ways to trigger a given command.

It also fits in well with a 'feature'-based gesture recognition system, such as GISpL. GISpL's broad reach covers the location of a gesture and the nature of the gesture's focus. Gestures themselves are modeled as sets of features, with each feature being in effect a constraint—any sequence of captured frames that fit all of the constraints for a given gesture is recognized as that gesture. The eleven classes of features, along with the filters and constraints for each, are what gives GISpL its power; movement is only one of these, and it's joined by rotation and scale, the presence and activity of hands or fingers, the presence or usage of specific objects (whether used as a tool as in CAPTIVE [Cha14] or held as in Vatavu's shape recognition system [VZ13]), the dimensions of objects in the region and elements of the movement such as instantaneous velocity of objects. There is gesture-level classification using 'flags' such as "sticky" or "oneshot" as well.

QualGest uses a path-based approach like Charade, but has a larger number of more complex pieces and classes. Each of the five parameters (hand shape, hand orientation, hand/arm movement, location and facial expression) requires classification. The usage of French Sign Language allowed a relatively small set of categories for each, however. Five basic hand configurations were identified (labeled "angle", "hook", "spread", "fist" and "straight"), with fingers able to be identified separately and their shape as departs from the base configuration defined (using "spread/clenched", "angle/hook/round", "contact" and "crossing" with another digit). Orientation is classified by the direction of the palm and of the metacarpal. Gesture movement is flexible, with a gesture "sentence" being one or more gestures and each gesture composed of a sequence of elementary movements

(the movement primitives supplied are “point”, “line”, “curve”, “circle” and “wave”)—reading much like a list of instructions to perform (which it is).

3.3.2.2 Needs of Classification

The two forms of classification of gesture mentioned earlier, “by path” and “by feature” could both be included in some form in Pantomime. However, each lends itself to a different approach, and a path-based classification should be more in line with existing task modeling—KLM, for example, functions by breaking down a task into sequential operators, and Isokoski’s straight-line reduction and Cao & Zhai’s “Corners, Lines and Curves” models both rely upon the shape of the drawn gesture. Thus the representation developed later in the chapter models itself more like the KLM than Vatavu’s feature-based taxonomy of hand posture, with feature recognition lying either in an addition to the base to clarify possible ambiguity or existing as an abstracted form of the base to represent patterns.

3.3.3 Comparison

The cousin to classification is comparison. While classification is about placing items into categories defined by a representation and defining relationships between those categories (or between categories and equations, constants or categories in other classification systems), comparison is about determining the relationship between items *using* features of the representation (a form of qualitative analysis). In particular for task modeling, comparison is most often used to determine which of two courses of action is superior for a given situation (a mapping or an interface, for example).

There are often multiple methods to achieve a goal in a system—for example, in most graphical file management programs there are multiple ways to delete a file using the keyboard, the mouse or a combination of both. Multiple versions of each device might be compared to determine which shape of keyboard or mouse is optimal for use [Mac91]. Finally, as in the case of an open input method such as gestures, the mapping of gesture to action could make a significant difference in

task performance. A good comparative representation should allow snap decisions and easy and quick optimization either on its own or, given constants and calculative elements, on a metric such as duration, difficulty or efficiency.

Task models "in the world" are often applied in practice as comparators. Haimson and Grossman [HG09], for example, use GOMSL (Goals, Operators, Methods and Selection Rules Language) to test a content extraction-assisted data entry method ("SysA") against a pure manual approach ("SysM"); they are able to quantify the expected difference in time between using SysA and SysM under the different possible circumstances (SysA's choices being accurate as-is, along with a range of different cases in which correction was needed). In the previous chapter, several cases in which the Keystroke-Level Model was used to choose between alternate implementations of hardware (such as the Xerox Mouse) or software (such as a ported CAD program) or even user habits (Such as the Bell South telephone operator protocol) were mentioned [JK96b]. A task model often reduces a usage scenario to a single value, "time to do this sequence of actions"; a representation of gestures should be able to fit right in to this kind of usage given the appropriate constants and equations.

Card, English and Burr [Car78] use the Fitts' Law task model to compare four different targeted devices against each other and against a hypothetical minimum performance established in other literature. A mouse, a joystick, and two arrow-key layouts were all tested in a classic targeting task and the data regressed to the Shannon Formulation to give two comparable quantities for each device—the reaction time and the bits per second as per information theory. This comparison was able to demonstrate a ranking of device performance (mouse, joystick, text keys and step keys) quite handily. Moreover, due to the fitting of each trial during the learning period, the authors were able to compare the learning speed for each device (with step keys lagging behind the other three methods). Gestures or gestural modes (such as one-handed, two-handed and while holding a device) could be similarly compared either within a representation of gesturing or around it.

3.3.3.1 Comparing Gestures

There is little comparison between gestures in the literature; gesture average performance values are reported [Wob07; Hol07], but there is no distinction between gestures inside the sub-domain reported on.

Charade is not meant as a comparator, but as a communicator and (to a secondary degree) a classifier. Different poses can be compared for similarity, but there's no mechanism for determining whether one pose is a better choice than another in the representation itself. However, it *can* be used to compare a mapping of gestures across a usage scenario, with a goal of minimizing pose changes from one gesture to the next; that is, a mapping in which gestures that follow another have a starting pose the same as the ending pose of the previous, which could allow easier chaining of actions. Conversely, actions that do not typically follow one another could have the inverse to prevent accidental triggering of actions. The movement segment allows some fast comparison between one-segment, two-segment and repeated movement-segment gestures. With an action frequency table, this might allow mapping frequent actions to the one-segment gestures and infrequent (or actions with a high accidental activation cost) to the two-segment gestures.

GISpL's constraint-based approach to features works poorly for comparison of gestures. With each gesture composed of one or more features and each feature being rooted in a set of parameters, there's no easy way to take two gestures and determine which is better for a given mapping. Gestures could be quickly ranked based on the number of features, and features perhaps on the complexity of their parameters, but there's no guarantee that fewer features would mean a gesture is easier to perform—only that they are easier to define for the recognition system. If accompanied with data on the desirability of certain regions for performance or of certain features in a context, than a comparison of desirability could be made—but that data would likely need to be individual to the interface regardless. It's clear that a constraint-based approach is unlikely to satisfy the needs of comparison.

A by-path approach is not guaranteed to allow for easy comparison either. QualGest, aimed at

gesture reproduction, lends itself only a little more to comparison tasks than GISpL. Gestures that keep a static hand pose could be considered simpler than those that change their hand pose. The different elemental movements could be ranked, and the number of movements changed together in a specific gesture might be counted. However, the targeted domain (sign language) tends to have very few such movements, meaning that comparison is likely be quite 'flat'. QualGest would need an accompanying routine that analyzes a gesture based on one or more fitness qualities to do comparison—and as it's meant for computerized reproduction, it doesn't have much need for one.

3.3.3.2 The Subgesture Problem

The subgesture problem, as noted in the gesture recognition literature [Alo09], is when the elements of a gesture are contained within the elements of a second gesture. While of less concern for a representation that is aimed at acting as a performance-comparing task model, having a subgesture situation recognizable is an asset for broader design decision-making.

A subgesture can lead to ambiguity in human-machine communication in several circumstances. If the enclosing gesture starts with the overlapping elements, then the recognition system may either trigger immediately when the enclosed gesture is registered as completed or if the rest of the enclosing gesture is not recognized. An enclosing gesture's pre-overlap elements being unrecognized would likewise cause the wrong gesture to be triggered. If either 'noise' movement or the end or start of chained gestures turn the enclosed gesture into the enclosing gesture, there will likewise be an error. Of the three representations of gesture described in this section, GISpL could allow a recognition system to detect the subgesture problem by testing all possible gestures for activation (as opposed to testing until one activates and then stopping). Charade could reveal a subgesture problem through visual inspection of the pictographs; QualGest could function similarly, but as its functionality moves the other way, a subgesture situation is of less concern.

3.3.3.3 Needs of Comparison

Comparison tasks often rely upon a representation's ability to perform classification or calculation tasks; thus a representation that doesn't lend itself to either will likely be a poor comparator for gestures. With the ability to classify, a representation may be able to overcome the subgesture problem; with the ability to calculate using a performance value (such as task duration), a representation may be able to make comparisons outside the representation itself (for example, being compared against the use of a keyboard).

A comparator needs to have an organization that has elements or features that can be considered superior to others. Given a continuous gesture, some manner of breakdown or *decomposition* is necessary, particularly given the need to compare the whole of one gesture to a part of another in order to deal with the subgesture problem. Both by-path and by-feature, the classification criteria discussed earlier, allow some form of decomposition. A gesture's shape can be decomposed as shown by both Isokoski's and Cao & Zhai's work, while a list of features can be considered a form of 'decomposition' that theoretically allows some form of comparison. However, while two gestures' feature lists would allow some recording of similarity through the presence of shared features, more layers of interpretation (such as some form of feature hierarchy or between-feature similarity table) would be needed for a more reliable judgment. As Pantomime is designed to be a simple model, the more straightforward "by path" approach to decomposition will be used.

3.3.4 Calculation

The last of the utility categories discussed for a representation of gestures is that of calculation—taking a representation and some constants and answering quantitative questions about the specific represented phenomenon. In the context of task models, frequently calculation leads to the ability to compare alternative ways of achieving a task goal. Classification of the phenomenon or of pieces of the phenomenon, when combined with carefully-designed experiments, often allows the assignment of constants to categories and from there calculation of performance values.

In literature surrounding gestures, most research is focused upon gesture recognition rather than evaluation; thus there is often a gap in the understanding that might have allowed models to evaluate gesture-based (or multi-modal) systems properly. And while non-qualitative comparison allows for design decisions for optimization, with constants and a way of calculating values additional questions can be answered. For example, knowing the amount of time it will take for input to be performed gives a system designer an understanding of the necessary responsiveness of a system—ie, how much time can be spent before the system needs to be able to recognize input again. Comparison outside of a given representation is difficult without a common value to compare upon—the duration of using a gesture-based interface can be compared to the duration of using a keyboard-based interface, for example.

But calculation in the context of gesture representation goes beyond numeric values indicating duration, effort or other metrics. It also involves logical relationships between gestures or between different pieces of a gesture. This can be particularly important when it comes to the spatial or temporal arrangement of gestures or the relationship between the movement of two hands in a two-handed gesture.

3.3.4.1 Allen's Temporal Calculus

Allen's Temporal Calculus [All83], while not a formalization specifically for gesturing, demonstrates this utility. By using a framework of logical operators, Allen is able to combine fragmentary temporal relationships between multiple parties to fill out the entirety of a timeline in a constraint-centered fashion. While simple in concept, it allows computers the temporal reasoning developed by human beings through experience; "if A happens before B, and C happens before A, then C must have happened before B".

Attempts to convert this directly to a spatial calculus have met with limited success, but the calculus could be used in the proposed gesture representation to define temporal and spatial relationships of of multiple foci in their movements (what are later referred to as "tracks"). This is

a second 'level' of gesture representation beyond the basic single-handed movements that were tested in the experiment, however.

3.3.4.2 Calculation of Gestures

The original Keystroke-Level Model [Car80] is an example of a representation designed principally for calculation, particularly predictive performance calculation. Tasks are broken down into a sequence of four physical operators (Keystroke, Pointing, Homing and Drawing), three of which have a performance constant and one, Drawing, which has a performance equation. There's a single Mental operator to cover preparation or thinking which is inserted according to a pair of rules and then removed according to another set of rules. There's a sixth operator, Response, in the case of there being a wait for a computer system to perform a task.

In the KLM, Drawing is the operator most akin to gesturing. It was meant to predict the performance of drawing a sequence of straight lines with the mouse on a defined grid of a particular size (about half a centimeter). The calculation for the performance duration used both the number of segments in the drawing and the total length of all segments, each weighted by a constant. Holleis [Hol07], in an adaptation of the KLM to cell phone usage, develops several additional operators, such as moving the phone, attention-shifts and distraction levels, and one additional operator; the Gesture operator. However, instead of taking inspiration from the KLM's Drawing operator or pulling from task models such as Fitts, the G operator is given a constant value of 0.8 seconds.

Charade has no strictly calculative component, being a purely descriptive representation for communication and recognition. However, given performance constants for forming hand poses, the transitions between hand poses and the movement stage, performance duration predictions could be made using Charade. For the movement stage, in fact, a representation of movement-based gestures such as the one as described later in this chapter, paired with the experimental constants and calculations discussed in the next, could be used for this purpose as a kind of "sub-representation".

GISpL, with its more complex constraint-based approach to features and the gestures they make

up, would need a similarly-complex calculative adjunct to perform calculative tasks. One feature, movement, could map to the proposed representation in a similar manner to Charade, but that leaves a number of other features, along with how features might affect the performance of other features. For example, in task modeling literature there's suggestion of a mild 'scaling' impact on performance [CZ07]; perhaps the scaling feature in GISpL might have a similar impact on gesture performance prediction. The usage of tools or other objects of interest may require including the time to acquire the object and bring it to a region. Even the region of performance may have a performance impact. To act as a performance predictor, GISpL would need a whole new element overlaid over the recognition architecture.

3.3.4.3 Needs of Calculation

The end-goal of this representation of basic movement-based gestures is to give a nuanced and accurate prediction of human performance, not just in isolation, but as part of the more general modeling frameworks currently used, such as the KLM, GOMS or ACT-R. As such, of these four categories, calculation is the most important. Communication utility is desirable, but written forms of sign language such as Stokoe's notation and more domain-specific representations such as Charade already demonstrate a blueprint for gesture communication that is intuitive and effective. Classification and comparison utility are useful for addressing design process needs such as the subgesture problem and gesture recognition algorithms, but comparison for the purpose of optimal mapping can arise from the results of calculation. Moreover, it is in the calculation category that existing representations of gestures are most lacking. Thus, determining the factors that lead to good calculative utility are a primary concern.

The end result of Pantomime must be a quantitative predictive task model—this is itself a constraint. Moreover, the form of the prediction must be compatible with existing task models such as the Keystroke-Level Model; it must predict gesture duration. The predictive model itself should be in the form of an equation that, like Fitts' Law and other task models, contains experimentally-

determined constants that can be fitted to data.

3.4 Building Pantomime

The examination of task models in Chapter 2 and of representations in this chapter lead to a list of desired qualities for the prospective representation and task model of gestures, entitled "Pantomime". These include:

- *It must lead to* a priori *predictions of gesture performance, based only on the shape of the gesture and understanding of the interface and not on factors such as gesture performer or the length of the movement.* (the goal of the research)
- *It must be able to be applied to any symbolic movement-based gesture, although the representation does not need to be unique.* (coverage of domain)
- *It should have a simple string-based encoding compatible with those of existing task models such as the KLM.* (communication)
- *It should have a "by path" approach to defining a gesture.* (classification)
- *It should allow comparison of two gestures, perhaps through breaking them down and comparing the pieces.* (comparison)
- *It should allow comparison of part of a gesture and another gesture, in order to detect subgestures.* (classification and comparison)
- *It should lead to predictions of gesture duration, so as to be compatible with existing task models.* (calculation)
- *It should be simple enough to be rapidly applied at little cost either manually or through an automated process.* (usable by practitioners)

These guidelines and existing task models of other domains of gesturing will guide the formulation of the Pantomime representation.

3.4.1 Decomposition and Discretization

A gesture's path is a continuous movement, but it can be decomposed into pieces. The Drawing operator of the KLM takes a diagram drawn on a grid and breaks it down into segments; from those segments it produces the equation $.9n_D + .16l_D$, where n_D is the number of straight-line segments and l_D is the total length of the whole drawing. The Keystroke-Level Model as a whole takes a task and breaks it down into the set of operators. Cao & Zhai's "Corners, Lines and Curves" [CZ07] similarly decomposes a drawn gesture into the three types of components specified.

Existing gesture recognition literature [MA07] suggest the use of temporal segmentation for the continuous movement of dynamic gestures; with both modeling and gesture recognition precedent supporting the approach, Pantomime decomposes gestures according to their path into components and presents the sequence of these components as that gesture's representation. In part following Cao & Zhai, the component types will be 'lines' and 'curves'; while transitions or 'corners' have a performance cost, Cao & Zhai found that cost to be quite small, and is more reflected in the impact of the transition upon the neighboring segments.

However, a curve has no natural division point as the beginning and end of a line segment does; Pantomime must either define a large range of different curved components, use a flexible equation that can be adapted to the specifics of a given performance…or discretize the curve. Isokoski's straight-line reduction [Iso01] of Unistroke input discretizes a curve into one or more straight lines, making any gesture into one that can be modeled rapidly and *a priori*, albeit with an uncertain accuracy (with a coefficient of determination noted by Cao & Zhai as ranging between .5 and .85), while Cao & Zhai use Viviani's power law of curvature to calculate the performance of an undiscretized arc of any sweep angle from that angle and the radius (With a complex curve, CLC still requires some discretization, as the power law of curvature acts only on an arc).

Both of these approaches are undesirable. The straight-line reduction for Unistroke input requires judgment as to how many lines to use—for example, should a circle be represented by three lines? Four? Or more? Moreover, the irregular fit suggests that the result may not achieve the 80% accuracy benchmark desired for a predictive task model. Corners, Lines and Curves still must discretize complex curves and requires the radius of the movement—which would make Pantomime no longer *a priori*. Moreover, the equation is perhaps more precise than is necessary - with the variance in human performance of gestures, the precise sweep arc may vary from performance to performance anyway.

Pantomime both decomposes and discretizes gestures, much like the straight-line reduction of Unistroke writing, but curves are cut into approximated fixed-sweep angle segments based on the granularity of the model.

3.4.2 Granularity and Orientation

With three component types (lines, clockwise curves and counterclockwise curves) and curves discretized at some level, the level of granularity of the representation needs to be determined. After all, a straight line vector can exist at any degree angle, leading to hundreds of possible straight line orientations alone; at least some abstraction into fewer number of orientations is necessary.

And, in fact, there is already fairly coarse granularity in human spatial communication and demonstration; we describe movement in 90-degree intervals easily enough, and the 45-degree diagonals almost as well, but rarely more precise without an explicit external guide. Few symbolic movement-based gestures will require angle differences somewhere between 45°and 90°.

The result is a “compass-rose” like arrangement, with eight possible straight-line components. Curves are treated similarly; the difference in actual human performance between a straight line and that same line ‘bent’ into a 45°arc is small, as is between the bent line and one bent further to 90°. Thus each curved component is a 90°arc, and with two directions of curvature, that leaves Pantomime with a total of 24 possible components.

3.4.3 Components and Sequences

In the representation, each component is composed of a number and a letter. The number (0-8) indicates the orientation of the component's vector on the compass-rose, with '1' being "northeast" and numbers increasing clockwise until '8', which is "north". The number 0 is used to represent no movement. The letter indicates the type of component; at present, there are only three - (s)traight line, 90°(c)lockwise curve and 90°countercloc(k)wise curve.

These represent components in two dimensions; adding a third dimension is generally not needed for the current generation of gestures, as they've been developed with vision in mind; "towards" and "away" are visually difficult to measure. In a truly three-dimensional gesturing environment, an alternative approach of "<x><y><z><component type>" is possible, with each of the axis letters replaced by '+', '-' or '=' for positive, negative and neutral movement in that axis.

As components represent sequential pieces of the whole gesture, the set of all components of a gesture arranged in chronological order is called the "component sequence". For example, a clockwise circle starting from the top would have a component sequence of "3c5c7c1c", while a downward-pointing triangle starting from the left point would have a component sequence of "2s5s7s". This is the essence of a Pantomime representation of a gesture.

3.4.4 Reduction into a Task Model

Transforming the component sequence into a simple task model, the component count model, follows the example of the Keystroke-Level Model. In the KLM, $T_{execute} = T_K + T_P + T_H + T_D + T_M + T_R$, where the total performance of each operator is calculated similarly to $T_K = n_K t_K$ (or the total keypress time is the number of keypresses times the value per keypress). For Pantomime, with lines and curves (and both types of curves presumed to be equivalent), the result is Eq. 3.1, with MT being the movement time, a and b being experimentally determined constants, l being the number of linear components in the gesture and c being the number of curved components in the gesture.

$$MT = (a\,l + b\,c)\ \text{milliseconds} \tag{3.1}$$

For the purposes of the experiments in the next chapter, a second task model, the stroke count model (Eq. 3.2) following in the lines of Isokoski's straight-line reduction can also be derived from the component sequence, where a is an experimentally determined constant and s is the total number of components, whether linear or curved.

$$MT = a\,s\ \text{milliseconds} \tag{3.2}$$

Testing both of these models will determine whether the two types of component have different performance values; that is, if users perform lines or curves faster or slower—and thus if a gesture using curves would be faster or slower than one using a similar number of lines.

3.5 Expansions to Pantomime

The purpose of the Pantomime representation is to produce the grounding and justification for the task models in Eq. 3.1 and Eq. 3.2. However, there are additional components that could be added to the representation to extend its usefulness beyond the current boundaries. These include a supplemental feature list to capture more of a gesture's unique characteristics, adding the capability to represent two-handed gestures, allowing variability for representing less rigid gestures or relational gestures and matching it with a similar representation of hand pose to complete the two halves of the gesture-space.

3.5.1 Feature List

Pantomime is a by-path approach to gesture representation. However, the alternative approach, by-feature, exhibited by GISpL, offers additional utility to a representation of gestures both inside and outside the goal of modeling performance.

The fundamental claim of the component count (and the stroke count) model is that the performance of a gesture depends entirely upon the performance of these discrete pieces that a gesture has been decomposed into—or, at least, sufficient to result in a reliable prediction of gesture performance across the majority of possible gestures. But what if there are gestures that do *not* fit this claim? With a feature list, Pantomime could still be used to determine which gestures display that abnormal performance and then determine the common features that may have led to a shorter- or longer-than-expected gesture duration.

Moreover, there are uses for a representation of gesture outside of the creation of task models. The three representations of gesture discussed earlier in this chapter, Charade, GISpL and QualGest, are designed to be used either to recognize gestures or to reproduce them with an animated figure. But to be able to accomplish either of these things, Pantomime's representation must distinguish between gestures that have the same component sequence but are visually distinct.

For example, a circle and a spiral can have the same signature, "3c5c7c1c", but one is a closed gesture (with the ending point approximately at the starting point) and the other is open. Even two spirals may be visually different as one moves inward and the other moves outward. Another example is the difference between a three-line shape in which the third line crosses over the first (such as a "crossed out" gesture) and one in which the third line *stops* at where the first line crossed, and either of them with a similarly-oriented triangle.

A simple way to add a feature list is to enclose the component sequence, enclose a feature set and concatenate the two together, like so: "{3c5c7c1c}{o}" (a circle), "{3c5c7c1c}{w}" (a spiral outward), "{3c5c7c1c}{n}" (a spiral inward), "{5s8s3s}{1x3}" (crossed out gesture) and "{5s8s3s}{1t3}" (stopped crossed out gesture). The feature could be encoded as a single letter similar to component types, as was done here, or listed out by label.

3.5.2 Tracks and Multiple Hands

While the Pantomime task model is focused upon the performance of simple one-handed gestures, there is no reason that Pantomime couldn't be applied to two hands or even more foci, in the case of multiple users or a very unusual or contrived two-handed-and-one-footed gesture. Two-handed gestures tend to have fewer components than one-handed gestures and the movement of each hand is often linked—mirrored or copied, for example. Regardless, however, Pantomime would need a way to distinguish two different component sequences.

This could be important for a task model of all free-space movement-based gestures, as the relative performance of bi-manual gestures and single-handed gestures with the same component counts are unknown; Welford, for example [Wel60], states that "in two-handed work, the hands cannot be treated as independent units".

One way of representing specific foci is by attaching a label inside the brackets indicating a particular hand, like so: "{L:4s7s2s7s}". For two hands, the bracketed sequences could be separated by a slash: "{L:5s3s}/{R:3s5s}". Custom labels could be adhered for multi-user gesture commands or other unusual circumstances.

With the movement of multiple foci, however, comes the question of temporal and spatial relationship. While Allen's Temporal Calculus previously discussed offers a way to define any relationship between two events (such as the movement of the left hand and movement of the right), adaptation to a spatial calculus has not so far been entirely successful. Given an appropriate adaptation, or borrowing from other representations such as QualGest, Pantomime might be able to represent bi-manual gestures spatially or temporally arranged.

3.5.3 Patterns and Variables

In some cases, a defined gesture in an interface has an element of variability to it. For example, a action invoked by "a circle" may require one with a particular component sequence or it may trigger upon *any* circle. Or a gesture may have a relative component, such as pointing to some object. How

can Pantomime represent this kind of gesture?

To some extent, patterns can be recognized in the component sequence - for example, a circle will have four curved components, all the same direction, with numbers either increasing (clockwise) by 2 or decreasing (counterclockwise) by 2. A proposed way to mark that a gesture's components are relative to one another could be a quote after either the component type letter or the directional number, such as "{3'c5'c7'c9'c}", to indicate that the marked numbers or letters are relative to one another. Similarly a single line (such as moving to point) could be made variable by the use of a question mark, so a "pointing component" would be read as "?s"

3.5.4 Hand Pose

One final example is the combination of the movement-based representation of Pantomime with a pose-based representation with a similar format. While such a representation has not been developed as part of the process of creating Pantomime, a string format could be placed after the component sequence (and optionally a feature list) like so: "{7k5k3k1k}{o}{<hand pose>}. If the hand pose changes, then the pose representation could have a tag indicating the index at which movement component the change occurs.

3.6 Summary

In this chapter, a simple gesture representation and task model called Pantomime was developed. Drawing inspiration and guidance from the gesturing domain, existing representations of both other task models and of gestures and a set of categories of use, the result is easy to understand and straightforward to use as a task model. The process of decomposing and discretizing a gesture was discussed and the formation of the component sequence was explained. Finally, task models were developed from the component sequence itself. With a complete task model ready to be tested against experimental data, the chapter finished with a set of ways in which Pantomime can be expanded beyond the simple framework currently needed to develop the task models that are the

goal of this work.

Pantomime meets the list of requirements developed in this chapter. Specifically:

- The component sequence does not use any data from a performance; thus the component count model that is based entirely on it can be considered *a priori*.
- Any path of movement can be translated to a component sequence, thus fulfilling the need for coverage.
- The simple string-encoded component sequence records the path of a gesture and decomposes it into discrete pieces that are comparable.
- The component count, given experimentally-determined constants, allows for rapid prediction of any gesture's performance once the path is described.

Considering the usage categories broadly, Pantomime can be communicated as either a component sequence string or in the form of spatially-arranged pictographs. It allows crude classification of gestures by both a complexity measure (the stroke count) and by a gesture's curved or linear nature—additional classification would require a feature list or other expansions, however. Gestures can be compared by their stroke or component counts, and subgestures can be identified with a simple substring search. With the component count model and experimentally determined constants, Pantomime can be used to make quantitative predictions of gesture performance.

While the representation itself makes no judgment as to which components are faster or how long a component sequence will take to perform, a quick count of components can give a ballpark estimate of relative performance of two gestures or as a measure of raw complexity of a gesture (complexity in Isokoski's straight-line reduction [Iso01] is equivalent to the number of strokes, while Kieras' NGOMSL uses the number of instructions in a task as a measure of the complexity of the task [Kie94]).

With the addition of experimentally-determined constants to the task model, performance time of gestures can be calculated *a priori* and compared with either each other or with other forms of

input modeled by existing frameworks such a GOMS; the component count model could be used as-is as a KLM "Gesture" operator. The possibility of other physical performance metrics, such as energy cost or session fatigue, being model-able using the component sequence have not been examined in this work.

In theory, the component sequence can be applied to any movement component of a gesture. This includes a movement component of a sign language, the drawing of a gesture with a finger or a stylus on a touchscreen or even conversational gestures. However, for some forms of movement the result would be meaningless - for example, a shrugging conversational gesture could have a component sequence for the up-pause-down movement, but the value of having that sequence would be minimal. When movement is very short and constantly changing directions, such as wriggling a finger or rapid shaking of the hand, the component sequence similarly would have little use; moreover, such continuous movements are often repeated a variable number of times depending on circumstance. Thus the task model is only recommended for application to gestures with movement with scale beyond “wriggling fingers”.

CHAPTER

4

EXPERIMENTAL ANALYSIS

In this chapter, three experiments on gesture performance are discussed and used to inform the task model produced from Pantomime for use in a Keystroke-Level Model environment. The first study was performed using the Leap Motion, a small camera-based gesture recognition device, while the second and third were performed with the Microsoft Kinect, a larger-scale camera-based gesture recognition device with lower resolution. The differing scale of gesturing between the first study and the other two was meant to identify any changes in performance, while the third study used the same hardware as the second so as to validate the performance of the models informed by the second.

In these studies, three models are considered. The first is a zero-parameter model [1] that, like

[1] The term "zero-parameter model" is used in two distinct senses in the literature. For one sense, consider the prediction of the duration of a single gesture of a given type. A model that can be expressed as a function $D_{gesture}()$, with no parameters, is a zero-parameter model. A model that takes into account the type of the gesture $D_{gesture}(type)$ would be a one-parameter

the Homing operator of the Keystroke-Level Model [Car80] or the Gesture operator of Holleis et al's extension of KLM to mobile tasks [Hol07], predicts gesture performance with a constant value. The second is a single-parameter model based upon overall stroke-count, similar to the multi-keypress K(n) operator of the KLM. The third is a two-parameter model based on the separate number of lines and quarter-circle curves into which Pantomime discretizes the continuous path of a gesture.

These models are compared in order to determine the amount of improvement in prediction that a component count model could offer over a mean-value model. Furthermore, the difference in performance between types of components can be determined both directly from the data and from the amount of improvement in predictive accuracy from differentiating the components in a Pantomime representation of a gesture.

Two statistical measurements will be used to compare the models, the root mean square error and the prediction error. The root mean square error (RMSE) measures the magnitude of the difference between observed data and model prediction, and is in the same units as the prediction. The prediction error measures the percentage difference between the model prediction and the mean duration for each gesture, a standard metric in Fitts' Law and Keystroke-Level Model studies [LJ05].

The two measurements were chosen because they have appeared in existing task modeling experiments as indicated in Chapter 2. Moreover, each serves a different purpose; the RMSE is a measurement that indicates the ability of the given model to explain the variance in the data set (or, more precisely, indicates to what extent the given model *cannot* explain the variance). The RMSE for two models can be compared, but the amount of the RMSE additionally suggests the variability or 'noisiness' of the data. Meanwhile, the prediction error is used to measure the suitability of the final model against the accuracy goal and shows how reliable the model could be in application to task modeling scenarios.

model, and so forth. In the task analysis literature a different sense is used: a zero-parameter model of some process must be able to make a priori predictions, in the absence of data from that process [Joh88]. Fitts' Law is an example of a zero-parameter model in this sense, while a hypothetical model that predicted movement duration as a function of observed movement distance would not be. All of the models described in this chapter are intended to be zero-parameter, *a priori* models in the latter sense.

4.1 Leap Motion Study

A pilot study was conducted as both a practice run for experimental protocol and to gather some preliminary data to explore for the initial model-building. This study was relatively informal, with the experimental apparatus brought to a location and volunteers solicited. The task was also stripped down, in accordance with experiments such as those done for Fitts' Law that simplified out usage context to acquire the fundamental mechanics of the motion. The objective was to acquire data to test regression to equations, starting with simple linear and moving towards more complex until a form that offered acceptable accuracy was determined. A secondary objective was to test the analysis protocols to be used in later formal studies.

4.1.1 Methods

The task was to perform gestures when prompted while a camera-based device, the Leap Motion, tracks the gesturing hand. Visual feedback was displayed in the form of a wire-frame 3D cursor and trace. The gesturing hand was tracked and the raw coordinate data in three dimensions stored for processing.

4.1.1.1 Task

Participants were given six gestures to learn, and were asked to perform them in specified four-gesture sequences. Each iteration of the sequence was prompted with the names of the four gestures to perform and then a cue, with an accurate rehearsal followed by three recorded iterations. Simple visual feedback of what the camera perceived was offered, but no other computer action resulted from successful or failed completion of a gesture.

The gestures given to the participants were selected for familiarity and ease of description—a circle, a lowercase 'd' with a double line, a lower case 'h', a capital 'P', a letter 'V' and a slash from the top-right.

4.1.1.2 Apparatus

The Leap Motion contains a pair of cameras, facing up, that observe a cone of space above the thumb-sized device. The effective range of the cone is about 2.5 to 3 feet vertically, with the maximum diameter of the cone being about 2 feet. The precision of observations is high, with the individual joints of the fingers easily detectable and trackable at 60 frames per seconds. The sensory space requires gesturing at a relatively small level; large arm movements would take the tracked hand out of range.

The interface was displayed on a laptop screen with the Leap Motion positioned in front of the laptop keyboard. The hand was tracked on-screen with a ball moving inside a wireframe box, with several past positions represented by a trail from the ball.

4.1.1.3 Participants

For the study, eight participants were gathered. All participants were students at NCSU, although from no particular department or level of education. Nearly all had "little to no" previous experience with gesture-command systems. There was no remuneration; participation was voluntarily upon approach and request.

Given the small size and portable nature of the laptop and Leap Motion system, participants were asked to perform in the immediate location. Thus the location was uncontrolled, as was the possibility of external distractors. Participants were seated during the experiment.

The number of gestures in a set was restricted to six to avoid cognitive overload. As training is not extensive and cannot go beyond a single session, a gesture set larger than six was presumed to introduce extra hesitation and artificially slow performance.

4.1.2 Data

The Leap Motion, while in theory capable of video recording, was at the time limited by the API to the production of the joint locations and approximate hand shape. Data was recorded in the form of

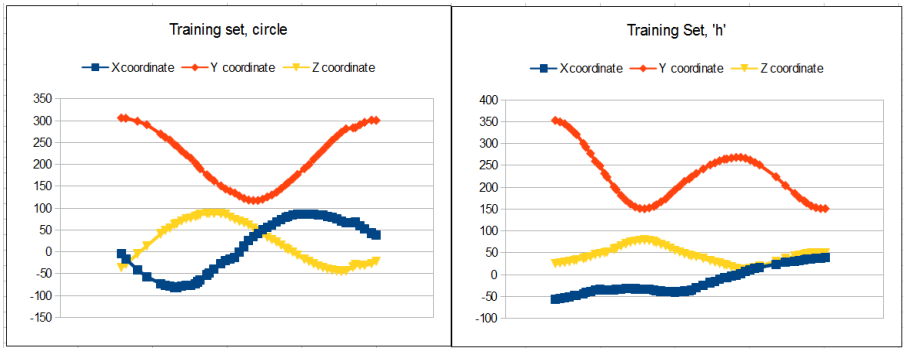

Figure 4.1 A scatter plot of two examples from the pilot data; “circle” and “lower h”

X-Y-Z coordinates (in millimeters), with Z discarded (all the gestures tested were two-dimensional, having no depth component), each having a timestamp in milliseconds (see Fig. 4.1), and turned into per-gesture records.

Each record was identified by participant identifier, order of the sequence in the experiment and the performance iteration (1, 2 or 3) of the sequence. The record also included the sequence start and end points, along with those of each gesture within it, the duration and identity of the gesture and the duration of the intervals between gestures of the same sequence.

The raw data was converted to the data records through the use of the scatter plot (as per Fig. 4.1), which allows reasonably reliable selection of boundaries of the gestures. The selection was cropped by an visual examination of the ’picture’ (the XY trace) of the selection. If the resulting gesture was formed correctly, the data points were classified as a performance and the duration of the gesture was calculated as the difference between the first and last data points’ timestamp.

Because visual inspection was used to crop performances, to lessen experimenter bias the duration of the prospective gesture was calculated after the gesture’s endpoints were finalized and the validity of the gesture was determined.

4.1.2.1 Data Records

Given eight participants, each performing twelve sequences of four gestures and each sequence being repeated three time, the maximum number of possible records was 1152. After sequences discarded for sensor errors, a total of 1112 gesture records (278 four-gesture logs) performed by eight participants were collected. Of those gestures, 1077 were accepted as accurate performances, giving a 3.14% error rate. The records included participant ID, gesture ID, start and end timestamps, gesture durations, as well as the total stroke, line and curve count for the identified gesture.

Each gesture record for the Leap Motion study contained the participant's random ID, the type of the gesture performed (deriving the line and curve counts and the stroke count) and start and end times (deriving the gesture duration).

4.1.2.2 Gesture Type

Splitting the records by gesture type shows different means for the gestures, as seen in Fig. 4.2. The gesture with the shortest component sequence (the 'slash') has the lowest mean, at 298ms, while gestures with longer sequences have higher means, such as the 'doubled-d' at 918ms, with linear components seeming to lead to a greater increase than curved components.

This suggests not only that a model that takes these differences in gestures into account would outperform a constant-time model, but that there is a difference in the performance time of the different type of components—both important assumptions of the component count model approach.

4.1.3 Models

The data, particularly the graph of duration by gesture type, suggests that a gesture characteristic-based task model will outperform a zero-parameter model. Thus the first step is to find the best-fitting constant-time model and examine the fitness of applying it as a prediction to the data. After, an intermediate-stage model, a similarly-optimal stroke count model will be produced along with the

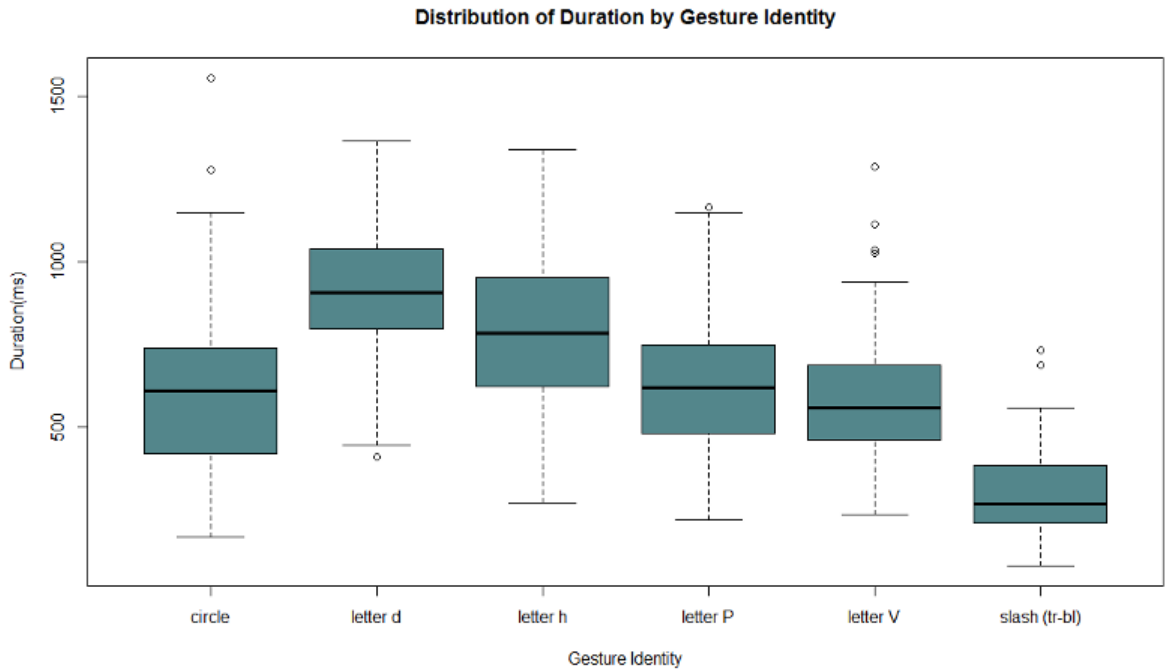

Figure 4.2 Performance by gesture type in the Leap Motion study

target component-based model to provide a zero-, one- and two-parameter model for comparison. The gesture set's characteristics are listed in Table 4.1.

4.1.3.1 The Mean-Value Model

The best-fitting zero-parameter model for a sample data set is the mean, which minimizes the least-squared error. For this data, the mean is 642ms. By convention, most task models work at a resolution of ten milliseconds, and so 642ms is rounded down to 640ms. In the task models of Holleis et al [Hol07], a mean value of 800ms is used for phone gesturing. Gestures can have widely varying performances based on the circumstances or particular gestures. Cao and Zhai [CZ07] have a mean of 1208ms for 'polyline' drawn gestures on a PDA and 438ms for "shapewriter" gestures on an ATOMIK keyboard. Wobbrock's "1$ recognizer" [Wob07], which uses a stylus to perform gestures similar to the ones tested upon a Pocket PC, includes 'fast' gestures that take approximately 600ms,

Table 4.1 The Leap Motion study gesture set's characteristics

Gesture Name	Stroke Count	Line Count	Curve Count	Mean (ms)
Circle	4	0	4	602
Lower d	4	2	2	918
Lower h	3	2	1	785
Capital P	3	1	2	618
Capital V	2	2	0	589
Slash	1	1	0	297

Table 4.2 RMSE values for the Leap Motion study's mean-value model in increments of 10ms

Constant Value	610	620	630	*640*	650	660	670
RMSE	270.2	269.2	268.6	*268.3*	268.4	268.9	269.8

although 'medium' and 'slow' gestures had means of 1153 and 1761ms.

When applied to the data, the model has a root mean square error of 268ms. By definition, the mean gives the lowest RMSE, but as seen in Table 4.2 that neighboring estimates are relatively close. Wobbrock's 'fast' gestures, which had a similar mean duration, had a standard deviation of 212ms and a performance error rate of 3.22% (similar to that of this study).

The critical drawback of the mean-value model is that the mean is going to be very dependent upon the set of gestures modeled. The more different gestures diverge in their mean performance values, either from each other or from the gestures in the set that inform the model the less accurate a constant value becomes. If an interface uses a gesture set that has performance profile close to that of "letter d", for example, then the mean-value model will be consistently biased.

$$MT = 640 \text{ milliseconds} \tag{4.1}$$

4.1.3.2 The Stroke Count Model

A stroke is an undifferentiated component—thus the stroke count of a gesture is the total number of components independent of type. The stroke count model multiplies this total by a constant and

Table 4.3 RMSE values for the Leap Motion study's stroke count model in increments of 10ms

Stroke Value	180	190	200	*210*	220	230	240
RMSE	263.6	253.2	246.2	*242.7*	243.1	247.3	255.0

the result is the predicted performance of the gesture. A process similar to that for the mean-value model suggests the per-stroke performance that minimizes RMSE is 210ms (rounded from 214ms, with an RMSE of 243ms), as noted in Table 4.3, leading to Eq. 4.2. As expected, RMSE is lower than for the mean-value model, even for estimates up to 30 ms away from the optimal.

$$MT = (210s) \text{ milliseconds} \tag{4.2}$$

4.1.3.3 The Component Count Model

The component count model treats the different types of components as having different duration values. These components are determined through discretization of a gesture's path, as indicated by the Pantomime code for a gesture. In the component count model, the *order* of the components is not relevant to the predicted duration, and neither is the *direction* of each component; any three-line gesture would have the same value, whether it's a triangle or a zig-zag.

The component model's optimal values can be found through the minimization of the RMSE, as per the other models. A curve's best-fitting value is 153 milliseconds (rounded to 150ms) and a line's is 305 milliseconds (rounded to 310), as seen in Table 4.4; the resulting model (Eq. 4.3, with c the number of quarter-circle components in the gesture and l the number of linear components) has a RMSE of 197ms, a further improvement over the stroke count model.

$$MT = (150c + 310l) \text{ milliseconds} \tag{4.3}$$

To test the stability of these estimates, cross-validation was performed upon the six gestures. In this process, each gesture is removed in turn and the values with the minimum RMSE were

Table 4.4 RMSE values for the Leap Motion study's component count model in increments of 10ms

	Curve Value						
	120	130	140	*150*	160	170	180
280	216.7	209.6	204.3	200.9	199.7	200.5	203.4
290	212.4	205.8	201.1	198.4	197.8	199.3	202.8
300	209.2	203.1	199.0	197.0	197.0	199.2	203.5
310	207.0	201.6	198.1	*196.7*	197.5	200.4	205.2
320	206.0	201.2	198.4	197.7	199.1	202.7	208.1
330	206.1	201.9	199.9	199.9	202.0	206.1	212.1
340	207.3	203.9	202.5	203.2	205.9	210.6	217.1

Table 4.5 The results of "take one gesture out" for each of the Leap Motion study models

	Base	Circle	Lower d	Lower h	Capital P	Capital V	Slash
Line (ms)	305	300	305	301	305	311	306
Curve (ms)	153	162	153	153	152	151	153
RMSE (ms)	197	188	197	191	196	200	207
Error (base)		1.7%	0.3%	2.8%	1.1%	3.6%	2.5%
Error (excluded)		7.7%	0.3%	3.8%	1.5%	5.6%	2.8%

determined, then the estimation error for the taken out gesture was calculated for those values. Results can be seen in Table 4.5. The estimation error rose slightly for each gesture except for the circle, which rose moderately, from 1.7% to 7.7%.

4.1.3.4 Comparing Models

These models can be compared by their RMSE values and by their estimation error, where estimation error is the percent difference between the estimated duration of a gesture of a given type and its mean value. This comparison is shown in Table 4.6.

The stroke count model dropped the RMSE of the mean-value model by 9.3% and the component count model dropped the RMSE from the stroke-count model by 18.9% (a total improvement from the mean-value model of 26.5%). Each model's estimations for each gesture was compared against the mean performance for that gesture and the difference calculated as a percentage. The component

Table 4.6 A comparison of the Leap Motion study's three models

Model	Equation	RMSE	Avg Estimation Error
Mean-value	$MT = 640$	268	30.3%
Stroke count	$MT = 210s$	243	23.0%
Component count	$MT = 150c + 310l$	197	2.0%

count model's error of 2.0% out-compete the stroke count model's 23.0% and the mean-value model's 30.3%.

Together, these measurements suggest that the component count model is a superior predictive model to use to model gestures. Moreover, given the very different values for the component types (310 and 150ms), the data supports differentiation of components as a useful distinction in the modeling of human gesturing. More importantly, in the context of task modeling, the component count model, if its predictions generalize, would constitute a very good engineering model for gestures.

4.1.4 Discussion

The Leap Motion study was designed as a trial run for the Kinect studies, as well as a way to support the fundamental assumptions of the component-based modeling approach. The assumption that gesture performance is strongly dependent upon gesture type was confirmed. The assumption that the discretization of a gesture into fixed components, a level of simplification and abstraction not seen in the calculations of Cao and Zhai for the drawing-based "Corners, Lines and Curves" [CZ07] (which also required performance-specific variables such as stroke length), would be sufficient for strong prediction was supported. And a relatively simple model based only on that discretization (Eq. 4.3) performed well; this model will be carried forward to the second stage for comparison.

4.2 Kinect Study

A second study was conducted using the Microsoft Kinect, a different camera-based gesture recognition device. A significant difference is that participants interacted with a simulated gesture-controlled application. The objective of this change was to acquire data that would mimic actual usage of a gesture interface. The primary purpose of this experiment was to develop a successful two-component model, much as was done in the Leap Motion study. Secondary purposes include comparison to the results of the previous study and slicing gesture-level data into component-level data to further confirm base assumptions.

4.2.1 Method

The core gesturing tasks in this study were embedded in a simple application designed to simulate goal-oriented usage. In particular, the participants were tasked with applying image transformations upon a series of pictures in a graphical interface.

Six different transformations were selected—turning a picture from color to grayscale, tinting it sepia in the style of old-fashioned photographs, rotating it exactly 45 degrees, flipping the picture as by a mirror, 'pulling' the picture inward as if it were warped by gravity and another mathematical warping that made it look 'wavy'. Each of these transformations would be triggered by a different gesture, and the application would update the image to display the result.

4.2.1.1 Task

Participants were given a set of 24 images in random sequence. They were directed to transform each image using two of the six possible gestures. The transformations ranged from simple flips and rotations to grayscaling and sepia-toning to a complex 'whirlpool' or 'wave' effect. The images displayed ranged from abstract designs to pictures of animals or common objects.

The pair of transformations for each image was pre-selected at random from the thirty possible

non-duplicating combinations. The pair was announced immediately before the image was loaded, the transformations were applied and the next pair was announced. Every four images the program was closed, the logs copied and the program re-launched. Due to inaccurate gesture recognition, a "Wizard of Oz" protocol (where the experimenter triggered the transformation after the gesture was complete) was used.

4.2.1.2 Apparatus

The Microsoft Kinect, Windows Edition consists of two cameras, a microphone and a motor-controlled adjustment mechanism. Some on-board pre-processing of the data from the cameras is done, and the rest is performed by the drivers in software. One of the cameras is an RGB spectrum camera, while the other is an infrared camera equipped with an infrared laser used to find the 'depth' of objects within the view.

The data available to the application from the hardware, firmware and drivers consists of three data streams. One is the RGB camera stream, one is the 'depth' stream and the third is the 'skeleton' stream, which uses pre-processed data to position each individual's joints in the field of view in three dimensions. The device can 'recognize' six figures and can simultaneously track the skeletons of two. Twenty joints (such as elbows, shoulders, knees and hips) or body parts (such as the head, hands, hip center and feet) are tracked in each frame of a skeleton stream.

A fourth stream, the interaction stream, can be created from the depth and skeletons data streams. The interaction frame allows location-sensitive action and detects an open or closed hand, along with more precise positioning data for hand location. The test application implemented the depth, skeleton and interaction streams.

The system detects individuals in a range between 0.8 and 4. meters away in an approximate 80-degree arc radiating from the device, but only 57 degrees allows for reliable detection. The first .4 meters and last half-meter has poor detection capability. The cone spreads slightly greater than 90 degrees vertically, but only the inner 43 degrees allows for reliable detection; the vertical limit at 3.5

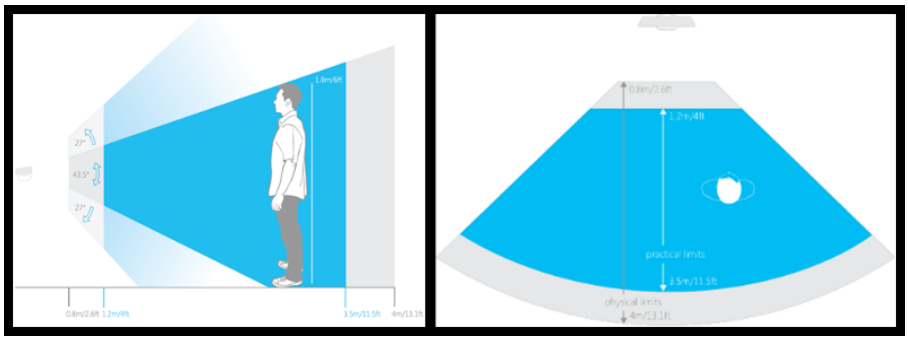

Figure 4.3 The field of view of the Microsoft Kinect for Window, courtesy of Microsoft

meters is listed as 2 meters. The Kinect has a 'near-range' model for higher precision that shrinks the outer limit by a meter. Near mode was turned on for the experiment. See Fig. 4.3.

The RGB camera has a resolution of 1920x960; the resolution of the infrared/depth camera is dependent upon the frame rate (due to the process of using a laser to find the depth of each 'pixel' during a frame) with a maximum of 320x240. Both cameras are locked to 30 frames per second.

While the Kinect is capable of video and audio recording, only the coordinates per frame of the participant's hands was recorded (in millimeters), and the non-gesturing hand's data was discarded. Each frame was also timestamped (in milliseconds). The frames recorded rate was variable, ranging from approximately 10 to 15 frames per second; frames with hand movement under a threshold were discarded. While the records were in millimeters, the nature of the Kinect's hand position calculation (from camera data) meant that precision was at a multi-millimeter level rather than at or below a single millimeter.

4.2.1.3 Participants and Task Environment

Twelve participants took part in the experiment; of these, two were removed for incomplete or garbled participant data. Participants were all undergraduate or graduate students in the computer science department, and ranged in age from 22 to 30. Two of the participants were female, the rest male. Nearly all had "little" (three participants) or "no" (six participants) previous experience with

gesture-command systems. Self-reported heights ranged from 155 to 185 cm. All chose to use their right hand to perform the gestures. Participants were remunerated with a ten dollar Amazon gift card. None of the studies shared any participants.

The lab environment was closed, with the participant standing approximately six feet in front of a projected display. The Kinect was mounted below the display and pointed at the participant. To the left of the projected screen was a white board with a guide displaying each of the six gestures by a 'trace' guide and the corresponding action. The experimenter sat to the right of the screen and directed the participant. Participation took approximately twenty minutes.

The Leap Motion study gesture set was chosen strictly for variety of performance—as gestures were performed in isolation, single-stroke gestures, for example, were viable for collection, whereas in a live application a single linear movement triggering an action would lead to numerous false positives.

The first Kinect study gesture set (listed in Table 4.8) was chosen to provide enough variety in line and stroke counts to allow the exposure of performance differences, but to also include gestures that should, according to the model, have similar performances but different vectors or arrangements of components. Differing means and distributions of these "gestural isomers" would indicate a possible flaw in the model or an anomaly that may require re-examination of the approach if severe enough. Due to the 'live' environment and concerns about participant memory and training, gestures were neither too short or too long.

4.2.2 Data

The records contain the same information as those in the Leap Motion study (participant, gesture type, familiarity and gesture duration), with the addition of the total length of the gesture performance. While the number of data points was collected, the varied frame rate made it useless for analysis purposes.

The raw data points were processed in the same manner as in the Leap Motion study, with the

exception of the distance between data points also being calculated. The beginning and end of each gesture was refined through both the XY scatter plot as previous and by points of no movement as determined by the power formula for distance between two three-dimensional points.

Of the 480 data records, 59 were removed for an indeterminable endpoint, malformed gesture or other problem, leaving 421 data records; a keep rate of 88%.

4.2.2.1 Gesture Type

Splitting the data by gesture type (Fig. 4.4) shows the relative means of the different gestures. Means and variances are a lot closer together than in the Leap Motion study, likely due to the similarity between several of the gestures; the stroke and component count models expect gesture isomers to have the same duration as each other.

Means ranged between 1131ms and 1569ms. A number of 'high' outliers, earlier seen in the participant/duration box plot, suggest some potential skewing of the linear estimation-derived model that may need to be addressed in model finalization.

It is in the gesture means the major difference between the two studies is clearest—in the Kinect study, the performance of gestures is significantly slower. One of the gestures, the circle, is shared between the two studies to act as a "signaler", a kind of sanity check—in the Leap Motion study the circle had a mean performance of 602ms and in the Kinect study it had a mean performance of 1147ms, approaching twice the duration.

Judging by both the means of gestures as a whole and the means for a common gesture, the Kinect gestures were both slower and more variable in performance than those performed for the Leap Motion. Normally in a time/accuracy tradeoff, slower performance would be correlated to more consistent performance—in this case the cause is suggested to be that of the *scale* of the movement. The Leap Motion has a small cone field of view, and gesturing is performed with the wrist and the forearm. However, the Microsoft Kinect records movement more coarsely, and gestures are performed with the forearm and upper arm. It could be that the key factor is the weight being

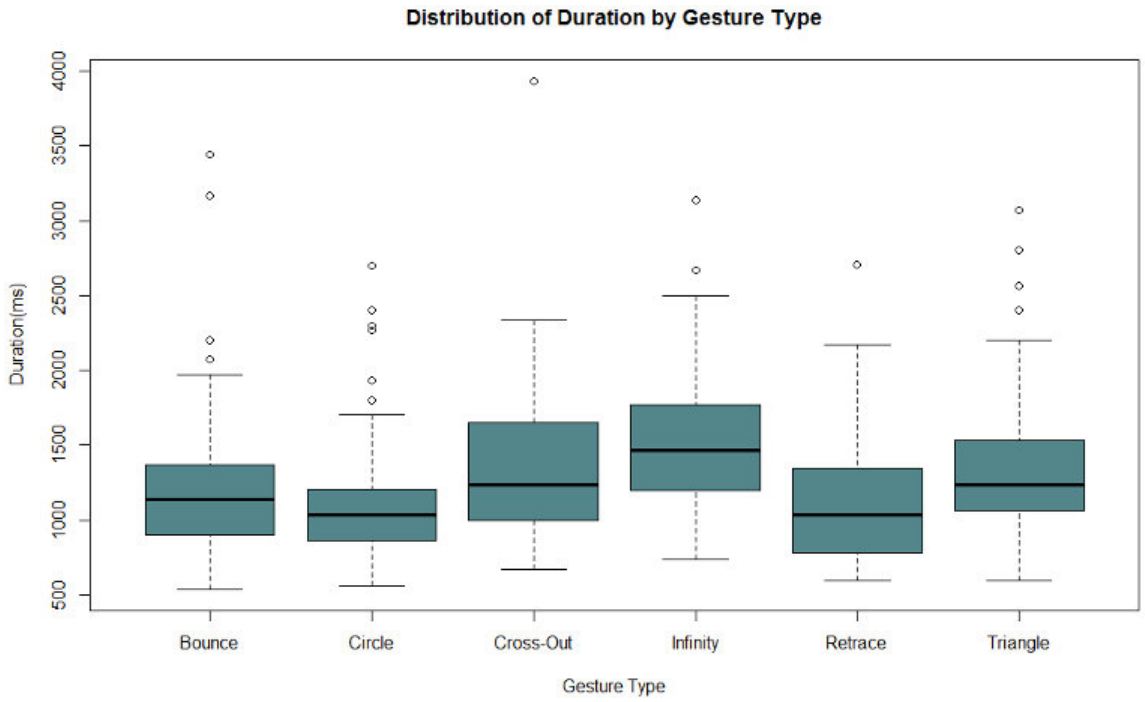

Figure 4.4 Performance by gesture type for the first Kinect study

moved—certainly Fitts' [Fit54] used both a 1-ounce and a 1-pound stylus in his tapping experiment and noticed different indices of performance and variances. Alternatively, the control of the hands and forearms may be more precise and easier on the cognitive side, leading to a slowdown in performance when the whole arm is used.

4.2.2.2 Component-Level Data

Two of the participants' records were extracted from the gesture-level data set and the gestures decomposed into components. This allows another avenue to examine the assumption of lines and curves being differentiated in performance and allows a glimpse at a lower, more mechanical level of performance. This division has limited power—the granularity of the Kinect movement (combined with frames where the hand has been lost track of) is such that a component could be only three or four data points long, and missing frames that occur where components meet lead to

Table 4.7 Per-participant values for curves and lines for component-level breakdown

Gesture-Level Model			
	Curve	Line	RMSE
934575	342	463	258
462910	299	401	244
Component-Level Model			
	Curve	Line	RMSE
934575	299	408	90
462910	262	357	68

uncertainty as to precise boundaries.

The participants used were selected at random and had means of 1243 and 1414ms over their gesturing, at neither extreme of performance. The results of examining both participants' performance on the gesture level for lines and curves and on the component level for the same (Table 4.7) suggest a clear division in performance between curves and lines.

More interesting is the difference between the calculations for lines and curves on the gesture level and the component level. In the component-level breakdown, transitions (intervals between data points in which little or no movement occurred) were removed. Anecdotally, not all components begin or end with these "micro-pauses", but on average they appear to cost approximately 13% of the duration of the components themselves over the course of the gesture.

The variable positioning of micro-pauses may at least partially contributes to the higher RMSE seen in the gesture-level model, although most is simply from the modeling of the gesture as a whole as opposed to each component individually.

4.2.2.3 Data Exploration Summary

The examination of gesture type, although on data more variable than in the Leap Motion study, suggests that a viable model could indeed be derived from their characteristics. The similarity in performance of gestures with similar component counts seems to suggest likewise that a model based on components will be consistent in its predictions.

Table 4.8 The Kinect study gesture set's characteristics

Gesture Name	Stroke Count	Line Count	Curve Count	Mean (ms)
Circle	4	0	4	1147
Infinity	4	2	2	1569
Flipped Triangle	3	3	0	1357
Retrace	3	3	0	1131
Cross-Out	3	3	0	1379
Bounce	4	0	4	1228

It also suggests, considering the difference in the scale of performance between the studies, a physiological or cognitive difference between performing with wrists and elbow and performing with elbows and shoulders.

The component-level breakdown seems to support the differentiated component approach and suggests a fairly consistent per-component overhead cost. This cost could come from managing acceleration and deceleration between components, or it could be that such moments are a natural cognitive 'break' to reassess task-level goals, process sensory data or attend to distractors.

4.2.3 Models

As with the Leap Motion study, three types of models were tested—the mean-value model, the stroke count model and the component count model. The best-fitting values for each model was determined using the relative RMSE.

After the models were compared using both the RMSE and the prediction error against the gesture means Table 4.8, a "take one out" protocol (along with the per-gesture predicted error) was used to detect any gestures that stood out from the others for further consideration.

The three finalized models were carried forward to be tested against the data from a second Kinect study, which acted as a validation data set as well as another source for data exploration and testing of the base assumptions of the modeling approach.

Table 4.9 RMSE values for the first Kinect study's mean-value model in increments of 10ms

Constant Value	1280	1290	1300	*1310*	1320	1330	1340
RMSE	506.6	506.2	506.023	506.020	506.2	506.6	507.2

Table 4.10 Estimation error values for the first Kinect study's mean-value model in increments of 10ms

Constant Value	1200	1210	1220	*1230*	1240	1250	1260
Estimation Error	10.2%	10.0%	9.8%	*9.65%*	9.73%	9.8%	9.9%

4.2.3.1 Mean-value Model

The constant value for the mean-value model determined by minimizing the RMSE for this data set (Table 4.9) is 1310 milliseconds (rounded from 1305ms), giving Eq. 4.4. This constant model, when applied, has a RMSE of 507ms. In comparison to the Leap Motion mean-value model, this is a significant increase in both value and variability. It should be noted that the range of gesture mean values in this study is less than that in the previous study—in the Leap Motion study, the shortest gesture had a mean performance time of 32.5% of the longest, while in this study the same percentage is 72.1%.

$$MT = 1310 \text{ milliseconds} \tag{4.4}$$

4.2.3.2 Stroke Count Model

In this Kinect study, the minimal RMSE is 534ms, at a value of 367ms (rounded to 370ms) per stroke, giving Eq. 4.5. This is actually *poorer* than that of the mean-value model, a departure from the relationship of the previous study.

Looking at the component-level breakdown of data, along with the component values in the Leap Motion study, suggests the reason. Linear components have a performance value well above that of curves—a model that does not differentiate them will regress to a value approximately in-between

Table 4.11 RMSE values for the first Kinect study's stroke count model in increments of 10ms

Stroke Value	340	350	360	*370*	380	390	400
RMSE	542	537	534	*533*	535	540	546

Table 4.12 Estimation error values for the first Kinect study's stroke count model in increments of 10ms

Stroke Value	340	350	360	*370*	380	390	400
Estimation Error	17.2%	16.8%	16.2%	*15.8%*	15.6%	16.0%	17.1%

them. The gesture set for this study, in order to test whether gestures that would have a similar predicted performance value (i.e., the same component counts) but are performed differently would have the same actual performance value, has several such gestures—three of the six consist of three linear components, and two of the six consist of four curved components.

As a stroke count would (assuming the component count model is correct) over-estimate a gesture composed of lines and under-estimate a gesture composed of curves, a gesture set that has gestures with fewer total components but more lines and gestures with more total components but more curves may cause the stroke count model to perform worse than a mean-value model.

$$MT = (370s) \text{ milliseconds} \tag{4.5}$$

4.2.3.3 Component Count Model

The best fit for the component count model is a value of 300ms for curves and 440ms for lines (Table 4.13), leading to Eq. 4.6, with an RMSE of 492ms. The error measurement is better than that for strokes and very slightly better than the mean-value model; the former fits the results from the first study, although the latter suggests minimal improvement with the more complex model.

Three of the gestures having the same component counts and two others sharing their own counts, combined with the greater variance of the data for this study in general, are likely the causes of the "flattening" of the root mean square error values across the three model approaches.

Table 4.13 RMSE values for the first Kinect study's component count model in increments of 10ms

	Curve Value						
	270	280	290	*300*	310	320	330
410	502	499	496	495.0	494.8	496	498
420	500	496	494	492.66	492.68	949	496
430	498	495	492	491	492	493	495
440	497	494	492	*491.1*	491.5	493	496
450	498	495	493	492.0	492.4	494	497
460	499	496	494.5	493.9	494.5	496	499
470	502	499	497.3	496.9	498	499	502

Table 4.14 Estimation error values for the first Kinect study's component count model in increments of 10ms

	Curve Value						
	270	280	290	*300*	310	320	330
410	10.0%	8.7%	7.7%	7.6%	7.7%	8.6%	9.5%
420	9.5%	8.2%	7.2%	7.1%	7.2%	8.1%	9.0%
430	9.0%	7.7%	6.7%	6.6%	6.7%	7.6%	8.5%
440	8.5%	7.2%	6.2%	*6.1%*	6.2%	7.1%	8.0%
450	8.0%	6.7%	5.7%	*5.5%*	5.7%	6.6%	7.5%
460	8.1%	6.8%	5.8%	5.6%	5.8%	6.7%	7.8%
470	9.1%	7.7%	6.8%	6.6%	6.7%	7.9%	9.2%

Table 4.15 The results of cross-validation for each of the Kinect study gestures on the component count regression

	Base	Cross-Out	Retrace	Flipped Triangle	Bounce	Circle	Infinity
Line (ms)	437	428	460	429	438	435	430
Curve (ms)	303	304	300	304	299	318	297
RMSE (ms)	492	482	496	498	488	505	481
Error (base)		4.9%	15.9%	3.4%	1.3%	5.7%	5.7%
Error (excluded)		6.9%	22.0%	5.2%	2.6%	10.9%	7.4%

$$MT = (440l + 300c)\,\text{milliseconds} \tag{4.6}$$

As in the previous study, a cross-validation procedure was applied to the six gestures (Table 4.15) to determine to what extent each gesture might have impacted the overall fit of the component count model. Two of the gestures had a moderate increase from estimation error to prediction error rate when excluded—the circle and the retrace. The circle is a common gesture between the Leap Motion and first Kinect studies, and in both cases had a similar (5.2% and 6.0%) increase in error during cross-validation. Meanwhile, the retrace gesture took an already-high prediction error, 15.9% and increased it by just over 6%.

The results are promising for the component count model. The component constants resulting from each gesture being excluded vary only slightly (with the notable exception of the retrace gesture and, to a lesser extent, the circle gesture), and the prediction error for the excluded gesture remains low for all gestures except the “retrace” gesture.

It is clear from both the base estimation error and the excluded estimation error that the retrace gesture has an abnormal performance under the component count model. Meanwhile, the circle, across both studies, also stands out, albeit to a lesser extent.

Table 4.16 A comparison of the Kinect study's three models

Model	Equation	RMSE	Estimation Error
Mean-value	$MT = 1310$	506	10.3%
Stroke count	$MT = 370s$	533	15.8%
Component count	$MT = 300c + 440l$	491	6.1%

4.2.3.4 Comparison of Models

Each model is based on the minimal RMSE value for the model's equation against the data. Examining the minimal estimation value for the same (Table 4.10, Table 4.12 and Table 4.14) shows that for the stroke count and component count models, the minimum estimation error is very close to the model values, within a single 10ms 'step'. However, the mean-value model continues to decrease the estimation error for eight such steps—likely due to the RMSE statistic being impacted by the high-value outliers in the data.

A comparison of the three models (Table 4.16) still suggests, as for the Leap Motion study, that a component count model is an improvement over both the mean-value model and the stroke count model, although the improvement is more minimal, with only a 3% improvement in the residuals (as measured by the RMSE).

Using the error of estimation of mean gesture performance values, these same ordering of models occurs, with the component count out-performing the mean-value model, and the stroke count model being outperformed by both, perhaps due to the gesture set as noted earlier.

There is one usually high error rate among the six gestures, however. The gesture for grayscale, the "retrace" ("5s1s5s"' as a Pantomime component sequence) has an error rate of 16.7% in the component count model; removing it shifts the estimation error closer to the estimation error found in the Leap Motion study. The reason for the departure from the other two gestures that share the same estimated value is uncertain, but is likely related to either the backtracking of the first stroke during the second, or the repeat of the first stroke during the third. There is insufficient data in the other studies performed to be certain, however.

4.2.4 Results

In summary, the clear results of the Leap Motion study, with the component count model better than the stroke count model and both better than the constant mean-value model, became less clear with the scaling up to the Kinect, although the component count model still outperforms the other two.

The three models in Table 4.16 will later be tested against the third study's validation data to confirm the relative utility of each, but the initial results suggest the discretization-based component-stroke model outperforms both an undifferentiated stroke count and a constant mean-value model both in fitting the data and in predicting gesture performance.

4.2.4.1 Assumptions of Pantomime

The comparison of gestures with the same estimated durations but different shapes in the gesture set supports the assumption of consistency—that is, that gestures with the same component counts will have similar performance times. If this had not been the case, then that would suggest that there is a necessary additional factor that the component count model does not capture.

The 'retrace' gesture assigned to the "Grayscale" command (Pantomime code "5s1s5s") performed faster than two other gestures that shared its component counts; this suggests that either reversing or retracing a component may be faster than performing a novel component.

The Kinect data had far more variance than the Leap Motion data; this may be inherent to the larger scale of movement or may be an artifact of noise specific to this data set.

4.2.4.2 Limitations

The models derived from the RMSE showed an estimation error that is quite low—both for the Leap Motion and this study, at about 2% and 6.1%. While this appears quite impressive, it not only is an estimation against trained data but it ignores the variability of the gesturing activity expressed in the coefficients of correlation and the root mean square error. Combined with the variances seen

in both the Leap Motion study and in other literature, it is clear that gesturing as a 'mid-level' task is less predictable than low-level mechanical tasks such as pointing or typing tasks. Moreover, the danger of over-fitting requires that these values be taken with caution without testing against fresh data.

The improvements seen in the RMSE values and the estimation error were more modest than those seen in the Leap Motion study, likely due to the relatively small range of possible values among the six gestures for stroke, line and curve counts.

4.3 Second Kinect Study

The third study, also performed on the Microsoft Kinect, is meant to test the results of the second study. The second study's gesture set was in some ways limited—all six gestures were composed of three or four strokes, and there were only three separate predicted performances (that for 3 lines, for 4 curves and for 2 lines and 2 curves). While this allowed the testing the consistency of performance across gestures with the same components but different order, with the accompanying discovery of the unusual performance of a "retrace" gesture, it, along with a much larger variance in the data than the Leap Motion study, also meant that improvement in correlation from the mean-value model through to the component stroke model was weaker. The gesture set for the third study has a larger variety of predicted performance values.

4.3.1 Methods

The second Kinect study, as a follow-on from the first, had the same methods and design parameters with the exception of a different gesture set (Table 4.17 and a different set of participants. The experimental environment and the software used was unchanged as well.

The gesture set had a slightly broader range of stroke counts (two to four) and component count predictions (four) than the first study's gesture set. Participants were gathered through the same methods.

Table 4.17 The Second Kinect study gesture set's characteristics

Gesture Name	Stroke Count	Line Count	Curve Count	Mean (ms)
Spiral	4	0	4	1547
Rocking	4	0	4	1374
Triangle	3	3	0	1676
Left Bracket	2	2	0	1144
Letter N	3	3	0	1602
4-4 Time	4	4	0	2318

4.3.2 Data

Data records were identical in format to those of the first Kinect study. One of the 12 participants was discarded for improper performance (gestures were consistently performed slowly) before processing; of the remaining 528 records, 66 were removed, leaving 462 data records; a keep rate of 87.5%.

4.3.2.1 Data Exploration Summary

Exploration of the data suggests that gestures are distinct in performances, as expected from the rests of the previous two studies, with similar variances in spite of different mean performances.

4.3.3 Models

The three models from the previous study were tested against the data from this validation study using first the RMSE and then prediction error; the former in order to compare the models against one another for relative fitness and the latter to consider accuracy against John's "80% accuracy with 20% effort" [JK94].

4.3.3.1 Comparison of Models

The first of the three models is the mean-value model, which found that a constant value of 1310 milliseconds minimized the RMSE. As noted in the table (Table 4.18), the mean value model's RMSE

Table 4.18 Comparison of the models to the validation data

Model Name	Equation	RMSE	Prediction Error	Swirl	Triangle	4-4 Time	N	Triangle	Bracket
Mean-value	$MT = 1310$ms	638	19.7%	15.3%	21.8%	43.5%	18.2%	14.5%	4.7%
Stroke count	$MT = (370s)$ms	652	24.6%	4.3%	33.7%	36.2%	30.7%	35.3%	7.7%
Component count	$MT = (440l + 300c)$ms	548	20.2%	22.4%	21.2%	24.1%	17.6%	23.1%	12.7%

was 638ms, with an average prediction error of 19.7%. The per-gesture prediction errors ranged wildly, however, from 4.7% to 43.5%.

This unpredictability is a problem for using the mean-value as a task model; it means that some gesture sets will simply be poorly predicted—40% is well beyond the limitation of the benchmark target.

The second is the stroke count model, using a value of 370ms per stroke. This stroke-count model performed worse than the mean-value model against the new data set as well, with an RMSE of 657ms and an average prediction error of 24.7%. Like the mean-value model, the per-gesture prediction errors varied significantly, from 4.3% to 36.2%.

The third is the component count model, using a value of 440ms per linear component and 300ms per curved component. Against the new data, the RMSE is well below that of the other two models, at 561ms, although the average prediction error is slightly above that of the mean-value model at 20.2%. The per-gesture prediction errors were more consistent, ranging between 12.7% and 24.1% and most hovering around 20%.

4.3.4 Results

The component count model continued to outperform both the stroke count model and the mean-value model when applied to new data as measured by the root mean square error. The average prediction error suggests that both the mean-value and component count models performed acceptably near the benchmark (while the stroke count performed somewhat below the target), but examining the per-gesture prediction errors showed a much greater variance for the mean-value and stroke count models, rendering them less reliable as task models.

4.4 Further Discussion

With the task model established and tested, there were several other things worth discussing in the data and the modeling process that may lead to future avenues of research. The third study was used as a validation study, with the focus upon testing the existing models against the data—however, with that accomplished the processes used to create the models using the second data set can be used to create new versions of the models. Those models can then be compared against the second data set, swapping the roles of the two studies.

While the Leap Motion didn't collect additional information, the two Kinect studies also collected a "length of gesture" variable determined from the sum of the distances between the raw data points. This variable has several potential uses, including seeing the variance in human performance, measuring the difference between the scale of gesturing in two data sets and, with the duration, determining the speed at which gestures were performed for a study. This also leads to the ability to 'scale' a model of gestures by the speed of gesturing.

4.4.1 Second Kinect Data Exploration

In the second Kinect study, the duration can be split by gesture type (Fig. 4.5), showing a clear split of performance, as per the previous two studies. Means range between 1144 and 1676 milliseconds.

The box plots for the gesture type vary more than in the first Kinect study, which matches the greater variety in the gesture signatures. The two three-line gestures are quite similar in spread, while the two four-curve gestures show a tighter spread for the "rocking" gesture than the spiral.

While the gesture sets for the two Kinect studies differ, there are two very similar gestures shared between them as a signaler similar to the circle gesture between the Leap Motion and first Kinect study; the triangle (flipped between studies). Comparison of mean performances of these gestures between the sets suggest that the validation study was performed about 25% slower; a potential concern for the models, as it suggests some unrecognized variable at play leading to the "slower but

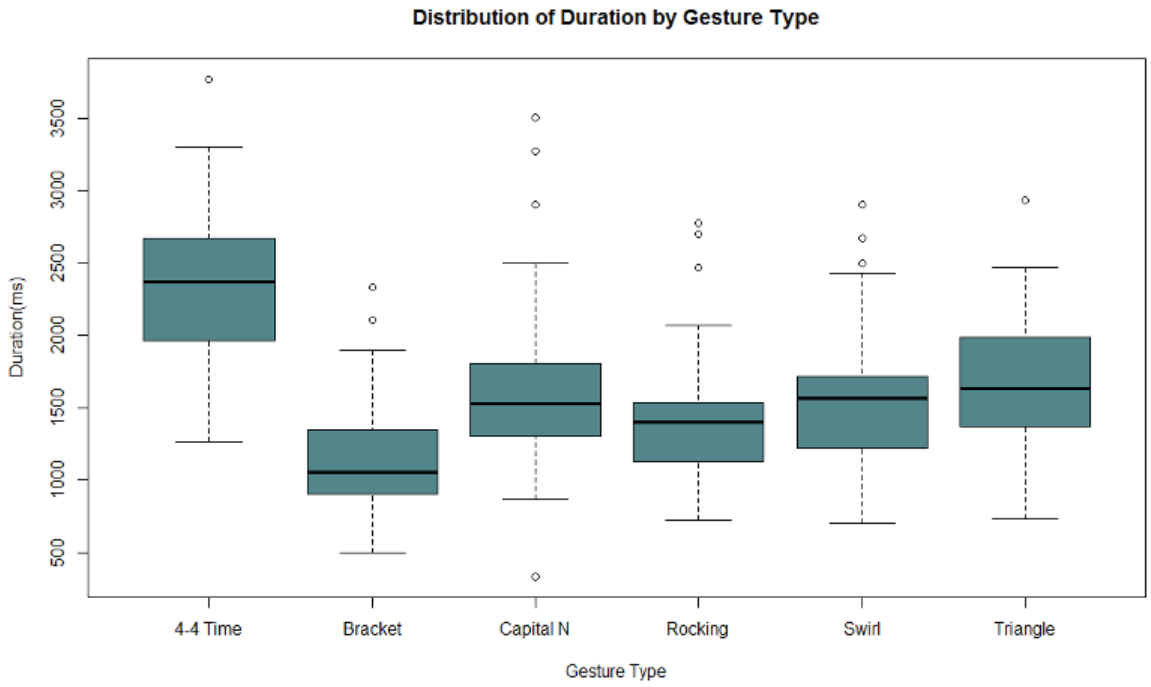

Figure 4.5 Performance by gesture type for the second Kinect study

more consistent performance" hinted at by the participant variance.

4.4.1.1 Second Kinect Study Models

Continuing the data analysis in the same form as the previous studies, the best-fitting values for all three model approaches were calculated for the second Kinect study's data (see Eq. 4.7, Eq. 4.8 and Eq. 4.9). The mean-value model fitted closest to the validation study data is 1600 milliseconds per gesture, in comparison to the 1310ms from the previous study and 1640ms from the "scaled" value, while the RMSE is 568ms.

The stroke count model fitted closest to the validation study data is 471 milliseconds, well off from the model derived from the previous study but close to the "scaled" stroke count model at 460ms. The RMSE 559, which shows a slight improvement over the data-derived mean-stroke model—further suggesting that the first Kinect study's gesture set demonstrated a vulnerability in

the undifferentiated stroke count.

The component count model fitted to the validation study data assigns 360 milliseconds to each curve and 560 milliseconds to each line, significantly greater than the unscaled finalized model but extremely close to the scaled finalized model. The RMSE was 447ms.

$$MT(mean-value) = 1600 \text{ milliseconds} \tag{4.7}$$

$$MT(strokecount) = (470s) \text{ milliseconds} \tag{4.8}$$

$$MT(componentcount) = (560l + 360c) \text{ milliseconds} \tag{4.9}$$

These models are quite different in magnitude from the equivalent models produced from the first Kinect study; 300ms difference for the mean-value model, 100ms per stroke for the stroke count model and 120ms and 60ms per line and curve respectively for the component count model (although the ratios between components are similar, 1.47:1 and 1.56:1). This leads to a question - what would be the results if the studies had been conducted in a different order?

4.4.1.2 Swapping the studies

When applied to the data from the first Kinect study, the models from the second Kinect study perform differently than the forward process. The 1600ms mean-value model has a RMSE of 586ms (as opposed to 638 when 'un-flipped') and a prediction error of 24.5% instead of 19.7% - fitting the data closer but less accurately predicting gesture performance. In this case the larger number of higher outliers in the second study likely reduces the RMSE (as the 'new' model over-predicts), while the lower average has a prediction accuracy cost.

The 470ms per stroke stroke count model has an RMSE of 646ms, close to the 658ms of the un-flipped model, and a prediction error of 27.9% (contrasting to 24.7% of the un-flipped model).

Interestingly, the relative RMSE of the mean-value and stroke-count model have flipped from the second to the first Kinect study—likely due to the same issue with the stroke count model and certain gesture sets noted earlier.

The 560ms per line and 360ms per curve component count model has an RMSE of 591ms, moderately higher than the 561ms of the un-flipped model in similar circumstances, and a prediction error of 25.7%. In this case the component count model is outperformed by the mean-value model on average, in a reverse of the circumstance when the two model approaches were trained on the first Kinect study and compared.

The per-gesture prediction errors do follow the same pattern as seen previously— the mean-value model's range from 1.9% to 41.4% fairly evenly, the stroke count model's likewise from 2.5% to 63.9% and the component count model's closer fairly tightly around 17.2% to 25.5%, with a single large outlier of 48.5% on the "retrace" gesture.

Removing that gesture drops the RMSE of the component count model to 570ms and the prediction error to 21.1%, far closer than the original arrangement, while the mean-value model's RMSE is reduced to 577ms and the prediction error to 21.1% as well. The same actually increases the stroke count model's RMSE to 670ms and prediction error to 28.6%. Determining the circumstances that led to this gesture's unusual performance would require additional studies, but could lead to additional component types or rules for a refined component count model.

While the model developed from the first Kinect study validated against the second Kinect study's data, it seems clear that there's a difference in the performance between the gestures of the two data sets. While some of the increase in prediction error and RMSE for the models can be explained by an anomalous gesture, it's worth considering what aspects make for a good validation data set for gestures.

A key aspect is gesture *variety*. In the first Kinect study, there is little variety in stroke and component counts, partially due to the need to test how consistent gestures with the same estimated duration would actually be, while in the second Kinect study and in the Leap Motion study there is

greater variety. In addition, the choice of six gestures to minimize the impact of recall and learning (and due to few total participants) may not be desirable for future experiments; either multiple gesture sets should be used (with a correspondingly larger participant pool) or a longer-term study with a large enough learning period to teach perhaps a dozen or more gestures instead.

The improved performance of the mean-value model is in part explained by that strong homogeneity of the gestures in the first Kinect data set, but another way to look for differences between the sets is through examination of an additional variable collected in the two Kinect studies, the gesture length.

4.4.2 Length of Gestures

The two Kinect studies collected the length of each gesture as well as duration; distance between each pair of data points in the gesture was measured and the total was recorded. While not directly relevant to the production of a task model of gestures usable *a priori* to predict gesture performance, it offers a glimpse into human performance of gesture from a different direction.

4.4.2.1 Length Variable for Kinect Studies

The length variable tracks to duration reasonable well in the first study; with a multiplier of 0.422ms per mm, the RMSE was 542ms, comparable to that of the stroke count model developed using the same data. However, a model based on length would not be *a priori*, and would therefore not be an engineering model as per the requirements. In addition, the information captured in the length variable can already likely be approximated with the number and type of components (i.e., the component count model); the other major factor that would go into the length of a gesture performance would be the scale of performance, which varies significantly between participants as seen in Fig. 4.6.

While not *a priori* information, the length does provide an indication of the speed of the gesturing; this could be used in future studies in a similar fashion to the “effective width” seen in some Fitts’

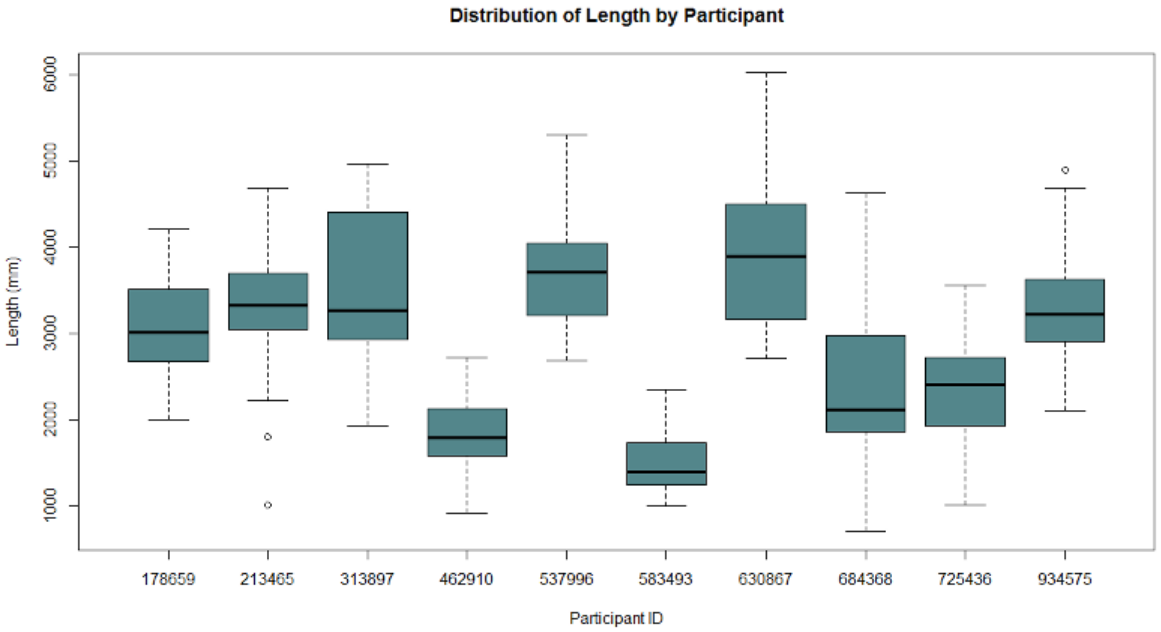

Figure 4.6 Movement distance by participant per gesture in the first Kinect study

Law studies.

Splitting the data by gesture and measuring the gesture length, as seen in Fig. 4.7, doesn't offer much information - naturally the amount of distance is going to closely align with how many different movements are needed to complete the gesture. However, splitting by *stroke and component counts* removes that as a factor and leaves us with an understanding of the distance to perform each component. This likewise could be used in future studies to compare the scale of gestures; unfortunately, the Leap Motion study data set did not include a similar variable.

In the second Kinect study, the distance variable, when treated as a model, shows an RMSE of 676ms. The best-fitting value (0.670ms per mm), however, compares with that of the first study (0.422ms per mm) to demonstrate a slower raw movement speed. In looking at the experimental task and protocols, the cause of this slowdown is unknown—it could be more hesitancy with the gestures in the set, an artifact of unwitting experimenter emphasis during teaching, time of day or some other uncontrolled variable. It could also be that the first study was performed particularly

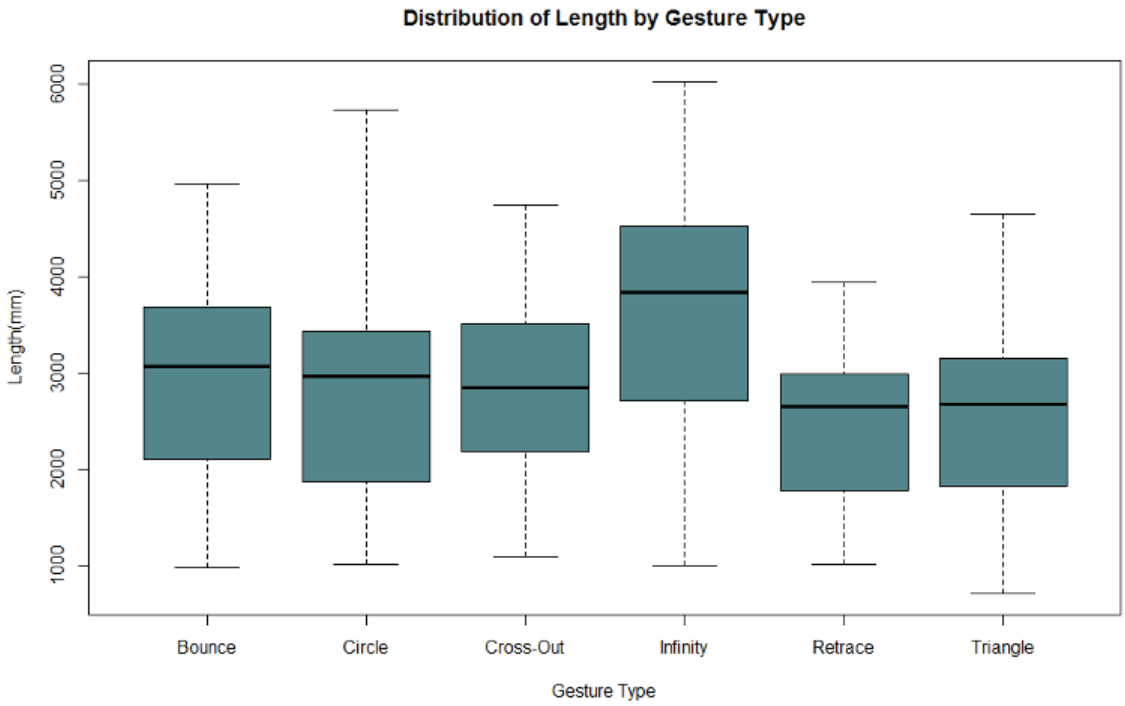

Figure 4.7 Movement distance by gesture type in the first Kinect study

hastily.

The length was also split by participant (Fig. 4.8), by gesture (Fig. 4.9) and by stroke and component counts (Table 4.20), as per the first Kinect study. In comparison, both the estimated stroke length and component lengths are shorter; that is, that independent of how many strokes or components a gesture consists of, the performance during the second Kinect study was 'tighter' in scale than in the first.

Table 4.19 The average length of undifferentiated strokes and of differentiated components in the first Kinect study

Variable	Mean (mm)	RMSE (mm)
Stroke	820	987
Line	901	959
Curve	746	959

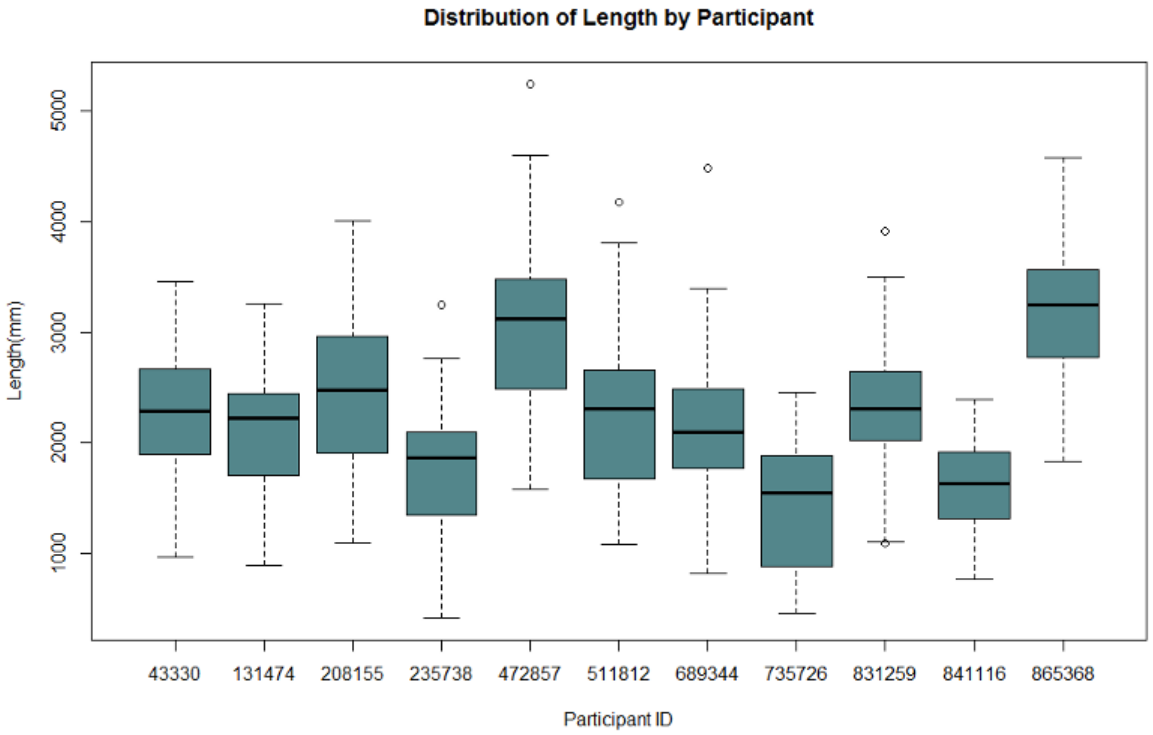

Figure 4.8 Movement distance by participant per gesture in the second Kinect study

Table 4.20 The average length of undifferentiated strokes and of differentiated components in the second Kinect study

Variable	Mean (mm)	RMSE (mm)
Stroke	683	729
Line	746	693
Curve	610	693

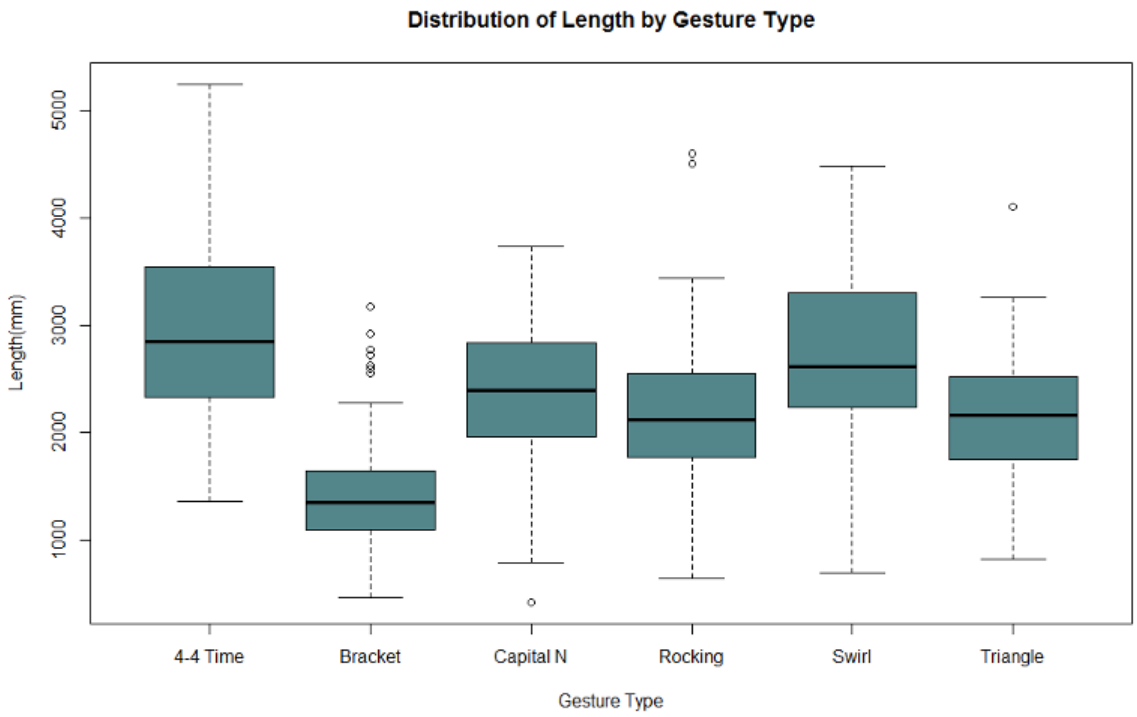

Figure 4.9 Movement distance by gesture type in the second Kinect study

The two data sets show a distinctive difference in speed of performance, along with a similar but inverse difference in the scale. These measurements could possibly be useful in calibration of gesture performance data across different sets that may have had different instructions or environments that would impact gesture performance. They also allow the measurement of external factors upon gesture performance; for example, speed and scale may vary between an interface that uses gesture alone and a multi-modal interface that uses gesture and speech.

More immediately, however, the difference in results of regression of length against duration confirms the existence of a difference between the two data sets that impacts prediction and model fit. This leads to an additional question to explore—could models of gesture be adjusted based on this information? While the relative impact of scale and speed upon performance is unknown and even whether such a factor would be linear, logarithmic or exponential is unknown and would need more data to determine, a scaling factor can be estimated for the sake of data exploration.

4.4.2.2 Scaling the Models

In the second Kinect study, all three models performed worse, both in terms of RMSE and prediction error, than the equivalent in the training sets. Moreover, all three models almost universally *under*-predicted the data, suggesting the existence of a bias in the data set.

The triangle included in both Kinect data sets (albeit rotated) additionally suggests a measurable performance difference between the two sets - as noted previously, the triangle in the second data set had a performance 24% slower than that of the triangle in the first data set. Together with the under-prediction and the difference in speed and scale shown by the comparisons of gesture length, this suggests it might be worth examining what a scaling factor (based on the two triangles) might do to the validated models. The results can be seen in Table 4.21.

For the mean-value model, prediction error still varies significantly (from 1.1% to 41.6%), although the RMSE drops significantly and the prediction error drops moderately. The stroke count model follows suit, ranging from 13.9% to 33.9% per gesture and the RMSE and average prediction

Table 4.21 Comparison of the scaled models to the Validation data

Model Name	Equation	RMSE (ms)	Avg Error	Swirl	Triangle	4-4 Time	N	Bracket	Rocking
Mean-value	$MT = 1620$ms	568	16.5%	4.7%	3.3%	30.1%	1.1%	41.6%	17.9%
Stroke count	$MT = (460s)$ms	560	2089%	19.0%	17.6%	20.7%	13.9%	19.6%	33.9%
Component count	$MT = (550l + 370c)$ms	447	4.3%	4.3%	1.5%	5.1%	3.0%	3.9%	7.7%

error dropping moderately as well. The component count's RMSE drops by a similar amount but the prediction error drops precipitously, nearly 16% to 4.3%, and keeps the same consistency in prediction error across all gestures as before.

The scaled-up models are actually quite close to the 'best-fit' models based on the same data, suggesting that the factor that slowed the performance down was constant across all gestures (as the scaling factor itself was determined from a pair of similar gestures). However, how such a scaling factor might be applied across data sets without common gestures is uncertain from the data; the ratio between the speed in each data set was 5:8 and the ratio between the scales in each data set was 6:5. Future experiments will be needed to test different possibilities.

4.4.2.3 Which Model is Best?

Two data sets performed at different speeds and scales naturally leads to a question - "Which is closer to the average performance of gestures at the Kinect scale? The first set, performed faster, slightly more variable and at a slightly larger scale, or the second set, performed slower and at a smaller scale?" With only two data sets there is no clear answer—it's only with more experiments, perhaps with larger gesture sets and more participants, that a satisfactory answer might be found.

Wobbrock [Wob07] lists drawn gestures as 'fast', 'medium' or 'slow' with performance times ranging from 600ms to 1761ms, for example, but what criteria might be used to determine if any given gesture is performed 'fast' or 'slow'? Fitts' Law research may hold an answer, with the changing of the width of a target until the error rate matches a 4% target, or the speed of performance (gained from the length and duration of a gesture) discussed above may allow the classification of gestures.

4.5 Summary Analysis

The previous chapter described a theoretical task model called the component count model, in which Pantomime components differentiated by type were used to make a prediction of performance time for gestures. The three experiments described in this chapter took the proposed Pantomime component count model and two alternatives, the undifferentiated stroke count model and the mean-value model, and tested them against collected data.

Throughout the three experiments, examination of the data indicated that the two base component types had consistently different performances, and that gesture durations could be predicted from the components that they decompose into. While the values found for the component count model(Eq. 4.10) based on a single six-gesture data set may likely be able to be refined with further experiments, they offer predictive accuracy approximately at the 80% accuracy benchmark established for task models by Bonnie E. John in her work on GOMS.

$$MT = (440l + 300c) \text{ milliseconds} \tag{4.10}$$

Preliminary models performed on a smaller physical scale for the Leap Motion demonstrated a clear improvement in both data fit and predictive value for the component count model and a mild one for the stroke count model. The larger scale gestures performed for the Kinect had a similarly larger performance time and a more homogenous gesture set, decreasing the improvement of the component count model over the mean-value model significantly and demonstrating a vulnerability of an undifferentiated component-based model such as the stroke count model. Application of the final Kinect data set to the previous Kinect gesture models showed a clear improvement in model fit in the component count model over the competing models.

Table 4.22 Example action frequencies for the experimental application.

Action Name	Flip	Rotate	Grayscale	Sepia Tone	Whirlpool	Waves
Frequency	10%	16%	24%	28%	8%	14%

Table 4.23 Existing mapping of gestures to actions and optimal mapping for validation data set.

		Existing Mapping			**Optimal Mapping**		
Gesture	**Duration**	**Action**	**Freq.**	**Cost**	**Action**	**Freq.**	**Cost**
Triangle (7s2s5s)	1320	Flip	.10	132	Flip	.10	132
Bracket (3s5s)	880	Rotate	.16	140.8	Sepia Tone	.28	246.4
Wave (3k1k5c7c)	1200	Waves	.14	168	Grayscale	.24	288
4-4 Time (4s7s2s7s)	1760	Sepia Tone	.28	492.8	Whirlpool	.08	140.8
Letter N (4s1s4s)	1320	Grayscale	.24	316.8	Waves	.14	184.8
Swirl (5k3k1k7k)	1200	Whirlpool	.08	96	Rotate	.16	192
TOTAL COST				*1346.4*			*1184*

4.5.1 Application to Optimal Mapping

The component count model can be placed as-is in KLM or other GOMS modeling techniques as an operator. However, there are other uses for a predictive task model of gestures. One way in which the quantitative predictions of the component count model can be used to improve a gesture design is optimizing the mapping of gestures to actions. An interface's possible actions are going to be used with greater or lesser frequency; if these frequencies can be estimated, the lowest-duration gesture can be matched with the highest-frequency action. For example, given the gesture set for the validation data and an interface in which the six actions those gestures triggered have the frequencies shown in Table 4.22, we can compare different mappings.

The mapping in the participants' performances leads to the costs shown in Table 4.23. A simple process of matching the highest-duration gesture to the lowest-frequency action leads to an optimal mapping.

4.5.2 Additional Results

All models, however, consistently under-predicted performance in the validation study. Distance information gathered as part of the two Kinect studies, along with a common rotated shape, indicated that gestures in the two studies were performed at different speeds. Scaling up the competing models by a factor derived from the common shape removed much of the bias, improving the fit of all three models evenly but improving the prediction accuracy of only the component count model significantly, from 20.2% to 4.3%. The source of the slower performance is currently unknown, but future experiments recording not only performance time but performance distance and/or having a common gesture could likewise compensate for this source of variance.

The unusually high prediction error for the "retrace" gesture in the first Kinect study is of interest; the discount on the performance time for this gesture suggests that some factor of its component sequence, such as the reversal or the repetition of the same exact stroke, might be worth further study to determine a possible rule for faster and more efficient gesture interfaces aside from what the difference in component costs may offer.

4.5.3 Context and Future Work

The current work, while successful in producing a usable task model for free-space movement-based gesture performance that is more consistent than a constant factor and demonstrating the differentiation of component production times, is only a beginning. Additional experiments could refine the equation further, allow the introduction of a simple or complex scaling factor for gesturing at different scales or lead to further improvements in both the task model offered here or in additional task models based on Pantomime's component sequence.

The component count model may be applicable to not just free-space gestures but to any unpaced movement-based gesture, from different scales of performance as exhibited here with the Leap Motion and the Kinect or to drawn gestures on a surface such as a PDA pad. However, the component count model would need to be fitted against each domain or a consistent predictable

relationship between individual component count models determined.

CHAPTER

5

CONCLUSION

This chapter provides a summary of the overall contribution of this work, from the context of a representation of gestures to the representation itself to experiments to inform and test the Pantomime task model and the fundamental assumptions of the component-based approach to gesture modeling. With the work defined, the possible uses of a functioning task model as well as the limitations of the work at present are outlined. The chapter then finishes with exploration of a number of avenues for future work, from refinement of the Pantomime task model to expansion into other domains and applications.

5.1 Work Summary

Chapter 1 defined the scope of the work as free-space movement-based gestures and discussed how a task model of gestures would be useful both for future task analysis research and for interface design in an input modality ripe for expansion in popular use. It specifies the driving focus of the research, *task models can represent a user's execution of free-space gestures to make predictions of duration at a level of granularity and accuracy sufficient to act as engineering models of performance*, and introduces Pantomime as a possible gesture task modeling system.

Chapter 2 discussed the domain of task models, focusing in particular upon two; the low-level mechanical modeling of Fitts' Law and the collection of task models that compose the Keystroke-Level Model modeling framework. Establishing a context for an additional task model, the chapter took care to note the ways in which task models have been evaluated and the criteria, such as the "80%/20%" benchmark, against which a task model of gestures might in turn be compared.

Chapter 3 introduced the idea of the Pantomime representation that sits behind the task model. Using three existing gesture representations used for non-modeling purpose, this chapter stepped through the ways in which a representation could be used and stepped through the process of construction a simple representation of a gesture through decomposition into a component sequence, finishing on a simple task model able to produce *a priori* predictions of gesture performance.

Chapter 4 discussed three experiments of gesture performed on two different devices to test the assumptions of the component-based approach to gesture modeling; both informing and validating the Pantomime component count task model. The component count model Eq. 5.1 met the engineering model benchmark with close to 80% prediction accuracy against new gestures and, being an *a priori* model, allowing well under the 20% effort of an experimental study desired. The chapter also explored the gesture data sets, leading to ways in which future gesture modeling experiments might be improved.

$$MT = (440l + 300c)\ \text{milliseconds} \tag{5.1}$$

5.2 Work Limitations

The Pantomime representation and task model are effective at describing and predicting gesture performance, extending the Keystroke Level Model and the modeling domain appropriately. However, there are currently limitations on both the model itself and upon the modeling of gestures in general that should be noted.

The experiments resulted in a task model that predicts gesture performance, but the number of gestures the model has been tested upon is limited; there were six gestures, many similar to one another, which led to the values given in the component count model. While performance on novel gestures suggests that the component count model would be similarly accurate on the hundreds of possible free-space movement-based gestures that could be generated, more data may be needed to demonstrate reliability.

The experiments that informed the model were also using a single device, the Microsoft Kinect. While there is no physical contact with a camera-based gesture recognition system, as the data from the Leap Motion shows gesture performance is affected by the chosen device. While the general observations concerning performance of components that make up a gesture are maintained over both devices' studies, comparative studies using the Kinect, the Leap Motion and perhaps even smaller- (or larger-)scaled gesture recognition systems may be needed to demonstrate universality of the model.

There are few individuals currently with extensive experience with gesture control systems; while some task-specific practice eliminates some inexperience-related errors and hesitancy, analysis of the data suggests that the values could be more consistent given subjects that, like in some of the KLM experiments mentioned by John and Kieras [JK96a], were already experts in the use of gestures. How well the Pantomime task model predicts truly 'expert' gesture performance cannot

yet be tested.

Although the Pantomime representation is discussed as including the potential for two-handed gesturing, the task model presently covers only single-hand symbolic gestures among all the possible free-space movement-based gestures. Two-handed input and gestures have not been modeled; the impact of the second hand upon the movement time of the first may be negligible or may be significant—and may depend on the relationship between the movement of each hand. Similarly, the impact of hand pose, particularly in the changing of hand pose during a movement, has also not been modeled.

Gesturing itself is a task at a more complex level than models such as Fitts' Law or the typing or homing operators of the KLM; multiple sequential movements with no fixed target lead to individual judgment during the task - where the spirit of Hick's Law (the time to select an option increases with the number of available options) [Hic52] may apply as much as that of Fitts' Law (the time to perform an action increases as the distance increases or the size of the target decreases). There may be a limit to the accuracy of any *a priori* task model of gesturing.

5.3 Future Experimental Work

With the establishment of a simple task model of gesture, there are a wide range of experimental work that can be performed. The previous chapter, in discussion, noted several paths for future experiments. Further data can continue to refine and further validate the model values discovered in the second study against a selection of different gestures, while targeted experiments could further examine the use of the component count model, length and duration to homogenize gesture studies in a fashion similar to the "effective width" used in Fitts' Law.

The performance of the "retrace" gesture in the second study suggests some gesture characteristic that acts to 'discount' performance. A study contrasting several similar gestures to counterparts that do not contain a similar component relationship may isolate this factor, which in turn could be applied to gesture interfaces for more efficient usage. While the focus of Pantomime is a task model

that can make predictions without data, further study of the relationship of length to duration in gestures in the context of a component-based model may allow for a single model to function at all levels of gesturing, from full-arm movement to that of a single finger.

Pantomime could be paired with a similar representation of hand pose, allowing a task model that incorporates both static and dynamic forms of gesturing to be generated and tested. Other current limitations of the framework, such as a lack of two-handed gesturing in the task model, could also be addressed. While the gesture performance itself is modeled, performance *around* the gesture, such as the transition between gestures, is not. Gesture performance in multi-modal systems could be addressed either through the impact of the alternate modalities (such as speech) upon gesture performance or as part of a larger modeling system such as GOMS.

The component-based approach may also be applied to surface-bound gestures such as described in models such as the minimal straight-line representation [Iso01] or Corners, Lines and Curves [CZ07]. With a model suitable for surface-bound performance, Pantomime's performance could be compared directly to alternative models to determine commonalities or differences between free-space and surface-bound gesturing.

5.4 Wrapping Up

The goal of this dissertation was to demonstrate that a task model can represent a gesture *a priori* and be able to make predictions of duration well enough to be used either alone or as part of a modeling framework such as the Keystroke-Level Model. The component-based approach of the Pantomime representation and task model has been developed to fulfill this goal and, in this respect, the focus of this work has been achieved - Pantomime's task model is accurate enough to be used as an engineering model. Moreover, the simple structure of the model allows it to be implemented much as existing KLM operators are - *a priori*, even by hand and rapidly deployed.

The experiments discussed in Chapter 4, together with the discussion, offer a template for future gesture performance research. Statistically, the root mean square error acts as a comparator between

prospective models to establish fit, and the prediction error is used both as an overall average and per-gesture type to determine if a model's performance renders it suitable as an engineering model. The inclusion of not just gesture duration but gesture length is likewise valuable if comparisons between data sets are to occur or to classify gestures by speed category as Wobbrock [Wob07] does.

This chapter has outlined a few ways in which this work can be advanced - refinement of the model, expansion to other gesture sub-domains and gestures in the context of a larger multi-modal interface among them.

BIBLIOGRAPHY

[AZ97] Accot, J. & Zhai, S. "Beyond Fitts' Law: Models for Trajectory-based HCI Tasks". *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. CHI '97. ACM, 1997, pp. 295–302.

[All83] Allen, J. F. "Maintaining Knowledge About Temporal Intervals". *Commun. ACM* **26**.11 (1983), pp. 832–843.

[Alo09] Alon, J. et al. "A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**.9 (2009), pp. 1685–1699.

[Ame02] Amento, B. et al. "The Sound of One Hand: A Wrist-mounted Bio-acoustic Fingertip Gesture Interface". *CHI '02 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '02. ACM, 2002, pp. 724–725.

[Bai08] Bailly, G. et al. "Flower Menus: A New Type of Marking Menu with Large Menu Breadth, Within Groups and Efficient Expert Mode Memorization". *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '08. ACM, 2008, pp. 15–22.

[BBL93] Baudel, T. & Beaudouin-Lafon, M. "Charade: Remote Control of Objects Using Free-hand Gestures". *Commun. ACM* **36**.7 (1993), pp. 28–35.

[Bil02] Billinghurst, M. *Pragmatics of Gesture Input*. Ed. by Buxton, W. & Buxton, J. L. Cambridge University Press, 2002. Chap. Gesture Based Interaction.

[Bol80] Bolt, R. A. "Put-that-there: Voice and Gesture at the Graphics Interface". *SIGGRAPH Comput. Graph.* **14**.3 (1980), pp. 262–270.

[CB03] Cao, X. & Balakrishnan, R. "VisionWand: Interaction Techniques for Large Displays Using a Passive Wand Tracked in 3D". *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. UIST '03. ACM, 2003, pp. 173–182.

[CB05] Cao, X. & Balakrishnan, R. "Evaluation of an On-line Adaptive Gesture Interface with Command Prediction". *Proceedings of Graphics Interface 2005*. GI '05. Canadian Human-Computer Communications Society, 2005, pp. 187–194.

[CZ07] Cao, X. & Zhai, S. "Modeling Human Performance of Pen Stroke Gestures". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. ACM, 2007, pp. 1495–1504.

[Cao08] Cao, X. et al. "Peephole Pointing: Modeling Acquisition of Dynamically Revealed Targets". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. ACM, 2008, pp. 1699–1708.

[CM88] Card, S. K. & Moran, T. P. “A History of Personal Workstations”. Ed. by Goldberg, A. ACM, 1988. Chap. User Technology: From Pointing to Pondering, pp. 489–526.

[Car78] Card, S. K. et al. “Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT”. *Ergonomics* **21** (1978), pp. 601–613.

[Car80] Card, S. K. et al. “The Keystroke-level Model for User Performance Time with Interactive Systems”. *Commun. ACM* **23**.7 (1980), pp. 396–410.

[Car83] Card, S. K. et al. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., 1983.

[Car15] Carvalho, D. et al. “Performance Evaluation of Gesture-based Interaction Between Different Age Groups Using Fitts’ Law”. *Proceedings of the XVI International Conference on Human Computer Interaction*. Interacción ’15. ACM, 2015, 5:1–5:7.

[Cha14] Chakraborty, A. et al. “CAPTIVE: A Cube with Augmented Physical Tools”. *CHI ’14 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’14. ACM, 2014, pp. 1315–1320.

[Cha07] Chan, L.-W. et al. “Gesture-based Interaction for a Magic Crystal Ball”. *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*. VRST ’07. ACM, 2007, pp. 157–164.

[Dha06] Dhawale, P. et al. “Bare-hand 3D Gesture Input to Interactive Systems”. *Proceedings of the 7th ACM SIGCHI New Zealand Chapter’s International Conference on Computer-human Interaction: Design Centered HCI*. CHINZ ’06. ACM, 2006, pp. 25–32.

[EB12] Echtler, F. & Butz, A. “GISpL: Gestures Made Easy”. *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. TEI ’12. ACM, 2012, pp. 233–240.

[Ech10] Echtler, F. et al. “Towards a Unified Gesture Description Language”. *Proceedings of the 13th International Conference on Humans and Computers*. HC ’10. University of Aizu Press, 2010, pp. 177–182.

[Fan07] Fang, Y. et al. “A Real-Time Hand Gesture Recognition Method”. *2007 IEEE International Conference on Multimedia and Expo*. IEEE, 2007, pp. 995–998.

[Fit54] Fitts, P. M. “The information capacity of the human motor system in controlling the amplitude of movement”. *Journal of Experimental Psychology: General* **47**.6 (1954), pp. 381–391.

[Gao04] Gao, W. et al. “A Chinese sign language recognition system based on SOFM/SRN/HMM”. *Pattern Recognition* **37**.12 (2004), pp. 2389 –2402.

[Gib01] Gibet, S. et al. “High-level Specification and Animation of Communicative Gestures”. *Journal of Visual Languages & Computing* **12**.6 (2001), pp. 657 –687.

[Glo12] Glomb, P. et al. “Choosing and Modeling the Hand Gesture Database for a Natural User Interface”. *Proceedings of the 9th International Conference on Gesture and Sign Language in Human-Computer Interaction and Embodied Communication*. GW’11. Springer-Verlag, 2012, pp. 24–35.

[GM99] Goldin-Meadow, S. “The role of gesture in communication and thinking”. *Trends in Cognitive Sciences* **3**.11 (1999), pp. 419 –429.

[Gra92] Gray, W. D. et al. “The Precis of Project Ernestine or an Overview of a Validation of GOMS”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’92. ACM, 1992, pp. 307–312.

[GW00] Guimbretiére, F. & Winograd, T. “FlowMenu: Combining Command, Text, and Data Entry”. *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*. UIST ’00. ACM, 2000, pp. 213–216.

[HG09] Haimson, C. & Grossman, J. “A GOMSL Analysis of Semi-automated Data Entry”. *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS ’09. ACM, 2009, pp. 61–66.

[HH09] Harrison, C. & Hudson, S. E. “Abracadabra: Wireless, High-precision, and Unpowered Finger Input for Very Small Mobile Devices”. *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*. UIST ’09. ACM, 2009, pp. 121–124.

[Hic52] Hick, W. E. “On the rate of gain of information”. *The Quarterly Journal of Experimental Psychology* **4** (1952), pp. 11–26.

[Hin02] Hinckley, K. et al. “Quantitative Analysis of Scrolling Techniques”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’02. ACM, 2002, pp. 65–72.

[Hol07] Holleis, P. et al. “Keystroke-level Model for Advanced Mobile Phone Interaction”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’07. ACM, 2007, pp. 1505–1514.

[Hol11] Holleis, P. et al. “A Revised Mobile KLM for Interaction with Multiple NFC-Tags”. *Human-Computer Interaction – INTERACT 2011: 13th IFIP TC 13 International Conference,*

Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV. Ed. by Campos, P. et al. Springer Berlin Heidelberg, 2011, pp. 204–221.

[Hou04] Hourcade, J. P. et al. "Differences in Pointing Task Performance Between Preschool Children and Adults Using Mice". *ACM Trans. Comput.-Hum. Interact.* **11**.4 (2004), pp. 357–386.

[Hud99] Hudson, S. E. et al. "A Tool for Creating Predictive Performance Models from User Interface Demonstrations". *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*. UIST '99. ACM, 1999, pp. 93–102.

[Iso01] Isokoski, P. "Model for Unistroke Writing Time". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. ACM, 2001, pp. 357–364.

[Joh96] John, B. E. "TYPIST: A Theory of Performance in Skilled Typing". *Human-Computer Interaction* **11**.4 (1996), pp. 321–355.

[JK94] John, B. E. & Kieras, D. E. *The GOMS Family of Analysis Techniques: Tools for Design and Evaluation*. Tech. rep. 1994.

[JK96a] John, B. E. & Kieras, D. E. "The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast". *ACM Trans. Comput.-Hum. Interact.* **3**.4 (1996), pp. 320–351.

[JK96b] John, B. E. & Kieras, D. E. "Using GOMS for User Interface Design and Evaluation: Which Technique?" *ACM Trans. Comput.-Hum. Interact.* **3**.4 (1996), pp. 287–319.

[Joh88] John, B. E. "Contributions to engineering models of human-computer interaction.(volumes I and II)". PhD thesis. Carnegie Mellon University, 1988.

[Jud14] Jude, A. et al. "An Evaluation of Touchless Hand Gestural Interaction for Pointing Tasks with Preferred and Non-preferred Hands". *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*. NordiCHI '14. Helsinki, Finland: ACM, 2014, pp. 668–676.

[Jud16] Jude, A. et al. "Reporting and Visualizing Fitts's Law: Dataset, Tools and Methodologies". *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '16. Santa Clara, California, USA: ACM, 2016, pp. 2519–2525.

[KB95] Kabbash, P. & Buxton, W. A. S. "The Prince Technique: Fitts' Law and Selection Using Area Cursors". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 273–279.

[Kal03] Kallio, S. et al. “Online gesture recognition system for mobile interaction”. *Systems, Man and Cybernetics, 2003. IEEE International Conference on*. Vol. 3. 2003, 2070–2076 vol.3.

[Kel06] Kela, J. et al. “Accelerometer-based Gesture Control for a Design Environment”. *Personal Ubiquitous Comput.* **10**.5 (2006), pp. 285–299.

[Kie94] Kieras, D. “GOMS Modeling of User Interfaces Using NGOMSL”. *Conference Companion on Human Factors in Computing Systems*. CHI ’94. ACM, 1994, pp. 371–372.

[Kie95] Kieras, D. E. et al. “GLEAN: A Computer-based Tool for Rapid GOMS Model Usability Evaluation of User Interface Designs”. *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST ’95. ACM, 1995, pp. 91–100.

[Kim07] Kim, D. et al. “Simultaneous gesture segmentation and recognition based on forward spotting accumulative {HMMs}”. *Pattern Recognition* **40**.11 (2007), pp. 3012 –3026.

[KB09] Kratz, S. & Ballagas, R. “Unravelling Seams: Improving Mobile Gesture Recognition with Visual Feedback Techniques”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. ACM, 2009, pp. 937–940.

[KZ04] Kristensson, P.-O. & Zhai, S. “SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers”. *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. UIST ’04. ACM, 2004, pp. 43–52.

[KB93] Kurtenbach, G. & Buxton, W. “The Limits of Expert Performance Using Hierarchic Marking Menus”. *Proceedings of the INTERACT ’93 and CHI ’93 Conference on Human Factors in Computing Systems*. CHI ’93. Amsterdam, The Netherlands: ACM, 1993, pp. 482–487.

[Lan93] Lane, D. M. et al. “Predicting the Skilled Use of Hierarchical Menus With the Keystroke-Level Model”. *Human-Computer Interaction* **8**.2 (1993), pp. 185–192.

[LoP00] LoPresti, E. et al. “Neck Range of Motion and Use of Computer Head Controls”. *Proceedings of the Fourth International ACM Conference on Assistive Technologies*. Assets ’00. Arlington, Virginia, USA: ACM, 2000, pp. 121–128.

[LJ05] Luo, L. & John, B. E. “Predicting Task Execution Time on Handheld Devices Using the Keystroke-level Model”. *CHI ’05 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’05. Portland, OR, USA: ACM, 2005, pp. 1605–1608.

[Mac92] MacKenzie, I. S. “Fitts’ Law As a Research and Design Tool in Human-computer Interaction”. *Hum.-Comput. Interact.* **7**.1 (1992), pp. 91–139.

[MB92] MacKenzie, I. S. & Buxton, W. “Extending Fitts’ Law to Two-dimensional Tasks”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’92. Monterey, California, USA: ACM, 1992, pp. 219–226.

[MZ99] MacKenzie, I. S. & Zhang, S. X. “The Design and Evaluation of a High-performance Soft Keyboard”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’99. Pittsburgh, Pennsylvania, USA: ACM, 1999, pp. 25–31.

[Mac91] MacKenzie, I. S. et al. “A Comparison of Input Devices in Element Pointing and Dragging Tasks”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’91. New Orleans, Louisiana, USA: ACM, 1991, pp. 161–166.

[MA07] Mitra, S. & Acharya, T. “Gesture Recognition: A Survey”. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **37**.3 (2007), pp. 311–324.

[Mot01] Mottet, D. et al. “Two-handed performance of a rhythmical Fitts task by individuals and dyads.” *Journal of Experimental Psychology: Human Perception and Performance* **27**.6 (2001), pp. 1275 –1286.

[Myu04] Myung, R. “Keystroke-level analysis of Korean text entry methods on mobile phones”. *International Journal of Human-Computer Studies* **60**.5-6 (2004). {HCI} Issues in Mobile Computing, pp. 545–563.

[Oka02] Oka, K. et al. “Real-time fingertip tracking and gesture recognition”. *IEEE Computer Graphics and Applications* **22**.6 (2002), pp. 64–71.

[Pan10] Pan, G. et al. “GeeAir: A Universal Multimodal Remote Control Device for Home Appliances”. *Personal Ubiquitous Comput.* **14**.8 (2010), pp. 723–735.

[Par08] Park, I.-K. et al. “An Implementation of an FPGA-based Embedded Gesture Recognizer Using a Data Glove”. *Proceedings of the 2Nd International Conference on Ubiquitous Information Management and Communication*. ICUIMC ’08. Suwon, Korea: ACM, 2008, pp. 496–500.

[Pas06] Pastel, R. “Measuring the Difficulty of Steering Through Corners”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’06. Montreal, Quebec, Canada: ACM, 2006, pp. 1087–1096.

[Pat97] Paterno, F. et al. “Engineering task models”. *Engineering of Complex Computer Systems, 1997. Proceedings., Third IEEE International Conference on*. IEEE, 1997, pp. 69–76.

[Pin13] Pino, A. et al. “Using Kinect for 2D and 3D Pointing Tasks: Performance Evaluation”. *Proceedings of the 15th International Conference on Human-Computer Interaction:*

Interaction Modalities and Techniques - Volume Part IV. HCI'13. Las Vegas, NV: Springer-Verlag, 2013, pp. 358–367.

[Que96] Quek, F. K. H. "Unencumbered gestural interaction". *IEEE MultiMedia* **3**.4 (1996), pp. 36–47.

[Que02] Quek, F. et al. "Multimodal Human Discourse: Gesture and Speech". *ACM Trans. Comput.-Hum. Interact.* **9**.3 (2002), pp. 171–193.

[Ram03] Ramamoorthy, A. et al. "Recognition of dynamic hand gestures". *Pattern Recognition* **36**.9 (2003). Kernel and Subspace Methods for Computer Vision, pp. 2069 –2081.

[RON89] Reitman Olson, J. & Nilsen, E. "Analysis of the Cognition Involved in Spreadsheet Software Interaction (Abstract Only)". *SIGCHI Bull.* **21**.1 (1989), pp. 123–.

[ROO90] Reitman Olson, J. & Olson, G. M. "The Growth of Cognitive Modeling in Human-Computer Interaction Since GOMS". *Human-Computer Interaction* **5**.2-3 (1990), pp. 221–265.

[Saf08] Saffer, D. *Designing gestural interfaces: Touchscreens and interactive devices*. O'Reilly Media, Inc., 2008.

[SW13] Sambrooks, L. & Wilkinson, B. "Comparison of Gestural, Touch, and Mouse Interaction with Fitts' Law". *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*. OzCHI '13. Adelaide, Australia: ACM, 2013, pp. 119–122.

[SW49] Shannon, C. E. & Weaver, W. *The Mathematical Theory of Communication*. IL, USA: Urbana, University of Illinois Press, 1949.

[ST07] Shi, Y. & Tsui, T. "An FPGA-Based Smart Camera for Gesture Recognition in HCI Applications". *Computer Vision – ACCV 2007: 8th Asian Conference on Computer Vision, Tokyo, Japan, November 18-22, 2007, Proceedings, Part I*. Ed. by Yagi, Y. et al. Springer Berlin Heidelberg, 2007, pp. 718–727.

[Sil00] Silfverberg, M. et al. "Predicting Text Entry Speed on Mobile Phones". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands: ACM, 2000, pp. 9–16.

[Ste05] Stein, R. B. et al. "Neuronal variability: noise or part of the signal?" *Nature Reviews.Neuroscience* **6**.5 (2005), pp. 389–97.

[SZ94] Sturman, D. J. & Zeltzer, D. "A survey of glove-based input". *IEEE Computer Graphics and Applications* **14**.1 (1994), pp. 30–39.

[Suk10] Suk, H.-I. et al. “Hand gesture recognition based on dynamic Bayesian network framework”. *Pattern Recognition* **43**.9 (2010), pp. 3059 –3072.

[TJ06] Teo, L. & John, B. E. “Comparisons of Keystroke-level Model Predictions to Observed Data”. *CHI ’06 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’06. Montreal, Quebec, Canada: ACM, 2006, pp. 1421–1426.

[TK08] Tudoreanu, M. E. & Kraemer, E. “A Study of the Performance of Steering Tasks Under Spatial Transformation of Input”. *Proceedings of the 46th Annual Southeast Regional Conference on XX*. ACM-SE 46. Auburn, Alabama: ACM, 2008, pp. 340–345.

[VZ13] Vatavu, R.-D. & Zaiti, I. A. “Automatic recognition of object size and shape via user-dependent measurements of the grasping hand”. *International Journal of Human-Computer Studies* **71**.5 (2013), pp. 590 –607.

[Vel15] Velloso, E. et al. “The Feet in Human–Computer Interaction: A Survey of Foot-Based Interaction”. *ACM Comput. Surv.* **48**.2 (2015), 21:1–21:35.

[Wel60] Welford, A. T. “The measurement of sensory-motor performance: survey and reappraisal of twelve years’ progress”. *Ergonomics* **3**.3 (1960), pp. 189–230.

[Wel69] Welford, A. et al. “Speed and accuracy of movement and their changes with age”. *Acta Psychologica* **30** (1969), pp. 3 –15.

[Wex95] Wexelblat, A. “An Approach to Natural Gesture in Virtual Environments”. *ACM Trans. Comput.-Hum. Interact.* **2**.3 (1995), pp. 179–200.

[Wil09] Willems, D. et al. “Iconic and multi-stroke gesture recognition”. *Pattern Recognition* **42**.12 (2009). New Frontiers in Handwriting Recognition, pp. 3303 –3312.

[Wil96] Wilson, A. D. et al. “Recovering the temporal structure of natural gesture”. *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*. 1996, pp. 66–71.

[WG08] Wobbrock, J. O. & Gajos, K. Z. “Goal Crossing with Mice and Trackballs for People with Motor Impairments: Performance, Submovements, and Design Directions”. *ACM Trans. Access. Comput.* **1**.1 (2008), 4:1–4:37.

[Wob07] Wobbrock, J. O. et al. “Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes”. *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST ’07. ACM, 2007, pp. 159–168.

[Wob09] Wobbrock, J. O. et al. “User-defined Gestures for Surface Computing”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. ACM, 2009, pp. 1083–1092.

[Yoo01] Yoon, H.-S. et al. “Hand gesture recognition using combined features of location, angle and velocity”. *Pattern Recognition* **34**.7 (2001), pp. 1491 –1501.

[Zha09] Zhang, X. et al. “Hand Gesture Recognition and Virtual Game Control Based on 3D Accelerometer and EMG Sensors”. *Proceedings of the 14th International Conference on Intelligent User Interfaces*. IUI ’09. ACM, 2009, pp. 401–406.

[Zim87] Zimmerman, T. G. et al. “A Hand Gesture Interface Device”. *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*. CHI ’87. ACM, 1987, pp. 189–192.