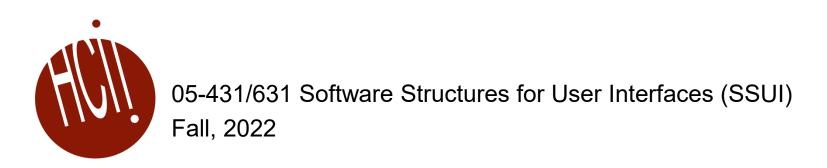
# Lecture 20: Toolkit support for Gestural Input Techniques



© 2022 - Brad Myers

-



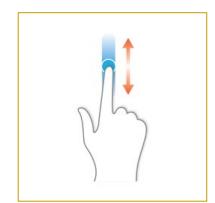
# Logistics

Final Project information is posted



#### What is a "Gesture"

- In HCI, an input to a computer where the path or other properties of the input is important to its recognition, not just the end points
  - Regular drag-and-drop just cares about where starts and finishes, so generally does not count as a "gesture"
- A recognizer is needed to interpret the path so it may be interpreted incorrectly
- Can be done with a mouse, a stylus or finger on touchscreen, or hands in the air in front of a camera
- Can be one or multiple fingers; one or multiple strokes
- On Smartphones, call "tap" a "gesture" to distinguish between tap, long-press, flick, drag, etc.
  - Depends on properties of the action or of the other actions
    - Location, but also speed, timing, etc.







#### **Advantages of Gesture Recognition**

- Very fast to enter
- Single gesture can give both parameters and command
  - E.g., cross out gesture tells both what to do and to what
- Large space of potential gestures
- Can be "natural"
- Can fit in easily with event-based programming
  - Assuming gestures simply invoke a command
- Can be integrated with the toolkit
  - E.g., get events when a gesture starts / finished



# **Disadvantages**

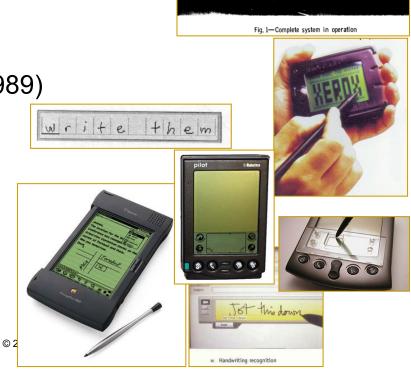
- No affordance user has to know they can be done
- No in-place information on what they look like
- User may find it hard to remember which gesture does what operation (especially if lots)
- System may recognize them incorrectly
- Often cannot be entered correctly by users with disabilities, etc.
- Can be unnatural if designed poorly
- Hard to provide feedback of what is happening, especially if continuous
- Implementation challenge: creating a good recognizer
- Designer must decide if rotation and size invariant

16 25 16 13



# Gestures → Character → Handwriting recognition

- Text entry using hand-printing and hand-writing
  - Rand tablet (1964)
  - PARC Tab QuickWriting (1989)
  - Go PenPoint (1991)
  - Apple Newton (1993)
  - Palm Graffiti (1996)
  - Windows TabletPC (2002)
  - EdgeWrite (2003)



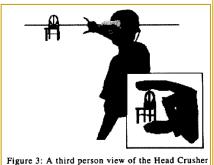


#### **Gestures in 3D**

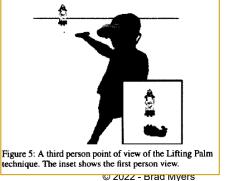
- Gestures for 3D manipulation
  - Mainly pose and path of fingers with datagloves
  - Also elaborate gestures in Teddy
  - May depend on path and timing
  - Wii controller gestures
  - **Kinect Body poses**







technique. The inset shows the first person view.







#### **Gestures for Proofreading**

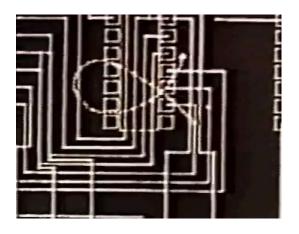
- Well-known proofreading symbols on paper
- Many investigated using these gestures
- COLEMAN, M. L. Text editing on a graphic display device using hand-drawn proofreader's symbols. In Pertinent Concepts in Computer Graphics, Proceedings of the Second University of Illinois Conference on Computer Graphics, M. Faiman and j. Nievergelt, Eds. University of Illinois Press, Urbana, Chicago, London, 1969, pp. 283-290.
- RHYNE, J. R., AND WOLF, C. G. Gestural interfaces for information processing applications. Tech. Rep. RC12179, IBM T.J. Watson Research Center, Sept. 1986.





#### **Trainable Gesture Recognizer**

- Applicon (circa 1970). An interactive trainable computer aided circuit design system using hand-drawn shapes to enter data and commands. Applicon. 16 mm film. <u>Video</u> (2:25 min excerpt)
  - From Bill Buxton <u>Lincoln Labs page</u>. See the <u>Wikipedia entry</u>

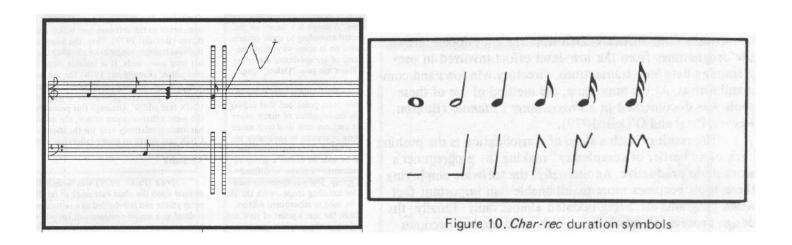






#### **Early Gesture Recognition**

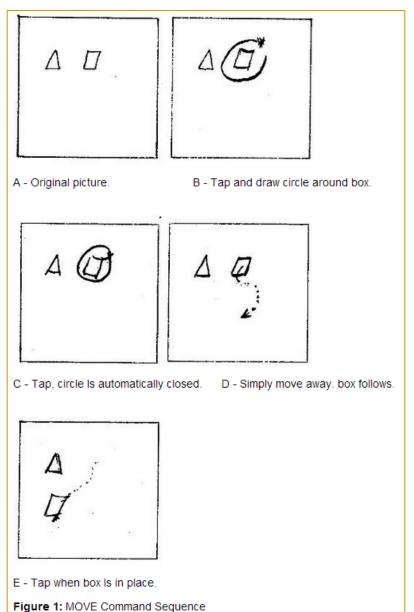
- Buxton, W., Sniderman, R., Reeves, W., Patel, S. & Baecker, R. (1979). The Evolution of the SSSP Score Editing Tools. Computer Music Journal 3(4), 14-25.
   [PDF] [video]
- Can draw gestures for the desired notes to enter music
- Start location determines pitch





# Early Graphical Editing Gestures

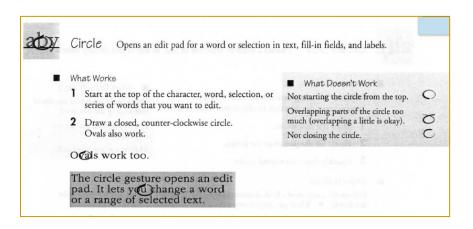
- Buxton, W., Fiume, E., Hill, R., Lee, A. and Woo, C. Continuous Hand-Gesture Driven Input. *Proceedings of Graphics Interface* '83, Edmonton, May 1983, 191-195. <a href="http://billbuxton.com/gesture83.html">http://billbuxton.com/gesture83.html</a> [video]
- Gestures for move and copy
- Copy is same except make a "C" gesture along the path after circling and before moving

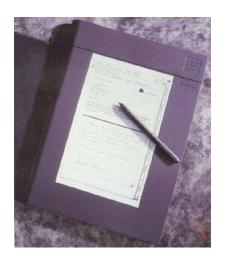




#### Go Corp's "PenPoint" OS

- Founded 1987, released in 1991
- Many gestures for editing, navigation, etc.
  - Flick to scroll and turn pages, circle, insert space, cross-out, insert word, get help, ...
  - Press and hold to start moving or selecting
- Special-purpose recognizer for the built-in gestures
- http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/a/pdf/ Go%20PenPoint%20Getting%20Started.pdf







#### **Dean Rubine's System**

- Dean Rubine at CMU (PhD CSD, 1991) created novel gesture interaction techniques
- Also, a novel "open-source" flexible algorithm, which researchers used for 16 years.
- Paper: Dean Rubine. 1991. Specifying gestures by example. In *Proceedings of the 18th annual* conference on Computer graphics and interactive techniques (SIGGRAPH '91). ACM, 329-337. http://doi.acm.org/10.1145/122718.122753
- Video: Dean Rubine. 1992. Combining gestures and direct manipulation. In *Proceedings of the SIGCHI* Conference on Human Factors in Computing Systems (CHI '92), ACM, actual video (10:20) or (ACM Ref for description)
- Powerful and influential system for single-stroke gesture recognition



1991





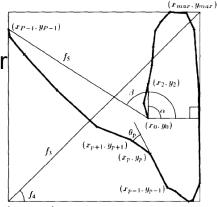
#### **Rubine's Gesture Innovations**

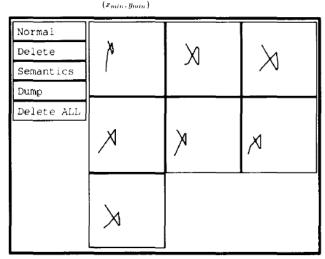
- "Eager recognition" can recognize a gesture while mouse button is still down as soon as it is unambiguous
  - Either wait for mouse pause, or immediately when unambiguous
  - Allows user to continue with direct manipulation
  - E.g., "L" gesture for rectangle, continue to drag for size
  - "C" gesture for copy, "curlicue" for rotate and scale
- Multi-finger gestures also supported
  - Two finger drag and resize
- local video, up through 6:00, 7:00-end



#### Rubine: Gesture recognition algorithm

- Trained with a small number of examples (e.g., 15)
  - Since done by a person, won't be identical
  - Examples should vary in whatever ways they will for the user
    - E.g., different sizes? Different orientations?
- Automatically looks for features of all gestures, that differentiates them
  - Uses a Machine Learning algorithm
    - Statistical Single-Stroke Gesture Recognition
    - Computes matrix inversions, discriminant values, and Mahalanobis distances
  - Experimentally picked a set of 13 features that seemed to work well
    - E.g, "cosine and the sine of the initial angle of the gesture, the length and the angle of the bounding box diagonal, ..."
- Implemented in a system called GRANDMA
- local video, 6:00 through 7:00

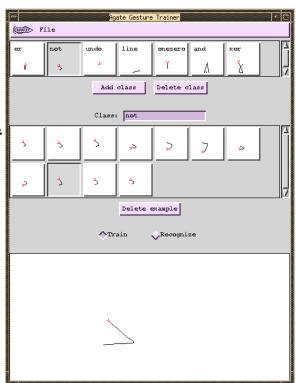




# Human-Computer Interaction Institute

#### **Uses of Rubine's algorithm**

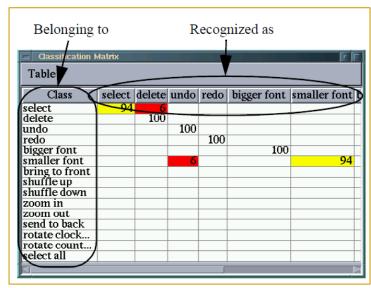
- Many subsequent projects re-implemented and built on his algorithm
  - We implemented it twice, both called "AGATE": A Gesturerecognizer And Trainer by Example
  - Integrated with the standard "interactor" event handling model
  - James A. Landay and Brad A. Myers. "Extending an Existing User Interface Toolkit to Support Gesture Recognition," Adjunct Proceedings of INTERCHI'93. Amsterdam, The Netherlands, April 24-29, 1993. pp. 91-92. (Garnet)
  - Brad A. Myers, Richard G. McDaniel, Robert C. Miller, Alan Ferrency, Ellen Borison, Andrew Faulring, Andy Mickish, Patrick Doane, and Alex Klimovitski, *The Amulet User Interface Development Environment*. 8 minute video. Technical Video Program of the CHI'1997 conference. ACM, <u>YouTube</u> (<u>local copy</u>) 8:27 total, gestures at 6:00-6:30





#### **Improving the Gestures**

- Allan Christian Long Jr., Quill: a gesture design tool for pen-based user interfaces, PhD thesis, UC Berkeley, 2001, (307 pages), pdf
- How to know if the gestures are too similar?
- Chris Long took the Rubine recognizer and analyzes if gestures are too "confusable"
- "Quill" tool
- Similarity in recognition space not necessarily the same as in human perceptual visual space
  - Now would be called "explainable Al"



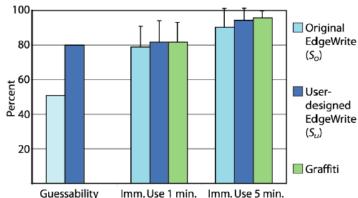


#### **User Designed Gestures**

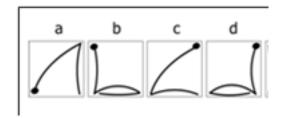
 Jacob O. Wobbrock, Htet Htet Aung, Brandon Rothrock and Brad A. Myers. "Maximizing the Guessability of Symbolic Input" (Short Talk). Extended Abstracts CHI'2005: Human Factors in Computing Systems. Portland, OR, April 2-7, 2005. pp. 1869-1872. pdf. http://doi.acm.org/10.1145/1056808.1057043

 When creating the EdgeWrite gestures, Jake Wobbrock wanted to know what users thought the gestures should be:

- "Guessability of the EdgeWrite unistroke alphabet was improved by users from 51.0% to 80.1%"
- Multiple phases
  - Participants told the constraints
  - Participants propose a set of gestures tricky not to bias answers with prompts
  - Get participants to resolve conflicts
    - since likely to create indistinguishable gestures



**Figure 1.** Guessability and immediate usability results. Error bars represent standard deviations. Graffiti data are from [7].





#### iPhone Gestures

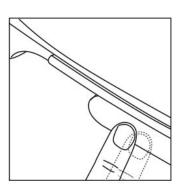
- Quick flick down / up / left / right
  - New behaviors starting in iOS7 in various apps (no affordance)
    - Left and right in Messages, Safari
    - Up and down in home screens
- Swipe down from top
- Swipe up from bottom
- Press and hold (long press)
- Press hard ("3D") now abandoned (same as long-press)
- Two finger zoom
  - Also in photos
- Two finger zoom and rotate
  - Google maps
- Undo shake
- Tilt up turn on
- Double click and hold = zoom
- Three finger tap accessibility zoom

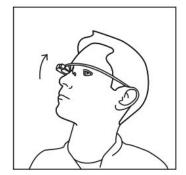
• ...



#### **Google Glass Gestures**

- https://support.google.com/glass/answer/3064184?hl=en
- Small touch pad on right side & Motion sensor
- Activate Glass: Tap the touchpad to turn the display on
- Swipe forward and back: affect content being shown
- Select an item: Tap
- Tilt head up / down: display on / off









#### **Android Gesture Builder**

- All Smartphones have libraries to support programming apps with gestures
  - Often provided to the code by "events" like "mouse-down"
     → "swipe-left"
- Android provides nice tool to define gestures by example
  - http://developer.android.com/training/gestures/index.html
  - http://android-coding.blogspot.com/2011/09/gesturesbuilder-create-your-gestures.html
  - http://android-developers.blogspot.com/2009/10/gestureson-android-16.html



#### **Funny**

 Tyson R. Henry, Scott E. Hudson, Andrey K. Yeatts, Brad A. Myers, and Steven Feiner. "A Nose Gesture Interface Device: Extending Virtual Realities," ACM Symposium on User Interface Software and Technology, Hilton Head, SC, Nov. 11-13, 1991. pp. 65-68. ACM DL or local copy and slides.













Figure 1: Alternative Noses: Bunny, Pig, Elephant, Mouse, Duck, Dog



# New! Michael Liu's "Wiggling" Gesture Grant Computer Interaction Institute

ite HUI

 Michael Xieyang Liu, Andrew Kuznetsov, Yongsung Kim, Joseph Chee Chang, Aniket Kittur, Brad A. Myers. "Wigglite: Low-cost Information Collection and Triage," *UIST '2022: The 35th Annual ACM Symposium on User Interface Software and Technology*, Bend, Oregon, October 29 -November 2, 2022, Article No. 32. Pages 1–16.

https://doi.org/10.1145/3526113.3545661

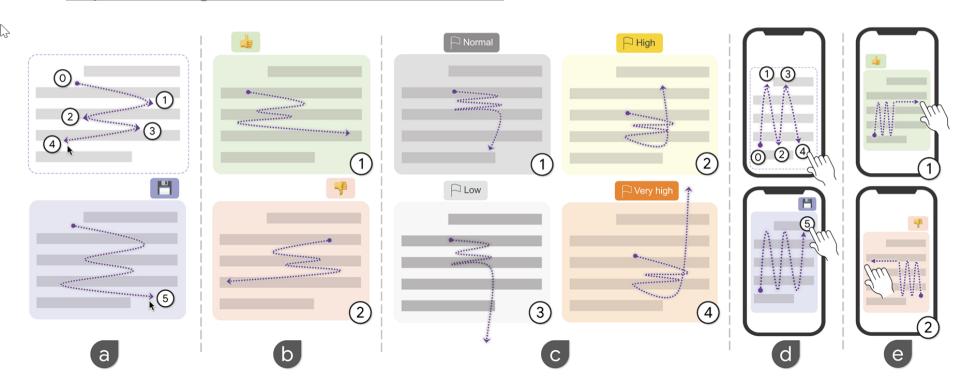


Figure 1: We introduce the "wiggling" technique: rapid back-and-forth movements of a mouse pointer on desktop (a) or a finger on mobile devices (d) that do not require any clicking to perform, yet are sufficiently accurate to select the desired content, while at the same time supporting an optional and natural encoding of valence rating (positive to negative) (on desktop: b1-2, on mobile: e1-2) or classification of priority (to facilitate triage) (c1-4) by ending the wiggle with a swipe in different directions.



#### Wiggling

- Custom detection algorithm
  - Tried many others which didn't work well:
    - \$1 [94] or the Protractor [62] recognizer
    - ML image classification models such as MobileNets [42]
  - Wanted eager recognition after 5 back-and-forth movements
- User evaluation low overhead, high accuracy
- Video of talk (15 min):
   https://youtu.be/ MH81Zuyj64

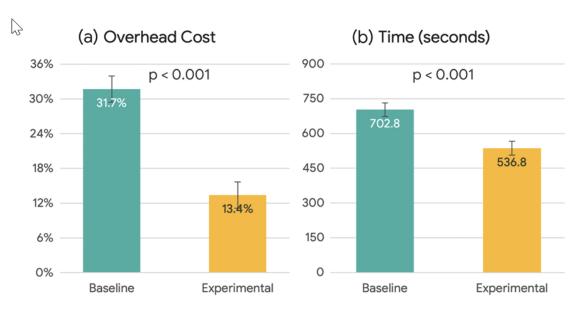
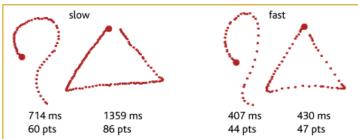


Figure 5: Using Wigglite incurred significantly less overhead cost (a) and helped participants finished the tasks significantly faster (b) when compared to the baseline condition in the user study.



#### Wobbrock's new recognizers

- Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. In Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07). ACM, pp. 159-168. <a href="http://doi.acm.org/10.1145/1294211.1294238">http://doi.acm.org/10.1145/1294211.1294238</a> or <a href="http://faculty.washington.edu/wobbrock/pubs/uist-07.1.pdf">http://faculty.washington.edu/wobbrock/pubs/uist-07.1.pdf</a>
- More efficient and simpler than Rubine's
- Became the new standard that others use for research
- Unistroke and multi-stroke versions
- Match candidate points to remembered templates
- Default: rotation, size and speed invariant



**Figure 3.** A slow and fast question mark and triangle made by subjects using a stylus on a Pocket PC. Note the considerable time differences and resulting numbers of points.



#### **Design criteria**

- Be resilient to variations in sampling due to movement speed or sensing;
- Support optional rotation, scale, and position invariance;
- Require no advanced mathematical techniques (e.g., matrix inversions, derivatives, integrals);
- Be easily written in few lines of code;

Thanks to Jake Wobbrock for these slides!



# Design criteria (cont.)

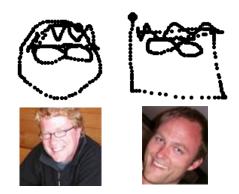
- Be fast enough for interactive purposes;
- Allow developers and application end-users to "teach" it new gestures with only one example;
- Return an N-best list with [0..1] scores that are independent of the number of input points;
- Be conceptually straightforward for easy comprehension, inspection, modification, extension, debugging, etc.

27



#### Live demo

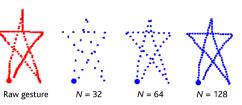
- ~100 lines of JavaScript
  - <a href="http://depts.washington.edu/acelab/proj/dollar/index.html">http://depts.washington.edu/acelab/proj/dollar/index.html</a>



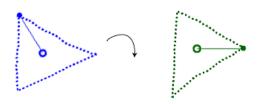


# Four easy steps (on next slides)

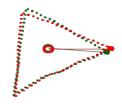
1. Resample



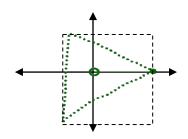
2. Rotate once to 0°



4. Compare to templates



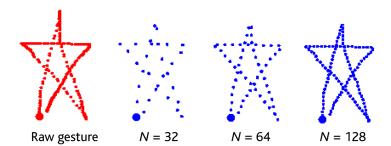
3. Scale, translate



# **Step 1: Resample**

Resample to N equidistant points

(Plamondon & Srihari 2000, Tappert et al. 1990, Kristensson & Zhai 2004)



- Removes clusters and gaps
- Accommodates different sampling rates
- Allows us to compare point C[k] to T<sub>i</sub>[k]

#### Step 1

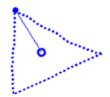
**Step 1.** Resample a *points* path into *n* evenly spaced points.

```
RESAMPLE(points, n)
         I \leftarrow \text{PATH-LENGTH}(points) / (n-1)
         D \leftarrow 0
         newPoints \leftarrow points_0
         foreach point p_i for i \ge 1 in points do
   4
   5
             d \leftarrow \text{DISTANCE}(p_{i-1}, p_i)
            if (D+d) \ge I then
   6
                q_x \leftarrow p_{i-1_x} + ((I-D)/d) \times (p_{i_x} - p_{i-1_x})
                q_y \leftarrow p_{i-1_v} + ((I-D)/d) \times (p_{i_v} - p_{i-1_v})
   8
   9
                APPEND(newPoints, q)
                INSERT(points, i, q) // q will be the next p_i
   10
   11
                D \leftarrow 0
            else D \leftarrow D + d
   12
         return newPoints
PATH-LENGTH(A)
         d \leftarrow 0
         for i from 1 to |A| step 1 do
            d \leftarrow d + \text{DISTANCE}(A_{i-1}, A_i)
   4
         return d
```



# Step 2: Rotate once to 0°

- No closed-form solution for finding best angular alignment (Kara & Stahovich 2004)
- Find angle from (x,y) to first point
  - Call it the "indicative angle"
- Rotate this to 0°
- Approximates rotational alignment between C and T<sub>i</sub>



# Step 2

#### **Step 2.** Rotate *points* so that their indicative angle is at $0^{\circ}$ .

```
ROTATE-TO-ZERO(points)
```

- 1  $c \leftarrow \text{CENTROID}(points)$  // computes  $(\bar{x}, \bar{y})$
- $\theta \leftarrow \text{ATAN}\left(c_y points_{0_y}, c_x points_{0_x}\right) // \text{ for } -\pi \leq \theta \leq \pi$
- 3  $newPoints \leftarrow ROTATE-By(points, -\theta)$
- 4 **return** newPoints

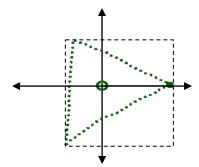
#### ROTATE-BY(points, $\theta$ )

- 1  $c \leftarrow \text{CENTROID}(points)$
- 2 **foreach** point *p* in *points* **do**
- 3  $q_x \leftarrow (p_x c_x) \cos \theta (p_y c_y) \sin \theta + c_x$
- 4  $q_y \leftarrow (p_x c_x) \operatorname{SIN} \theta + (p_y c_y) \operatorname{Cos} \theta + c_y$
- 5 APPEND(newPoints, q)
- 6 **return** newPoints



# **Step 3: Scale, translate**

- Scale to a square
  - Non-uniform
  - Aspect-invariant
- Translate centroid to origin



# Step 3

**Step 3.** Scale *points* so that the resulting bounding box will be of  $size^2$  dimension; then translate *points* to the origin. BOUNDING-BOX returns a rectangle according to  $(min_x, min_y)$ ,  $(max_x, max_y)$ . For gestures serving as templates, Steps 1-3 should be carried out once on the raw input points. For candidates, Steps 1-4 should be used just after the candidate is articulated.

```
1 B \leftarrow \text{BOUNDING-BOX}(points)

2 foreach point p in points do

3 q_x \leftarrow p_x \times (size \mid B_{width})

4 q_y \leftarrow p_y \times (size \mid B_{height})

5 APPEND(newPoints, q)

6 return newPoints

TRANSLATE-TO-ORIGIN(points)

1 c \leftarrow \text{CENTROID}(points)

2 foreach point p in points do

3 q_x \leftarrow p_x - c_x

4 q_y \leftarrow p_y - c_y
```

APPEND(newPoints, q)

return newPoints

SCALE-TO-SQUARE(points, size)



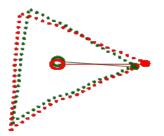
#### At this point...

- ... all templates  $T_i$  and any candidate C have been treated identically.
- Now we must compare C to each  $T_i$ .
  - To which  $T_i$  is C closest?



# **Step 4: Compare to templates**

- Compute score [0..1] for each  $(C, T_i)$ 
  - Score is based on average distance between corresponding points
  - Score should be for <u>best</u> angular alignment for (C, T<sub>i</sub>)
    - Requires search over angles



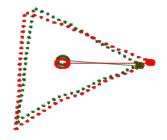


#### "Seed & search"

- Find angle at which avg. point distance is minimized
- Could use brute force
  - +1° for 360°



- ±1° for ±180°
- There can be local minima





#### Step 4

**Step 4.** Match *points* against a set of *templates*. The *size* variable on line 7 of RECOGNIZE refers to the *size* passed to SCALE-TO-SQUARE in Step 3. The symbol  $\varphi$  equals  $\frac{1}{2}(-1 + \sqrt{5})$ . We use  $\theta = \pm 45^{\circ}$  and  $\theta_{\Delta} = 2^{\circ}$  on line 3 of RECOGNIZE. Due to using RESAMPLE, we can assume that A and B in PATH-DISTANCE contain the same number of points, i.e., |A| = |B|.

```
RECOGNIZE(points, templates)

1 b \leftarrow +\infty

2 foreach template T in templates do

3 d \leftarrow DISTANCE-AT-BEST-ANGLE(points, T, -\theta, \theta, \theta_{\Delta})

4 if d < b then

5 b \leftarrow d

6 T' \leftarrow T

7 score \leftarrow 1 - b / 0.5 \sqrt{size^2 + size^2})

8 \mathbf{return} \langle T', score \rangle

PATH-DISTANCE(A, B)

1 d \leftarrow 0

2 \mathbf{for} i \mathbf{from} 0 \mathbf{to} |A| \mathbf{step} 1 \mathbf{do}

3 d \leftarrow d + \mathbf{DISTANCE}(A_i, B_i)

4 \mathbf{return} d / |A|
```

```
DISTANCE-AT-BEST-ANGLE(points, T, \theta_a, \theta_b, \theta_{\Lambda})
           x_1 \leftarrow \varphi \theta_a + (1 - \varphi) \theta_b
        f_1 \leftarrow \text{DISTANCE-AT-ANGLE}(points, T, x_1)
           x_2 \leftarrow (1 - \varphi)\theta_a + \varphi\theta_h
       f_2 \leftarrow \text{DISTANCE-AT-ANGLE}(points, T, x_2)
            while |\theta_b - \theta_a| > \theta_{\Lambda} do
                if f_1 < f_2 then
                   \theta_b \leftarrow x_2
                    x_2 \leftarrow x_1
                   f_2 \leftarrow f_1
                   x_1 \leftarrow \varphi \theta_a + (1 - \varphi) \theta_b
                   f_1 \leftarrow \text{DISTANCE-AT-ANGLE}(points, T, x_1)
    12
                else
            \theta_a \leftarrow x_1
                  x_1 \leftarrow x_2
                  f_1 \leftarrow f_2
    15
                  x_2 \leftarrow (1 - \varphi)\theta_a + \varphi\theta_h
                  f_2 \leftarrow \text{DISTANCE-AT-ANGLE}(points, T, x_2)
    18 return MIN(f_1, f_2)
```

DISTANCE-AT-ANGLE(points, T,  $\theta$ )

- 1  $newPoints \leftarrow Rotate-By(points, \theta)$
- 2  $d \leftarrow \text{PATH-DISTANCE}(newPoints, T_{points})$
- 3 return d

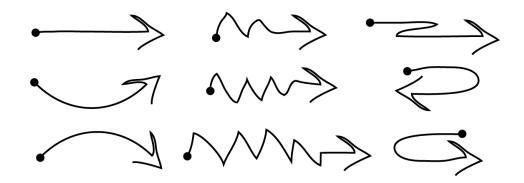


#### **Limitations of \$1**

- Depends on 2-D pairwise point comparisons
  - Resilient to differences in sampling, rotation, scale, position, aspect, and velocity/acceleration
  - But no ovals vs. circles, rectangles vs. squares (This can be added on a per-gesture basis)
  - No differentiation based on speed
- 1-D gestures (lines) should not be scaled in 2-D
- No features are used
  - Gesture "classes" require different templates with the same name



#### arrow "class"



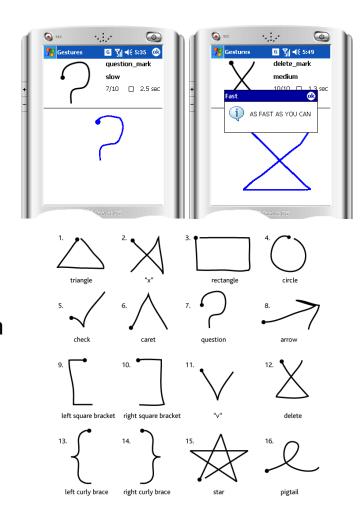
#### **Evaluation**





#### **Method**

- Tested \$1, Rubine, DTW
- 10 subjects
- 16 gestures
  - 10 × (slow, med, fast)
  - 4800 gestures total
  - 1-5 (disliked, liked)
- Train on E=1-9, test on one random from 10-E. Repeat 100× for error rate.
- 1,248,000 total tests
- 3-factor within-subjects design (recognizer, speed, num. train)





#### **Recognition errors**

• \$1 0.98% errors (1.21% without searching over angles)

DTW 0.85%Rubine 7.17%

With 1 template:

\$1: 2.73%DTW: 2.14%

With 9 templates:

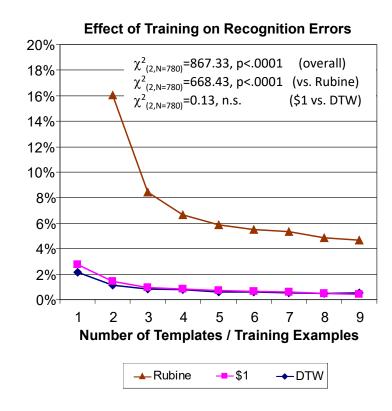
\$1: 0.45%DTW: 0.54%

DTW: 80× slower than \$1

Speedup possible, but complex

Rubine with 9 train: 4.70%

• (Rubine reported ~6.5%.)





# Wobbrock's subsequent gesture work

- AGATe: AGreement Analysis Toolkit for calculating agreement in gesture-elicitation studies (CHI'2015)
- GHoST: Gesture HeatmapS Toolkit for visualizing variation in gesture articulation (ICMI '2014)
- GREAT: Gesture RElative Accuracy Toolkit for measuring variation in gesture articulation (ICMI'2013)
- GECKo: GEsture Clustering toolKit for clustering gestures and calculating agreement (GI '2013)
- \$P: Point-cloud multistroke recognizer for recognizing multistroke gestures as point-clouds (ICMI '2012)
- \$N: Multistroke recognizer for recognizing multistroke gestures as strokes (GI '2012)
- \$1: Unistroke recognizer for recognizing unistroke gestures as strokes (UIST'2007)

46

#### **More References**

- From Bill Buxton, www.billbuxton.com
- The first gesture-related stuff that I did was the single-stroke shorthand that I developed for entering music to the score editor. This was the stepping stone to Unistrokes and that to Grafitti.
  - Buxton, W., Sniderman, R., Reeves, W., Patel, S. & Baecker, R. (1979). The Evolution of the SSSP Score Editing
    Tools. Computer Music Journal 3(4), 14-25. [PDF] [video]
- The paper that you referred to as well as the accompanying video can be found here;
  - Buxton, W., Fiume, E., Hill, R., Lee, A. & Woo, C. (1983). <u>Continuous Hand-Gesture Driven Input. Proceedings of Graphics Interface</u> '83, 9th Conference of the Canadian Man-Computer Communications Society, Edmonton, May 1983, 191-195. [video]
- For a review of Marking and Gesture stuff, see the following two draft chapters of the yet-to-be-finished (!) input book:
  - http://www.billbuxton.com/inputManuscript.html
  - Marking Interfaces
  - Gesture Driven Input
- IMHO, the most useful thing that I have written that guides me, at least, in terms of gestures, is:
  - Buxton, W. (1986). Chunking and Phrasing and the Design of Human-Computer Dialogues, Proceedings of the IFIP World Computer Congress, Dublin, Ireland, 475-480.
- The two things that I always discuss when I speak about gestures are:
  - Kreuger's work & introduction of the pinch gesture, etc.: <a href="http://www.youtube.com/watch?v=d4DUleXSEpk">http://www.youtube.com/watch?v=d4DUleXSEpk</a>
  - Richard Bolt's work combining gesture and speech: http://www.youtube.com/watch?v=RyBEUyEtxQo
- There is also some nice examples from Lincoln Lab:
  - Applicon (circa 1970). An interactive trainable computer aided circuit design system using hand-drawn shapes to
    enter data and commands. Applicon. 16 mm film. 2:25 min excerpt
    - The quick story is this Applicon was a spin-off from Lincoln Labs, and the recognition stuff was born there. Fontaine Richardson, who was behind the work, was a key person at the lab. There was little published on this stuff, but it was the offspring of Sketchpad, and I believe the first commercial system to use these types of gestures or gestures at all.
  - http://www.billbuxton.com/Lincoln.html
  - http://www.billbuxton.com/LincolnLab.pdf
- An old summary which still has some relevance:
  - Buxton, W. (1995). <u>Touch, Gesture & Marking</u>. Chapter 7 in R.M. Baecker, J. Grudin, W. Buxton and S. Greenberg, S. (Eds.)(1995). <u>Readings in Human Computer Interaction: Toward the Year 2000</u> San Francisco: Morgan Kaufmann Publishers.