Sketch Recognition

AW2 Colloquium by Hauke Wittern 28.05.2009
Agenda

- Introduction
  - Vision
  - Definition of sketch recognition
- Research on sketch recognition
  - Today’s sketch recognition systems
  - Recent research topics
  - Using and recognizing sketches in software development
- Differentiation and relevance of related work
- Summary
Vision: Sketching of software models and prototypes on multitouch-tables

- Improve software engineering with multitouch-tables
- Freehand sketching of software models (UML etc.)
- Recognize sketches for
  - Prototyping
  - Simulation
Definition: Sketch Recognition

“Sketch recognition is the automated understanding of a drawn diagram, attempting to recognize the intent of the user while allowing the user to draw unconstrained diagrams.” (Tracy Hammond, [1])
Example video
ASSIST: A Shrewd Sketch Interpretation and Simulation Tool

Source: http://www.youtube.com/watch?v=d7eGypGOIOc
Benefits of digital sketches

- Natural way of devising and communicating ideas
- More effective than diagram tools with mouse etc.
- No constraints

- Digital sketches support rich features:
  - editing
  - searching
  - layout
  - archiving
Today’s sketch recognition systems
Research Groups with most influence on today’s Sketch Recognition systems

- Randall Davis, Massachusetts Institute of Technology
- Tracy Hammond, Texas A&M University (former MIT)
- Christine Alvarado, Harvey Mudd College (former MIT)
- Beryl Plimmer, University of Auckland
Single-Domain vs. Multi-Domain Sketch Recognition

- Single-Domain Sketch Recognition System: limited to the recognition of a *specific domain*
  - Tahuti (UML Class Diagrams) (by Hammond and Davis 2002)

- Generic Multi-Domain Sketch Recognition System: Can be trained to recognize sketches from *any domain*
  - SketchREAD (by Alvarado and Davis 2005)
  - InkKit (by Plimmer et al. 2007)
  - DSketch (Brieler and Minas 2008)
Different approaches to recognize Sketches:

- **Vision based (InkKit)**
  - recognize shapes by what they look like
  - Template matching

- **Geometry based (Tahuti, SketchREAD, DSketch)**
  - Break down shapes into primitives

- **Gesture based (SUMLOW ([4] 2003))**
  - recognize shapes by how they are drawn
  - Single stroke paths are compared to templates
Incremental vs. complete recognition

- **Complete recognition**
  - Recognition is performed
    - on demand (e.g. InkKit) or
    - after idle time (e.g. SUMLOW)
  - Restricts the user

- **Incremental / continuous recognition (e.g. SketchRead):**
  - Recognition is performed after a stroke is drawn
  - More difficult to implement
    - The system must know when to wait for more information
  - Long recognition time might interrupt the user
Shape Definition Language vs. training with examples

Two methods to define the shapes in a domain:

- **Shape Definition Language**
  - E.g. LADDER, SkG, Xml (DSketch, [3])

- **Training with examples**
  - E.g. InkKit
  - Easy and quick development
Tradeoff between natural input and constrained sketches

- Sketching on Paper is a natural activity with no constraints
- Sketching on an electronic device is potentially as natural as on paper

Problem:
- unconstrained sketches are difficult to recognize
- constrained sketching is not natural

Consequence:
- Allow ambiguities and resolve them
Resolving ambiguities and error recovery

- Calculate probability of ambiguous interpretations

- Approaches to resolve ambiguities
  - Use diagram wide probabilities (InkKit)
  - Use context, semantic and syntactic information (SketchREAD, [Brieler and Minas])
Recognizing text – Approach by Plimmer & Freeman

- **Idea:** separate recognition of text and drawings
- **Divider component**
  - Analyzes strokes
  - Assigns each stroke a probability if it is a letter
Recent Research Topics
Collaborative Input on a single sketch

- Whose stroke is whose?

- Advantages:
  - Improved sketch recognition
  - Collaboration:
    - Correct drawing history
    - Enables users to ask the contributor what he intended
    - Documentation: who contributed what part
Collaborative Input on a single sketch
- Approach by Eoff and Hammond [2]

- Differentiate users by tilt, pressure and speed

- Results:
  - The manner a user sketches is fairly consistent
  - Sketching mannerisms are distinct from user to user
  - Strokes can be classified with high accuracy
Collaborative Input on a single sketch
- Approach by Eoff and Hammond [2]

- Future Work:
  - Additionally use context information for identifying the creator of a stroke
  - User studies on more domains
Cross-Domain Sketch Recognition

Scenario:
- People are sketching different views of a system
- The sketches are from different domains (e.g. ER, Classdiagram, Gui, Petri net, …)
- The sketch recognition system interprets the sketches and its relationships correctly
- This enables code generation or simulation
Cross-Domain Sketch Recognition

- Implementation by Plimmer et al.
  - Relationships between diagrams are expressed by connecting the elements with rubberbands
Cross-Domain Sketch Recognition

Two possible implementations of the interpreter proposed by Plimmer et al.:

- Information carrier (shared object)
  - Easy to implement
  - Implementation order must be predefined
  - Interpretation of each diagram only once therefore the required information may be not available

- Communication protocol
  - Information on demand
  - Implicit coordination of the interpretation order
Applying Layout Algorithms

- Diagrams with optimized aesthetics are easier to read and understand (crossing edges, symmetry, …)

- Goal: transform the diagram while preserving its informal character
Existing layout algorithms are used for positioning elements.

Edge morphing is used to maintain the informal character of the edges:
- Compress or stretch
- Maintain normal incidence angles
- Maintain appearance intersections

Problem: overstretched edges are looking like a formal line.
Applying Layout Algorithms

- Open questions:
  - How to remain much of the users drawing style?
  - Evaluate the effects on the users when applying layout algorithms
  - Compare hand-drawn, optimized and formal graphs
Usability of Sketch Recognition Systems

- How to create an ergonomic sketching UI?
  - How to trigger recognition
  - How to give recognition feedback
  - How to inform users about recognition errors
Results by Alvarado et al. [5]:

- Recognition triggers should be user-triggered
- Recognition feedback should provide minimal clutter and transform users’ strokes as little as possible.
- Users are willing to correct errors after they are done drawing
- Errors must be predictable and/or understandable
Using and recognizing sketches in software development
Using and recognizing sketches in early phases of software development

- Modelling Tools:
  - Tahuti
  - SUMLOW
  - Calico

- Sketching is preferred over traditional UML tools
Using sketches in early phases of software development

- The kind of modeling in early design phases differs from modeling in later stages.
- In early design phases Models are
  - exploratory
  - incomplete
  - informal
  - of mixed abstraction levels
- According to Mangano et al. software engineering can therefore benefit from sketching tools.
Differentiation and relevance of related work
Differentiation from my work

- Primary objective of my work differs:
  - How to improve software engineering with multitouch tables?
  - Can sketch tools make modeling easier? How?

- Different hardware with different capabilities: multitouch table instead of whiteboard or tablet pc
  - No experiences with sketching on multitouch tables yet
  - How well does sketching on multitouch tables work?
  - What are the differences?
  - What do we have to take into consideration?
Relevance of related work

- Results of the research on sketch recognition engines is the basis for developing a sketch tool for multitouch tables.
- Results of the research on collaborative input, usability of sketch tools and layout algorithms must be taken into account to ensure the usability and suitability.
Relevance of related work

- Studies substantiate that sketching tools can improve software development. Thus my future research about sketching software models on multitouch tables is promising.
- The research on Cross-Domain Sketch Recognition is important for simulation and prototyping.
Summary
Summary

- Today's sketch recognition systems
  - can reliably recognize sketches from multi domains
  - are a basis for future sketch recognition systems

- Research currently focuses on
  - Improving reliability and usability
  - Collaboration

- Research on sketching in software development should be intensified
References

[1] Tracy Hammond,
*IUI’09 Workshop Summary: Sketch Recognition*, IUI ’09: Proceedings of the 13th international conference on Intelligent user interfaces, Sanibel Island, Florida, USA, 2009

[2] Brian D. Eoff and Tracy Hammond
*Who Dotted That ‘i’?: Context Free User Differentiation through Pressure and Tilt Pen Data.*

[3] Florian Brieler and Mark Minas,

[4] Qi Chen, John Grundy and John Hoskin,

[5] Paul Wais, Aaron Wolin and Christine Alvarado:
References


References


- HONG, Jason ; LANDAY, James ; L, James ; LONG, A. C. ; MANKOFF, Jennifer: *Sketch Recognizers from the End-User’s, the Designer’s, and the Programmer’s Perspective*. In: Sketch Understanding, Papers from the 2002 AAAI Spring Symposium, 2002, S. 73–77


References

- RUBINE, Dean: *Specifying gestures by example*. In: SIGGRAPH Comput. Graph. 25 (1991), Nr. 4, S. 329–337. – ISSN 0097-8930