Pattern Marks: A Stylus Graphical User Interface that takes Advantage of the Redundancy in Languages

Per Ola Kristensson

Department of Computer and Information Science

Linköpings universitet, Sweden

Advisor: Shumin Zhai

Dynamics & Text Entry Workshop 2006, Glasgow University

Who am I?

- PhD student about to graduate this year
- Background in cognitive and computer science
- Parts of the work presented here was carried out when I was working at IBM Almaden Research Center, San Jose, California, USA (2003-2005)

Talk overview

- Automatic stylus keyboard typing correction
- Shorthand writing on a stylus keyboard
- Pattern marks – concept and design implications

- … and lots of **demos**

Background: Pen computing

- Great premise, but many failures
- Text entry and command input is slow and error-prone
- Commercial pen UIs are micro-versions of the desktop GUI
- Can research help?

Text entry on mobile computers

- How do we write text efficiently “off-the-desktop”
- Explosion of mobile computers – smart phones, PDAs, Tablet PCs, handheld video game consoles
- Can we achieve QWERTY touch-typing speed?

Modeling stylus keyboard performance using the laws of action

- Fitts' law – speed-accuracy trade-off in pointing

$$T = a + bID$$

$$ID = \log_2\left(\frac{D+W}{W}\right)$$

- Crossing law

[Fitts 1954, Accot and Zhai 2002]

How to “break” Fitts’ law

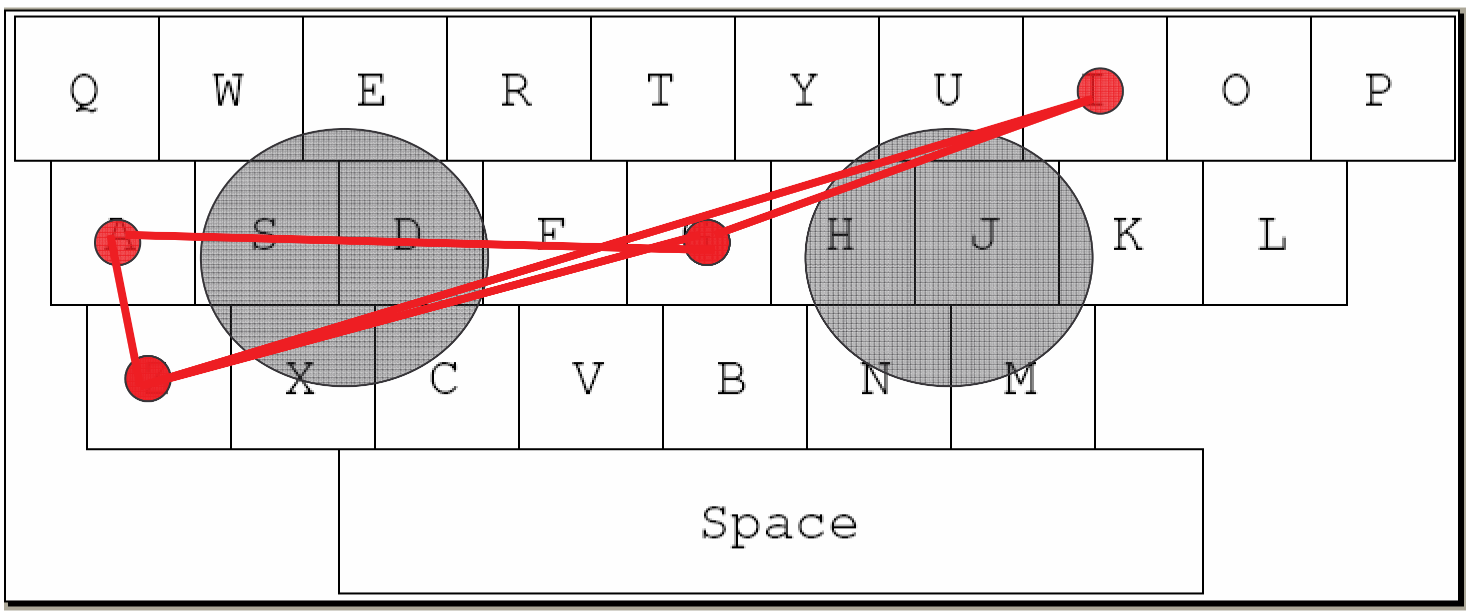
- D/W relationship in Fitts’ law

$$\frac{D}{W}$$

- Break *D* – minimize the distance the pen travels
- Break *W* – maximize the target size

The QWERTY stylus keyboard

- "Obvious" approach – transplanting QWERTY to the pen user interface



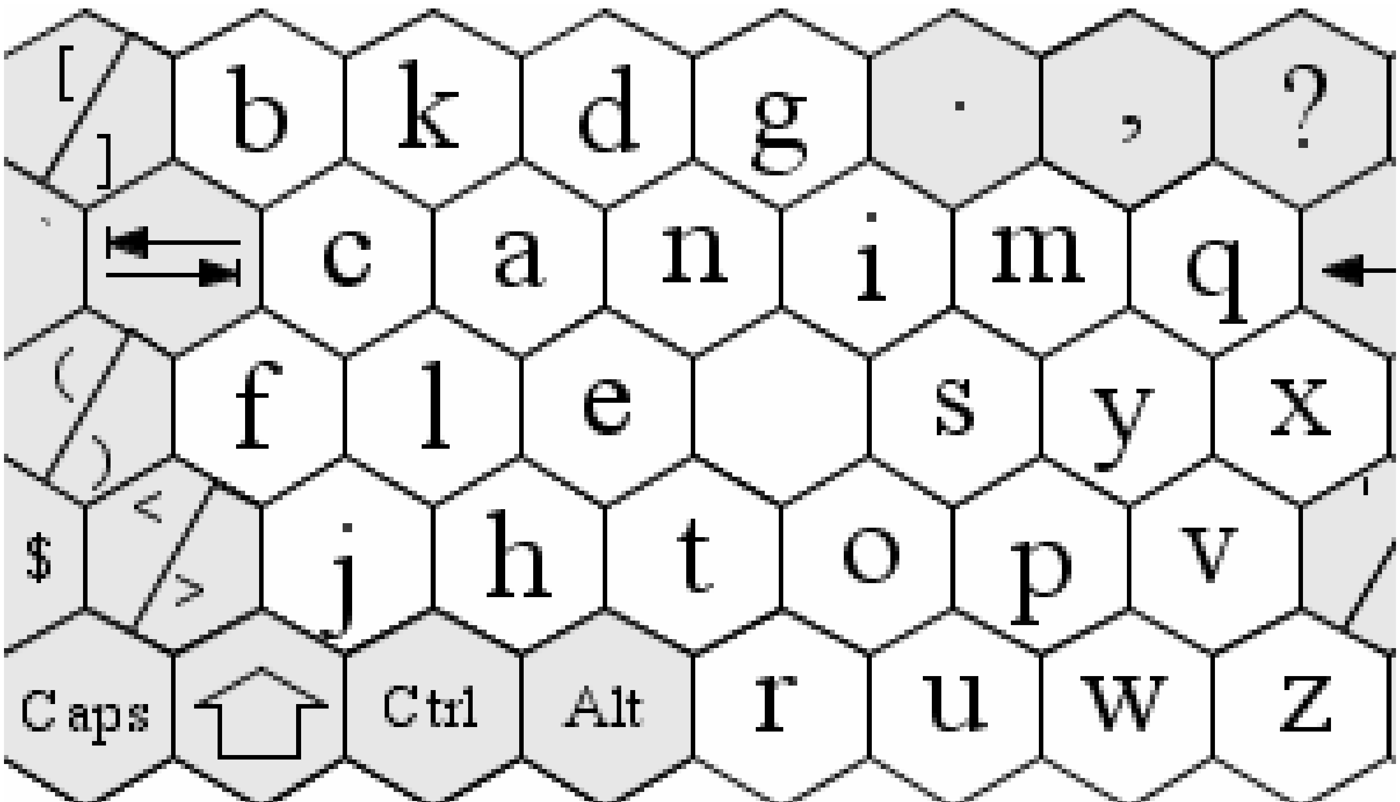
Example: Breaking *D*

- The distance between frequently related keys should be minimized
- A model of stylus keyboard performance:
    1. Fitts' law
    2. Digraph statistics (the probability that one key is followed by another)
- Using the model compute optimal configuration

[Getschow et al. 1986, Lewis et al. 1992]

Example: ATOMIK

- Optimized by a Fitts' law – digraph model using simulated annealing [Zhai et al. 2002]



Elastic Stylus Keyboard

Breaking the Fitts' Law W constraint

[Kristensson and Zhai 2005]

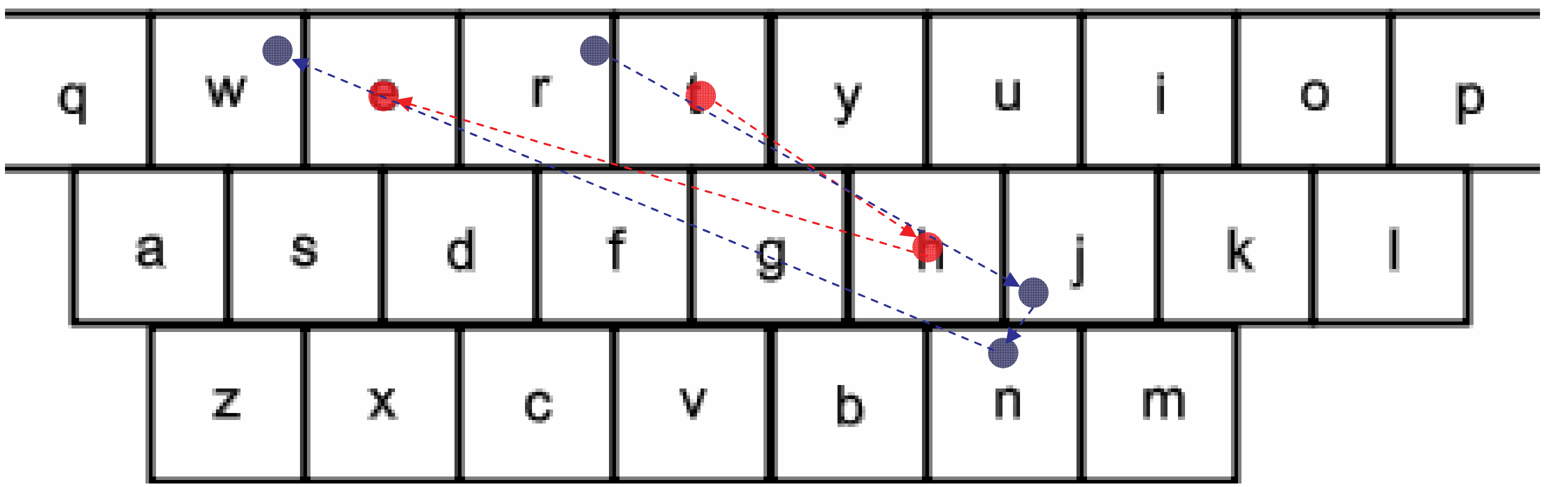
Problems with stylus keyboards

1. Error prone
   - Unlike physical keyboards, stylus keyboards lack tactile sensation feedback
   - Off by one pixel results in an error
2. Bounded by the Fitts' law accuracy trade-off
   - Trying to be faster than what Fitts' law predicts results in more errors

Two observations

1. Not all key combinations on a stylus keyboard are likely
   - A lexicon defines legal combinations
2. Stylus taps are “continuous” variables
   - Stylus taps form high resolution *patterns*
   - Words in the lexicon form geometrical patterns
   - Using pattern matching we can identify the user’s input

Example



t h e

r j n w → the

Elastic Stylus Keyboard (ESK)

- Pen-gesture as delimiter
- Edit-distance generalized to comparing point sequences instead of strings
- Handles erroneous insertions and deletions
- Indexing to allow efficient computation, despite quadratic complexity of the matching algorithm
- Can search 57K lexicon in real time on a 1 GHz Tablet PC

ESK video demonstration

Video (1)

Can an ESK “break” Fitts’ law?

- Regular QWERTY stylus keyboarding has an average estimated expert speed of 34.2 wpm
- Since we relax or “break” the *W* constraint in Fitts’ law (the radius of the key), can we do better?

Testing phrase (57K lexicon, no errors allowed)	User 1	User 2
the quick brown fox jumps over the lazy dog	46.3	37.7
ask not what your country can do for you	45.4	40.1
intelligent user interfaces	51.3	51.8

Shorthand Writing on a Stylus Keyboard

Breaking the Crossing Law W constraint

[Kristensson 2002, Zhai and Kristensson 2003, Kristensson and Zhai 2004, Kristensson 2005]

SHARK shorthand – ShapeWriter

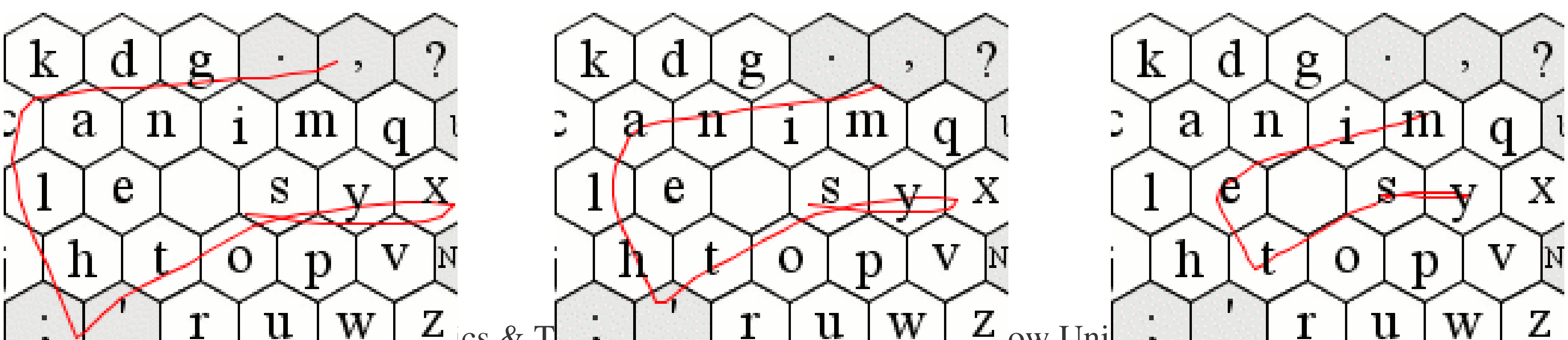
- SHARK = ***Sh***orthand-***A***ided ***R***apid ***K***eyboarding
- Typing on a stylus keyboard is a verbatim process
- Instead of tapping the letter keys of a word
- …gesture the patterns directly

Writing the word “system” as a shorthand gesture

Gradual transition from closed-loop visually tracing the keys to open-loop memory recall

Feedback

- Recognized word is drawn on the keyboard
- Presents ideal gesture on keyboard
- Morphing of user's pen trace towards the recognized sokgraph
  - The animation suggests to a user which parts of a gesture that are the farthest away from the ideal sokgraph



SHARK shorthand video demonstration

Video (2)

Advantages

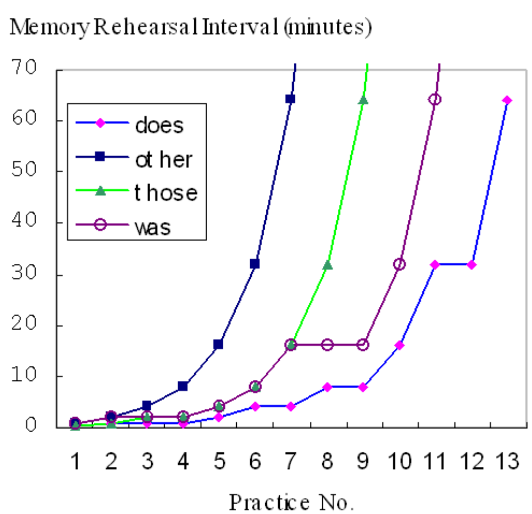
- Users can be productive while training: in-use learning
- The most frequently used words in a user's vocabulary get practiced the most
- Easy mode for novices (visual-guided)
- Fast mode for experts (memory recall)
- Transition from novice to expert is continuous
- Keyboard acts as a mnemonic device

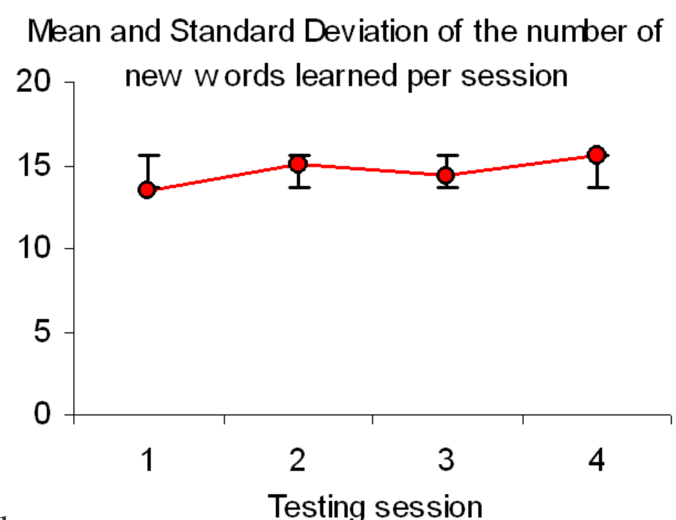
Empirical "records" (wpm) (**old**)

Testing phrase (8K lexicon, no errors allowed)	User A	User B
"The quick brown fox jumps over the lazy dog"	69.0	70.3
"Ask not what your country can do for you"	51.6	60.0
"East west north south"	74.4	72.9
"Up down left right"	74.1	85.6

Learning

- Can user's learn the sokgraphs?
- Expanding Rehearsal Interval (ERI) training
- Users can on average learn 15 sokgraphs per 45 minute training session





Related work

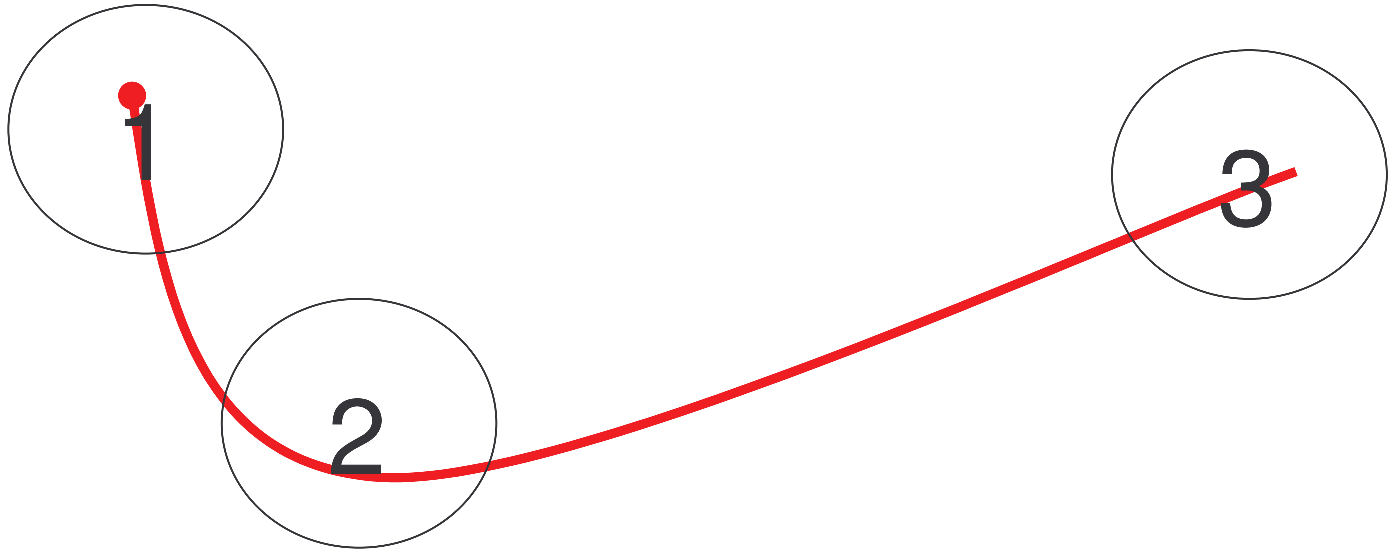
- Specialized alphabets
  - Unistrokes [Goldberg and Richardson 1993]
  - Graffiti
  - EdgeWrite [Wobbrock, Myers, Kemnbel, 2003]
- Wipe-activated keyboard [Montgomery 1982]
- Quikwriting [Perlin 1998], Cirrin [Mankoff and Abowd 1998]
- T-Cube [Venolia and Neiberg 1994]
- Dasher [Ward, Blackwell, MacKay 2000]
- Stylus keyboards [Lewis et al 1992 -, Mackenzie et al 1995 -]
- Marking menus [Kurtenbach and Buxton 1993]

Dimensions of a mobile text entry method

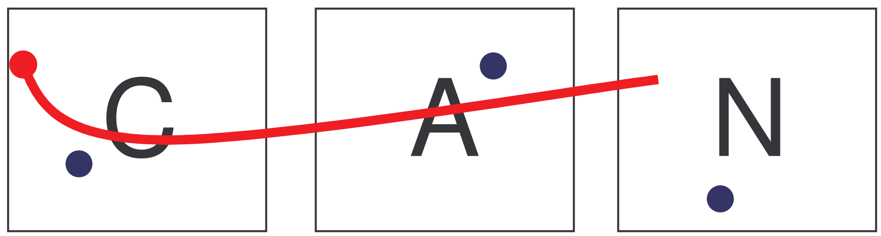
- High performance
- Ease of initial use
- Learning
- Comfortable
- Small form factor
- Fun and exciting

Pattern marks

- A *mark* is the technical term for pen-gesture suggested by Kurtenbach [1993] in his PhD thesis
- A *pattern mark* is a *mark* defined by connecting a series of nodes in a user interface to form a trajectory



Discrete and continuous



- Gesturing “can” vs. “an” (red mark)
- Tapping “can” vs. “an” (blue dots)

Usage of pattern marks

- Automatically correcting stylus keyboard typing
- Shorthand text writing on handhelds
- Enter textual commands
- Creating new interesting fluid user-interfaces

Summary of pattern marks

- Defined as the trajectory connecting a series of GUI nodes
- Allow a smooth learning transition from slow closed-loop visual tracing into fast open-loop pattern recall gesturing
- Primarily usage is fast text entry
- But other fluid-action applications are possible

Current challenges

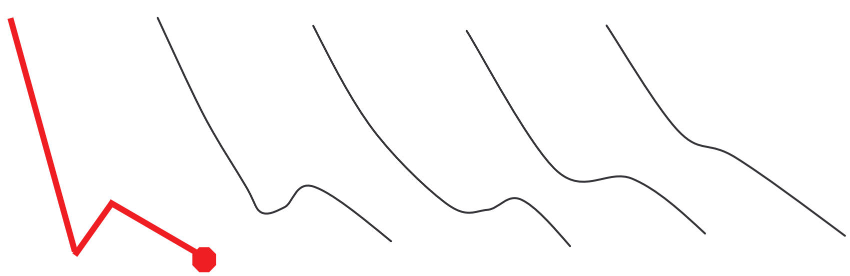
- Finding ways to create an efficient unambiguous pattern mark interface automatically
- Explore the speed-accuracy tradeoff between precise articulation of pattern marks vs. quick flicks
- Explore learning – which components of learning are important for long-term retention?
- Languages – how does the behavior of pattern mark recognition change as the lexicon size increases?

Exploring the "sloppiness space"

- Pattern recognition accuracy depends on how similar patterns are
- Larger lexicon = more confusable patterns?
- … but in fact, most confusable words in SHARK shorthand and ESK are very frequent, since frequent words tend to be smaller
- How does a user know how the limits of the system?

What is a recognition error?

- Speed accuracy trade-off
  - How fast people can do gestures?
  - How “sloppy” people get?
  - What is “reasonable”?
  - Users pushing the system beyond any chance of recognition



Speed and Learning

- Less information = faster articulation?
- Chunking
  - Tapping = sequentially entering small “chunks” of information
  - Gesturing = one chunk of information
- Motor memory, different muscles involved, more feedback when gesturing than tapping

Thank you!

Per Ola Kristensson

http://pokristensson.com

kristensson@acm.org

or use google