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

Massachusetts Institute of Technology

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)

CSAIL

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?

Massachusetts Institute of Technology

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

CSAIL

Real-time Recognition

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

Massachusetts Institute of Technology

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

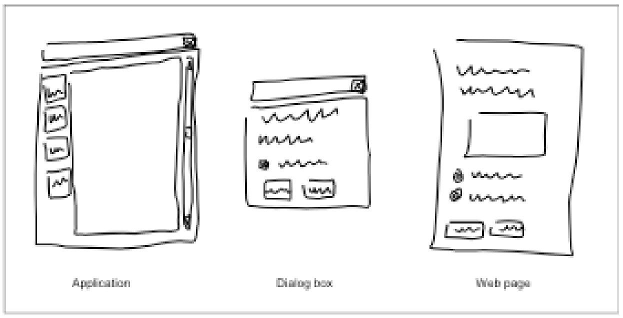


Figure 1. Hand-sketched SILK diagrams.

- Designer sketches UI (for web)

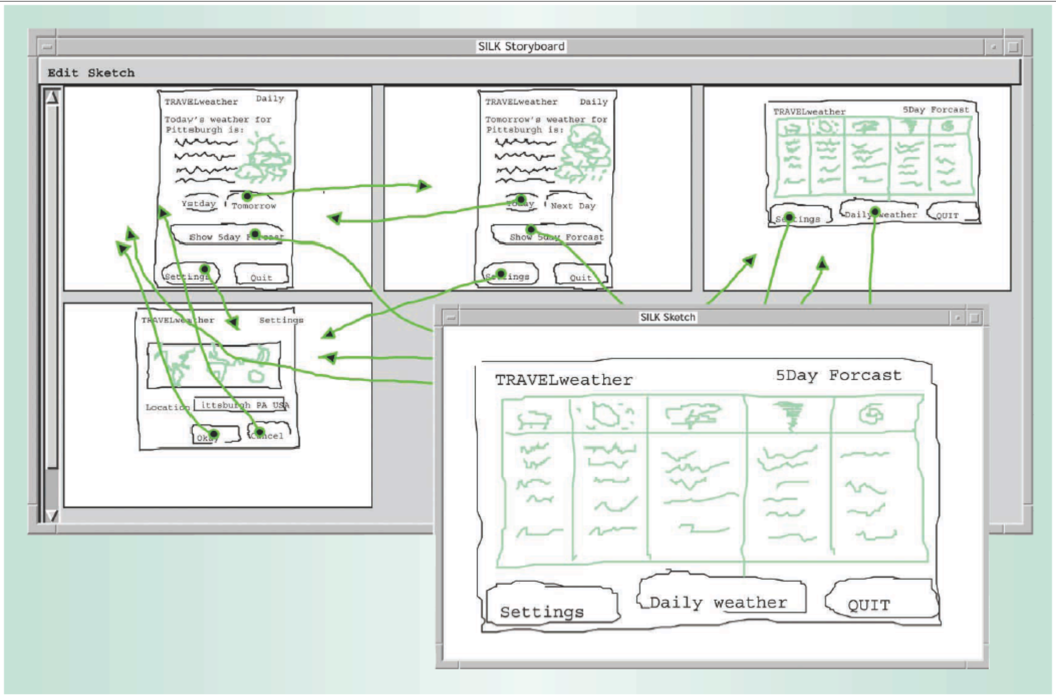


Figure 1. A SILK sketch (front) and storyboard (rear) for a weather application. An experienced user-interface designer created the sketch in about 30 minutes during a usability test. SILK recognizes common shapes such as ellipses and rectangles and provides feedback to designers about what it thinks they drew. The designer has sketched a screen that shows a five-day weather forecast (front). The designer has also created buttons below the forecast and drawn arrows from the first two (rear) so that when the user presses the buttons, he will go to the location settings dialog or see a weather report if he has set the right

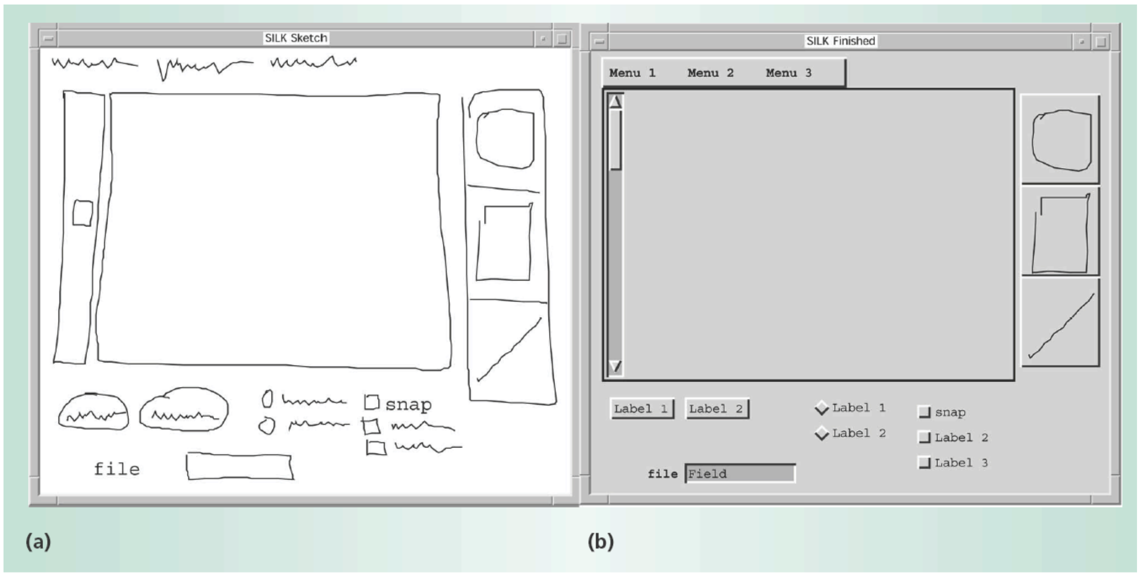


Figure 2. Interface widgets that SILK recognizes (a) during sketching and (b) in the transformed interface. From top to bottom, left to right: menu bar, scrolling window, palette, button, radio button, check box, and text field. SILK also recognizes vertical and horizontal sequences of some of these widgets as panels—for example, a check box panel, illustrated in the lower right. The sketched shapes in the palette on the far right were not transformed because they represent arbitrary bitmapped decorations. The designer can go back later and replace them with other images.

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

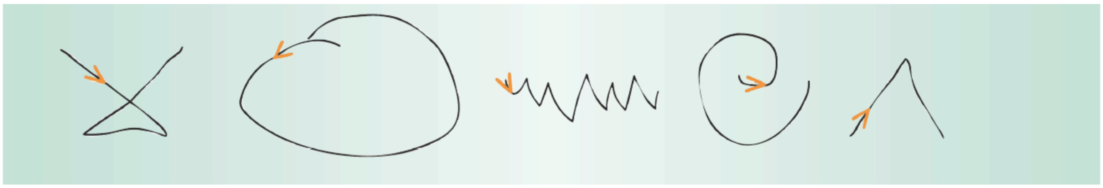


Figure 3. Editing gestures that SILK supports. From left, the first gesture deletes widgets or other objects in the sketch. The next three gestures help inform the recognition process: group objects, ungroup objects, and cycle to the next best inference. The last gesture lets the designer insert typed text or replace a text squiggle with typed text. The arrows show the direction to draw the gestures for best recognition.

Lots of ambiguities

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

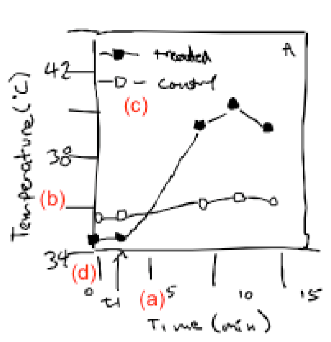


Figure 2. A sketched diagram containing several examples of ambiguity.

Very similar to Galaxy

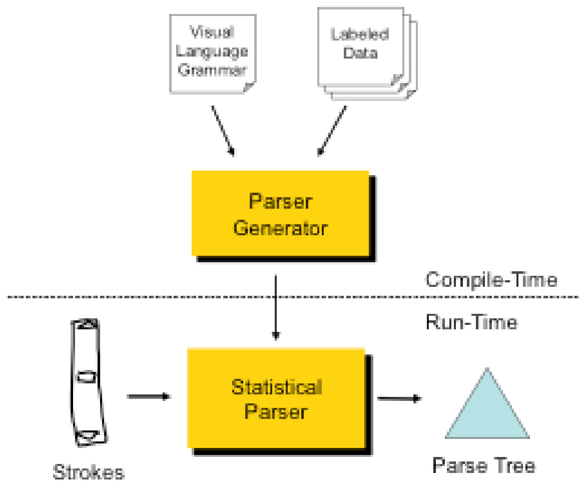
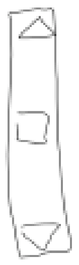


Figure 3. High-level system schematic.

Visual Language Syntax



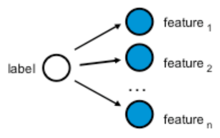
```
vscroll ::= border:vrect handle:square
upArrow:upTri downArrow:downTri {
  // top and equal width
  dist(upArrow.NORTH,border.NORTH).
    range(0,20);
  widthRatio(upArrow,border).range(.4,1.1);

  // bottom side and equal width
  dist(downArrow.SOUTH,border.SOUTH).
    range(0,20);
  widthRatio(downArrow,border).
    range(.4,1.1);

  // center and equal width
 deltaX(handle.CENTER,border.CENTER).
   range(-20,20);
 deltaY(handle.CENTER,border.CENTER).
   range(-30,30);
 widthRatio(handle,border).range(.4,1.1);
}
```

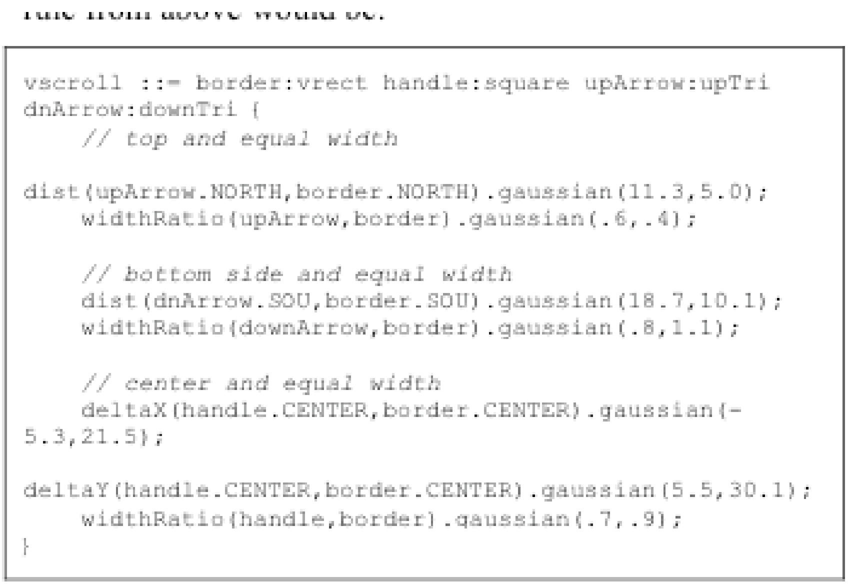
Probability to the rescue

- 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



$$p(l\,|\,\{f_i\}) = \frac{p(l \wedge \{f_i\})}{p(\{f_i\})} = \frac{p(l)\prod_{\{f_i\}} p(f_i\,|\,l)}{\sum_l p(l)\prod_{\{f_i\}} p(f_i\,|\,l)}$$

Fixup the description



rule from above would be:

```
vscroll ::= border:vrect handle:square upArrow:upTri
dnArrow:downTri {
    // top and equal width

dist(upArrow.NORTH,border.NORTH).gaussian(11.3,5.0);
    widthRatio(upArrow,border).gaussian(.6,.4);

    // bottom side and equal width
    dist(dnArrow.SOU,border.SOU).gaussian(18.7,10.1);
    widthRatio(downArrow,border).gaussian(.8,1.1);

    // center and equal width
    deltaX(handle.CENTER,border.CENTER).gaussian(-
5.3,21.5);

deltaY(handle.CENTER,border.CENTER).gaussian(5.5,30.1);
    widthRatio(handle,border).gaussian(.7,.9);
}
```

A parse in action

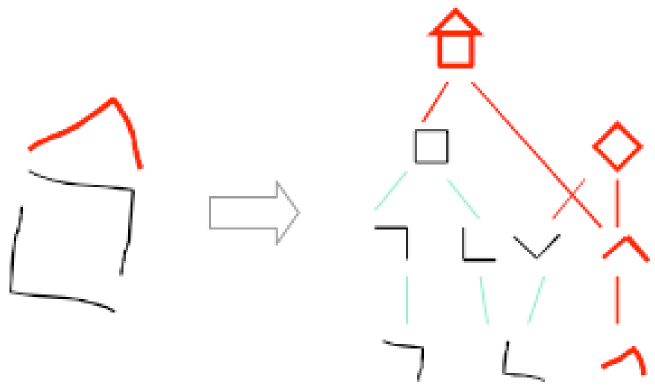


Figure 7. An incremental parse only computes the necessary new nodes in the tree.

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

Massachusetts Institute of Technology

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

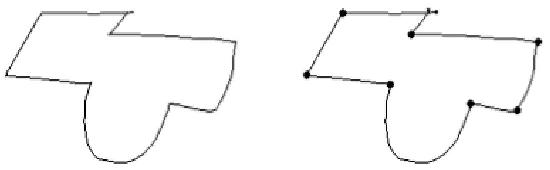


Figure 1: The stroke on the left contains both curves and straight line segments. The points we want to detect in the vertex detection phase are indicated with large dots in the figure on the right. The beginning and the end points of the stroke are indicated with smaller dots.

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

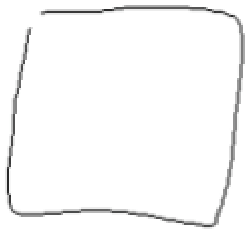


Figure 2: Stroke representing a square.

direction, curvature & speed

- Find places with
  - minimum speed
  - maximal curvature

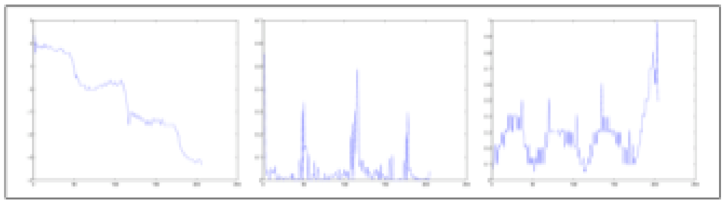


Figure 3: Direction, curvature and speed graphs for the stroke in Fig. 2

CSAIL

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

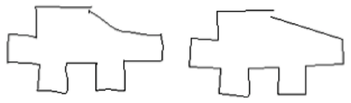


Figure 6: At left the original sketch of a piece of metal; at right the fit generated using only curvature data.

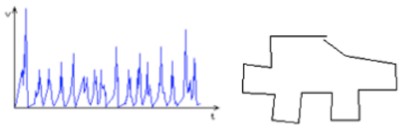


Figure 7: At left the speed graph for the piece; at right the fit based on only speed data.

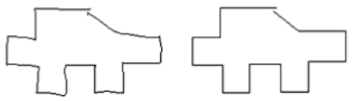
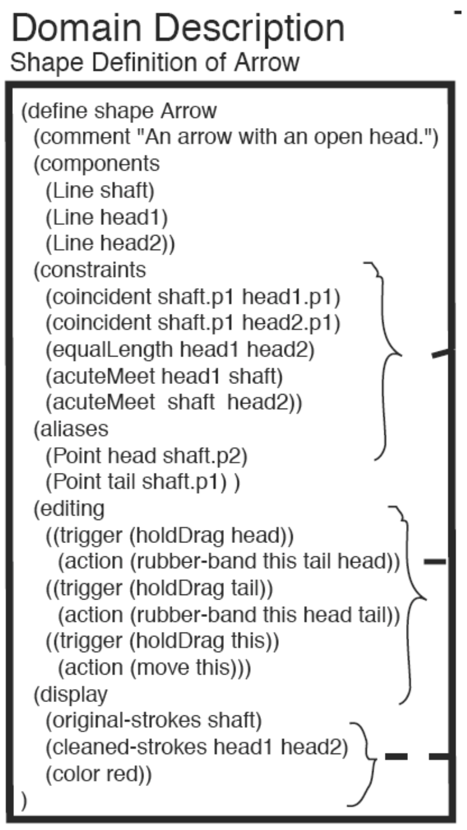


Figure 10: At left the original sketch of a piece of metal revisited, and the final beautified output at right.

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



Domain Description
Shape Definition of Arrow

```
(define shape Arrow
  (comment "An arrow with an open head.")
  (components
    (Line shaft)
    (Line head1)
    (Line head2))
  (constraints
    (coincident shaft.p1 head1.p1)
    (coincident shaft.p1 head2.p1)
    (equalLength head1 head2)
    (acuteMeet head1 shaft)
    (acuteMeet  shaft  head2))
  (aliases
    (Point head shaft.p2)
    (Point tail shaft.p1) )
  (editing
    ((trigger (holdDrag head))
      (action (rubber-band this tail head))
    ((trigger (holdDrag tail))
      (action (rubber-band this head tail))
    ((trigger (holdDrag this))
      (action (move this)))
  (display
    (original-strokes shaft)
    (cleaned-strokes head1 head2)
    (color red))
)
```

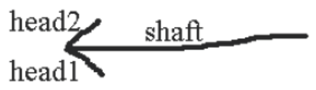
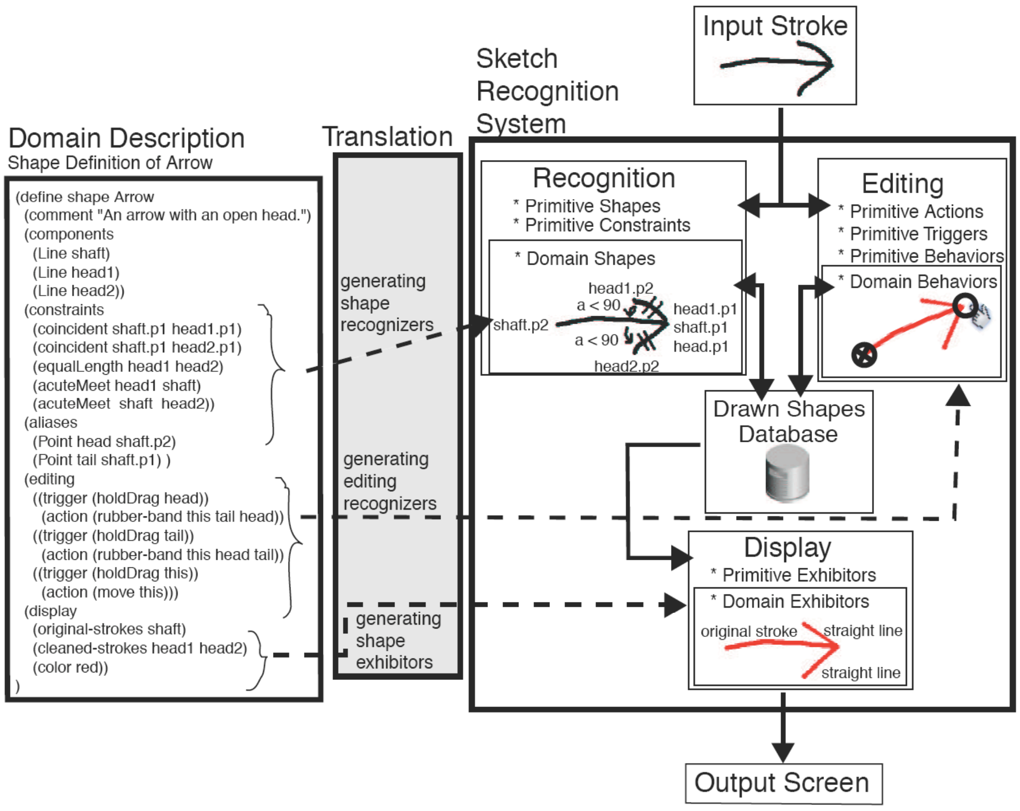


Figure 1: An open arrow.



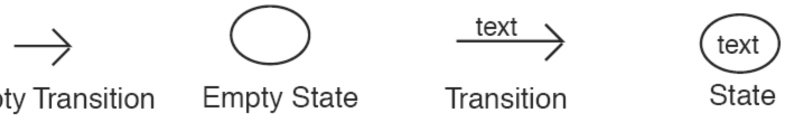
Domain Description

Shape Definition of Arrow

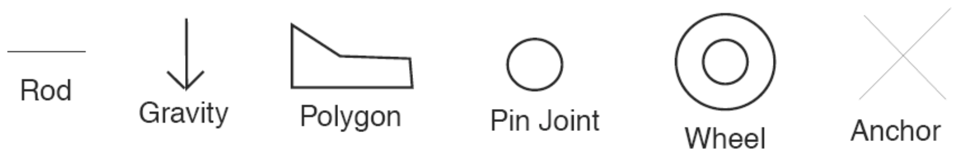
```
(define shape Arrow
  (comment "An arrow with an open head.")
  (components
    (Line shaft)
    (Line head1)
    (Line head2))
  (constraints
    (coincident shaft.p1 head1.p1)
    (coincident shaft.p1 head2.p1)
    (equalLength head1 head2)
    (acuteMeet head1 shaft)
    (acuteMeet  shaft  head2))
  (aliases
    (Point head shaft.p2)
    (Point tail shaft.p1) )
  (editing
    ((trigger (holdDrag head))
      (action (rubber-band this tail head))
    ((trigger (holdDrag tail))
      (action (rubber-band this head tail))
    ((trigger (holdDrag this))
      (action (move this)))
  (display
    (original-strokes shaft)
    (cleaned-strokes head1 head2)
    (color red))
)
```

Some basic shapes that have been defined

Finite State Machines

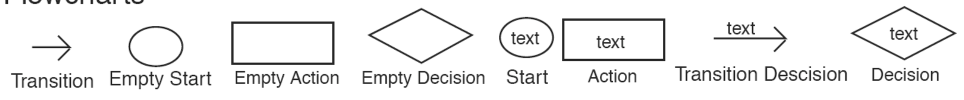


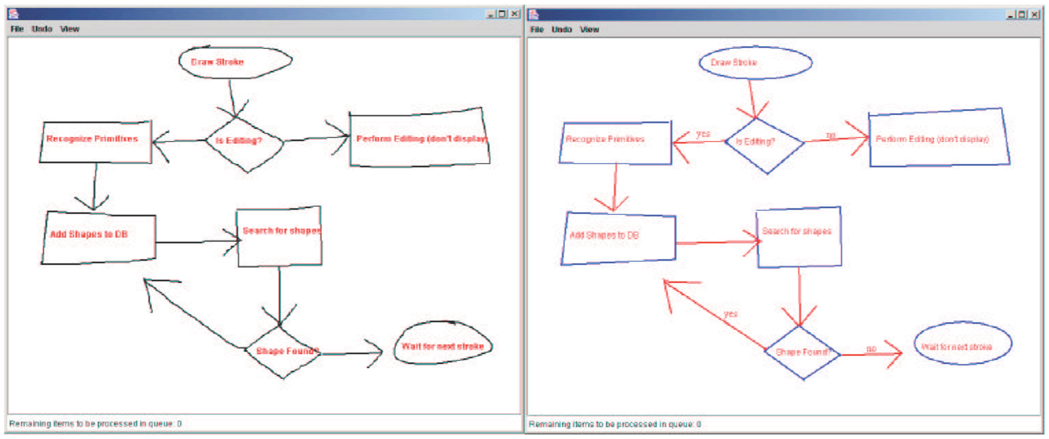
Mechanical Engineering Diagrams



Sketching Flowcharts

Flowcharts

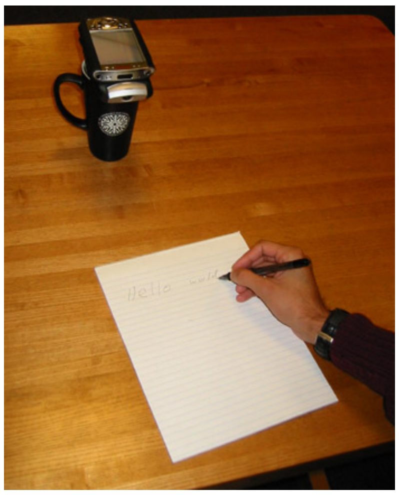




Remaining items to be processed in queue: 0

Figure 4: Flowchart of domain independent system drawn using system. Left side represents the original strokes. Right side represents the cleaned up drawing.

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]l[12][45]+[78]?)?[78]*[4]*(1?[2][369]+l1[25][369]*)([369]+[25]+
8?[147]?[258]*[369]+l[25]*8?[147]+[258]+[369]*)([369]*[58][74]+l[369]+[58][74]*)
```

B