Sketching Interface

Larry Rudolph
April 24, 2006

Motivation

- Natural Interface
    - touch screens + more
- Mass-market of h/w devices available
- Still lack of s/w & applications for it
- Similar and different from speech
    - how?

Comparison to speech

- Noisy environment -- can write but cannot talk
- Sketches useful after communication is over
- Can express things for which there are
  - too many words
  - no words
    - picture is worth at least 1,000 words
- Compare to GUI?
  - GUI provides fixed, visible vocabulary
    - sketching has invisible domain
  - Sketching like speech relies on user’s familiarity

Perceptual User Interface (PUI)

- Vision, speech, gestures are come to mind
    - Hey, don’t forget sketching
- Sketching modes
    - formal -- CAD tools
    - informal
        - ambiguity encourages the designer to explore more ideas in early stages
        - ignore details such as color, alignment, size
    - both?
        - do not to do both from scratch. when ready, fix up informal sketch

Differences in strategies

- Recognize vs. Don’t recognize
  - Similar to speech trade-offs
    - word recognition
    - sentence (concept) recognition
- When is recognition done?
  - stroke-based (while drawing)
  - image-based (after drawing is done)

Why no recognition

- actually, a spectrum of recognition
- quickly prototyping user interfaces
    - easier than using CAD tools
    - easier to brainstorm; be creative
- what to do with recognition errors?
    - separate window?
    - nothing: do not want to interfere?

Some projects

- Assist (Davis -- MIT / CSAIL)
    - more about this later
- Silk (Landay and Myers 2001)
    - Sketching Interfaces Like Krazy
    - more in next slildes
- some others not discussed
    - Burlap (Mankoff, Hudson 2000)
        - "mediation" used to correct recognition errors
    - DENIM (Lin, Newman 2000)
        - sketch tool for web designers
        - minimize the amount of recognition

Real-time Recognition

- Start with visual language
  - syntax in a declarative grammar
- consider multiple ambiguous interpretations
- use probability to disambiguate

How Silk Works

- As designer sketches, silk recognizes them
- Assumed to use touch-screen
- Add behavior through “storyboarding”
  - drawing arrows between related screens
- SILK transforms rough design to real one

Silk for Web Design

Image removed due to copyright restrictions.

- Designer sketches UI (for web)

SILK's Editing Gestures

- Recognizes gestures through Rubine's algorithm
  - statistical pattern-recognition trains classifiers
  - used only 15 to 20 examples for each primitive
- To classify gesture, compute its distinguishing f.
  - angles, point-to-point distances

Image removed due to copyright restrictions.

Lots of ambiguities

- Attachment
  - text to line
- Gap
  - omitted values
- Role
  - what is legend?
- Segmentation
  - single terminal represents multiple syntactic entities
- Occlusion

Very similar to Galaxy

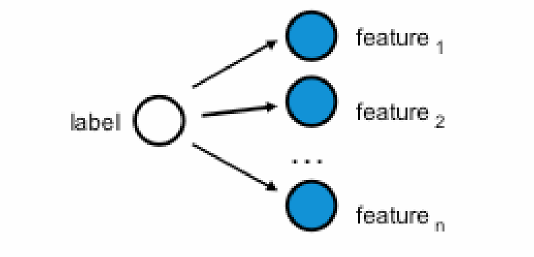
- Image removed due to copyright restrictions.

Visual Language Syntax

Image removed due to copyright restrictions.

Probability to the rescue

- To give a label to an element in drawing, base it one multiple features
- Use Bayes Theorem
  - prob this is the label given these features
    - probability given this label, would have these features
    - accounting for the likelihood of these features here



$$p(l \mid \{f_i\}) = \frac{p(l \wedge \{f_i\})}{p(\{f_i\})} = \frac{p(l)\prod_{\{f_i\}} p(f_i \mid l)}{\sum_{l} p(l)\prod_{\{f_i\}} p(f_i \mid l)}$$

Fixup the description

A parse in action

Domain dependent

- Like speech, good results require limiting of the domain
- Accuracy not very good a couple of years ago
- Must do more analysis in each domain

MIT Assist's Approach

- Interprets and understands as being drawn
    - sequence of strokes while system watches
- Very limited domain -- mechanical engineering
- general architecture to
    - represent ambiguities
    - add contextual knowledge to resolve ambiguities
    - low-level --- purely geometric
    - high-level -- domain specific

More detail

- delay commitment -- until body is done
- timing is crucial
  - too early, not enough information
  - too late, not useful to user
  - people tend to draw all of one object before moving to a new one
    - longer figure remains unchanged, more likely new strokes will not be added

General strategies

- Simpler is better
  - more specific is better
  - user feedback
  - single stroke rather than bunch of parts
- rule based system
  - not virturbi-like search

Early Processing

- Find line segments
  - so find the vertices
  - not so easy
    - wrong geometry
  - round corners

direction, curvature & speed

- Find places with
  - minimum speed
  - maximal curvature

Image removed due to copyright restrictions.

One is not enough

- Use average based filtering
  - divide into regions of max curvature and min speed
  - curvature & speed not uniform
  - different approx on each
- combined is best

Images removed due to copyright restrictions.

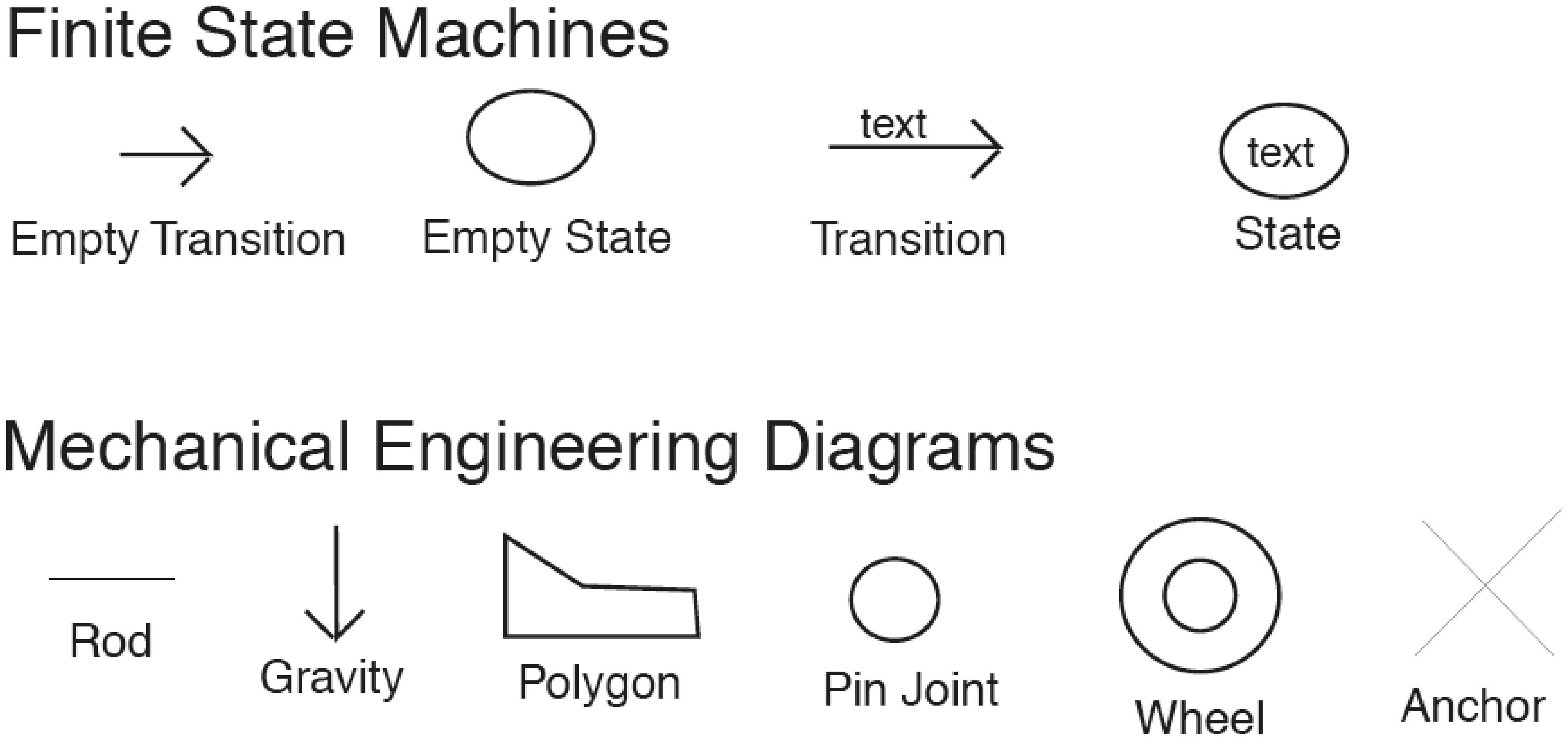
Description of shapes

- Built-in, basic shapes fine, but limited
- Want hierarchical, composible shapes
- One approach
  - constrained rule-based
    - 2-d is harder than 1-d, so constrains work better
  - language for describing shape

Domain Description in Ladder

Image removed due to copyright restrictions.

Some basic shapes that have been defined



Sketching Flowcharts

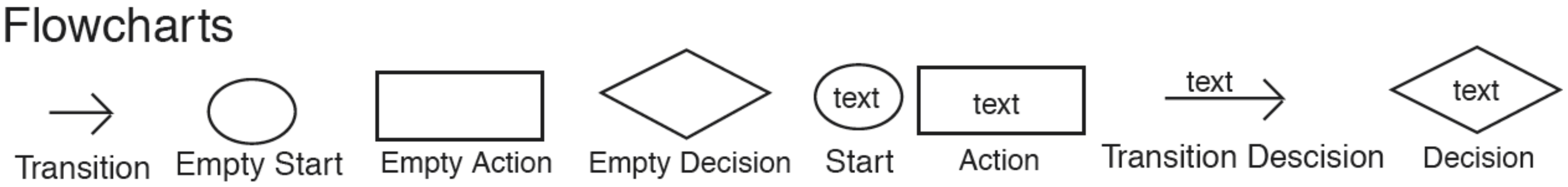
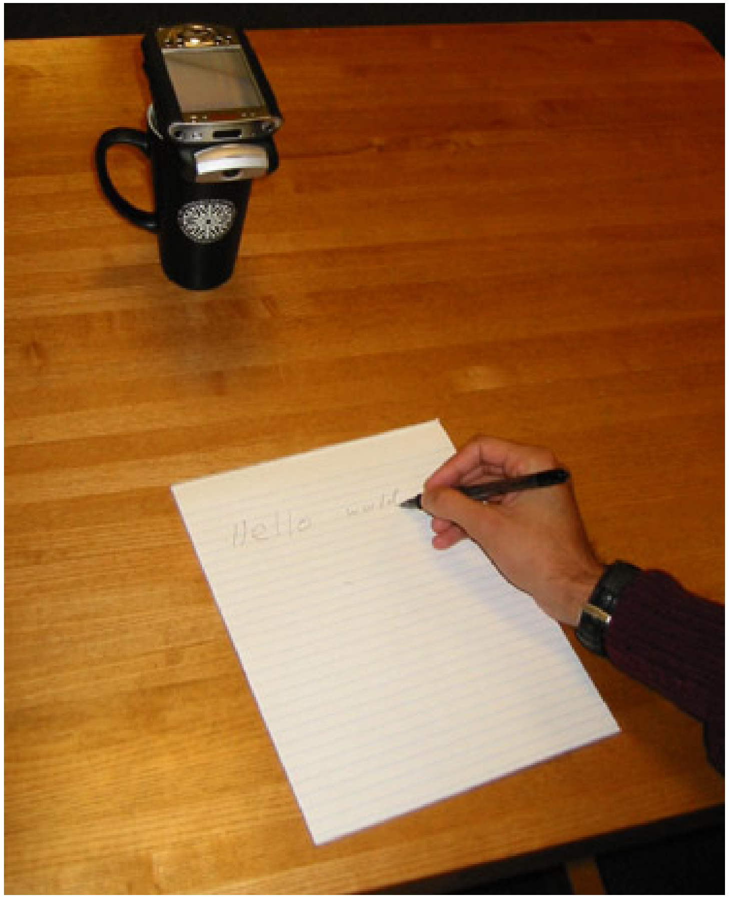


Image removed due to copyright restrictions.

PADCAM:
A human-centric sketching user interface



PADCAM:
A human-centric sketching user interface

- Use any pen
- Use any paper
- Draw as usual
- Strokes captured with timing info
  - as if done on touch screen
- If system crashes, still have notes

Xstroke

```
#       1 2 3
#       4 5 6
#       7 8 9

# The extents of the grid will be automatically inferred based on the
# bounding box of the input stroke. This makes xstroke robust to many
# stroke distortions including translation and independent scaling
# along the X and Y axes.
#
# For example, an intuitive stroke for the letter L might be:
#
#        Key L = 14789
 #          Key L = 147?89                (7?  means 7 is optional)
                                          [1 2]  means 1 or 2
```

What letter is this?

```
([12]*[45][78]|[12][45]+[78]?)?[78]*[4]*(1?[2][369]+|1[25][369]*)([369]+[25]+
8?[147]?[258]*[369]+|[25]*8?[147]+[258]+[369]*)([369]*[58][74]+|[369]+[58][74]*)
```

B