Stroke processing & gestures

Karan Singh

dgp
Dynamic Graphics Project
University of Toronto
www.dgp.toronto.edu

Issues in Digital Sketching

2D

- Stroke filtering. (clothoids, multi-stroke… what are we filtering?)
- Stroke Processing. (sketch widgets, gestures…)
- Strokes and multi-touch. (gestures, symmetric drawing…)
- Stroke appearance (NPR, neatening…)
- Stroke dynamics (pressure, tilt, direction, temporal order…)
- Seamless UI Control (sketch widgets, crossing menus, gestures…)
- Navigation (paper manip., onion skinning…)
- 2D curve creation: (What are desirable curves, how do we perceive them in relation to our design knowledge?).
- Stroke Perception (what spatio-temporal information do they convey?)

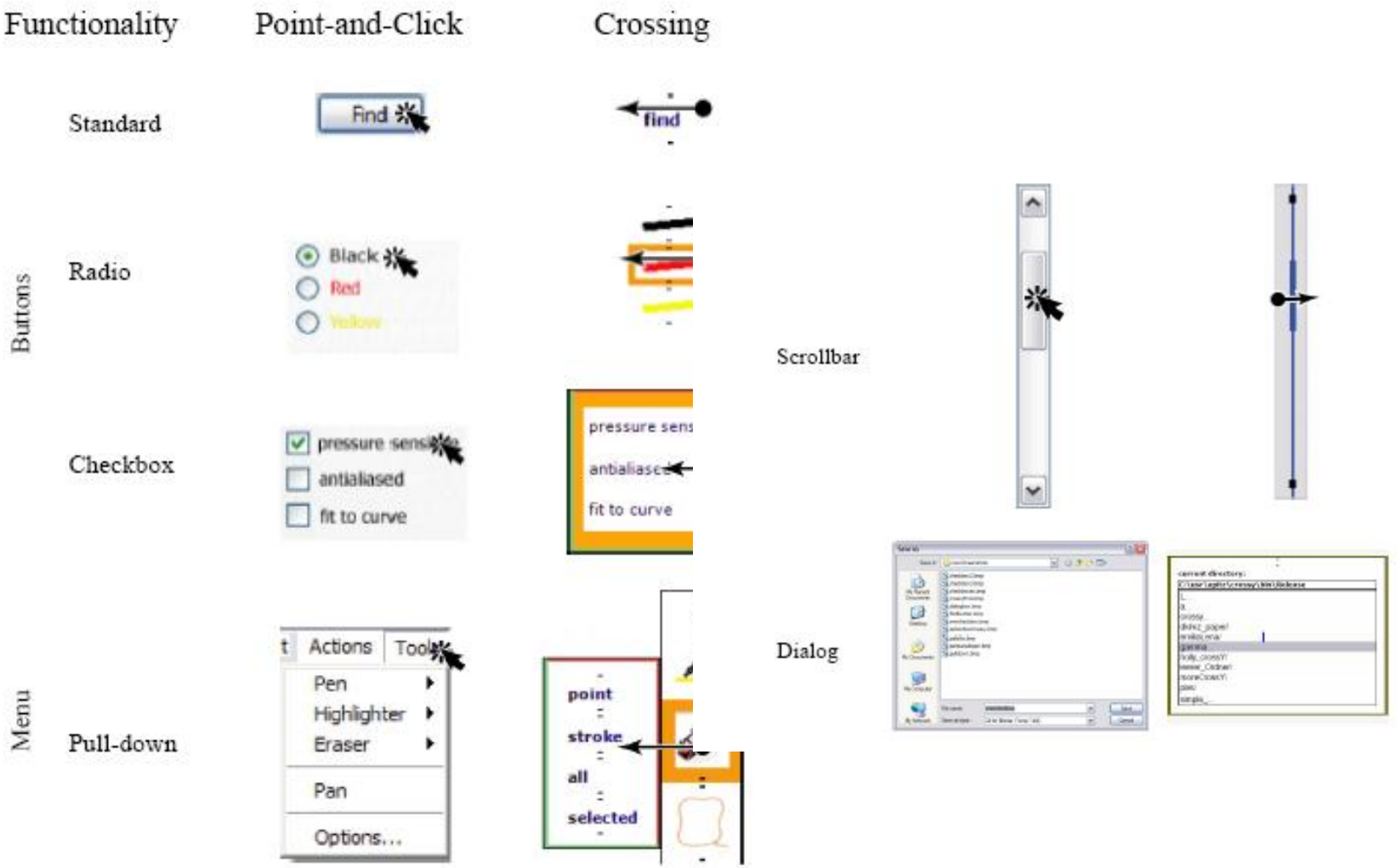
3D (Additional dimension for 3D design, animation or 2D design explorations)

- 3D Navigation. (camera tools, single/multi-view, view bookmarks…).
- 3D curve creation: (2D stroke to 3D curves perception & inference).
- Animation (motion trails, evolving shape fronts…)
- Alternate Designs (co-locating them in space…)

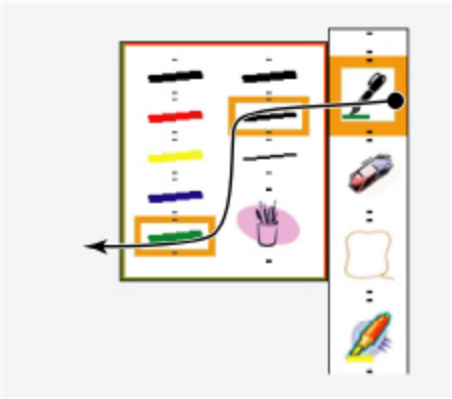
Stroke processing: widgets & UI

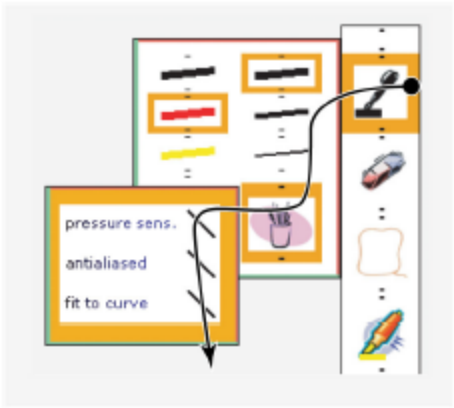
- WIMP GUI not well-suited to sketching.
- Typical 3D manipulation widgets not suited to sketching.

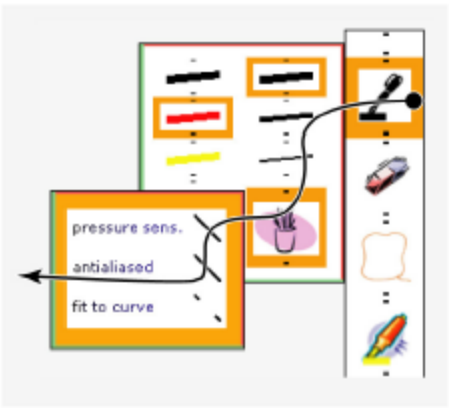
CrossY: Crossing-Based Drawing

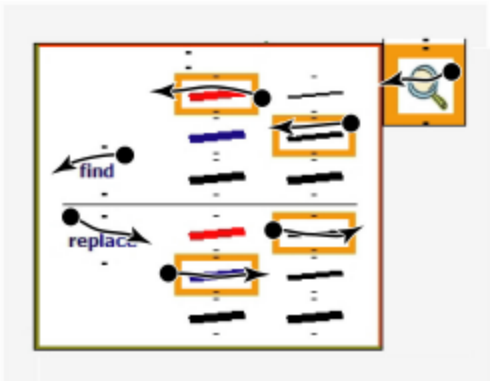


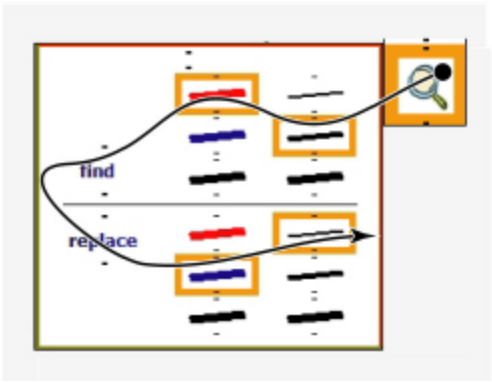
CrossY: Crossing-Based Drawing

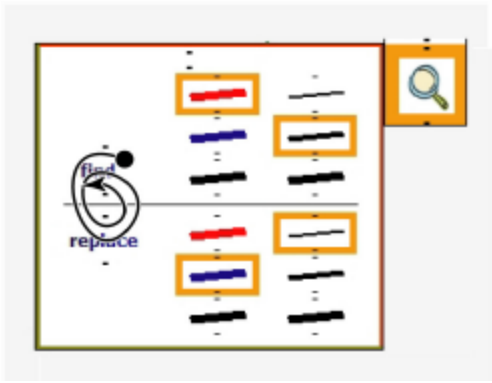






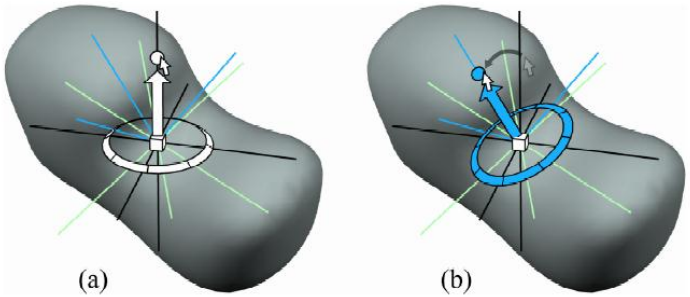




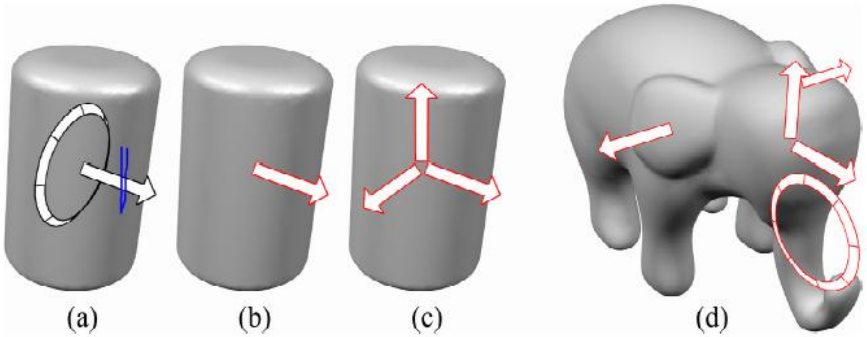


[**Apitz, G. and Guimbretière, F.**
CrossY: A Crossing-Based Drawing Application *ACM UIST, 2004*]

Sketch widgets



suggested axes



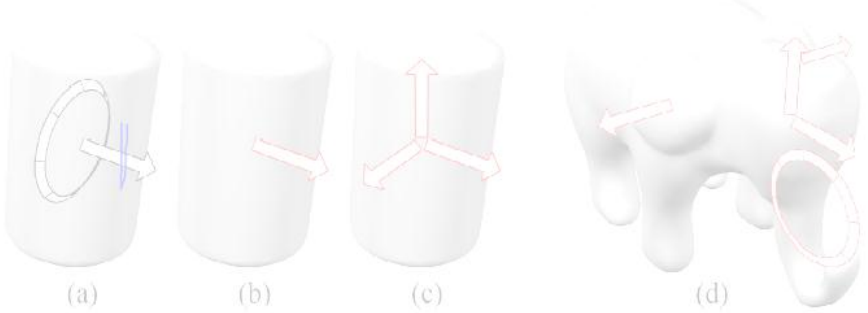
crossing interaction and composition

[**Schmidt, Singh & Balakrishnan** Sketching and Composing Widgets for 3D Manipulation, *Eurographics 2008*]
http://www.dgp.toronto.edu/~rms/pubs/SketchWidgetsEG08.html

Stroke processing: Sketch widgets

suggested axes

Sketching 2D projections of desired 3D strokes is hard!!



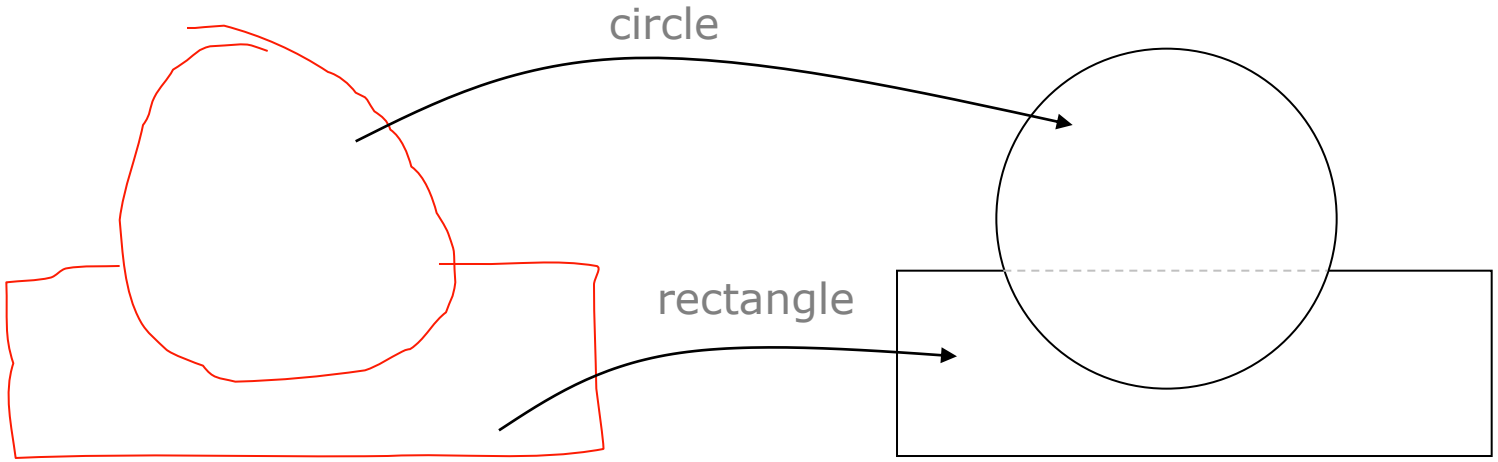
crossing interaction and composition

[**Schmidt, Singh & Balakrishnan** Sketching and Composing Widgets for 3D Manipulation, *Eurographics 2008*]
http://www.dgp.toronto.edu/~rms/pubs/SketchWidgetsEG08.html

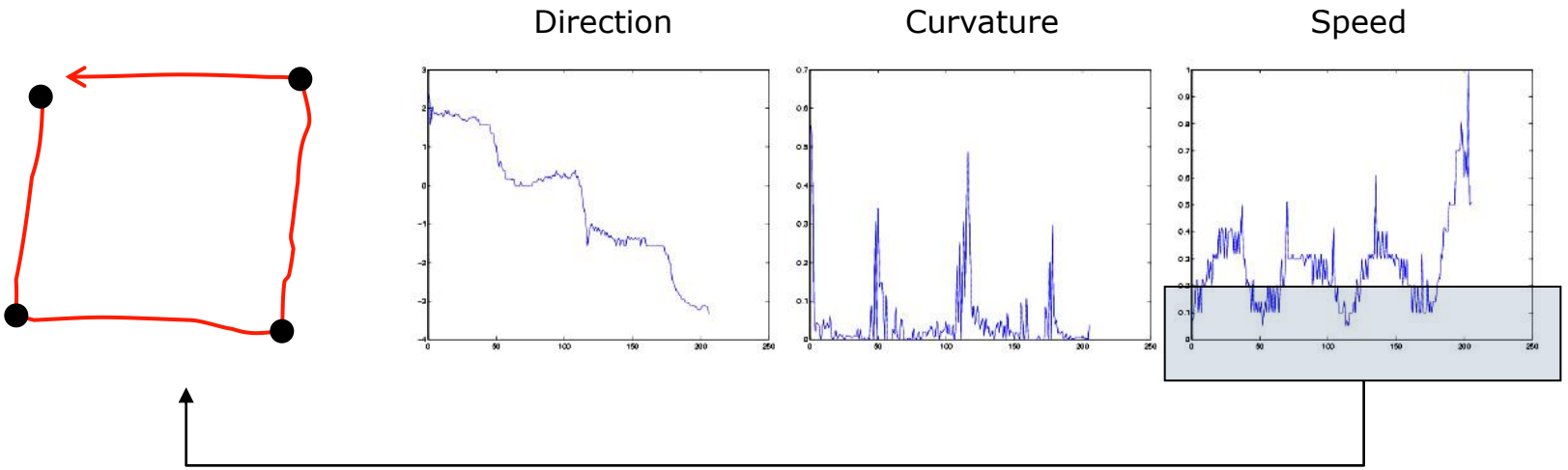
Further stroke processing

- Abstraction.
- Segmentation, classification, recognition.
- Beautification.
- Oversketching.

Stroke recognition



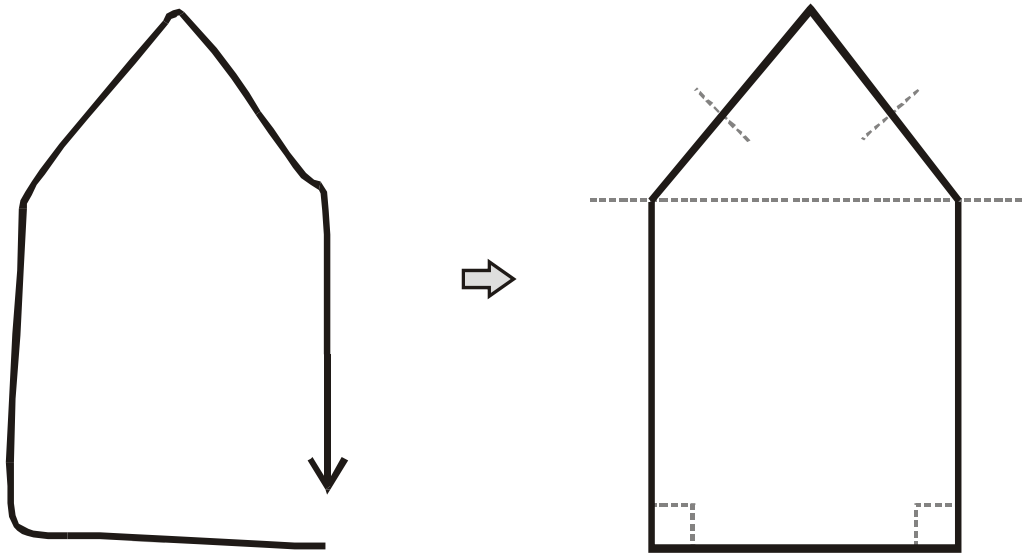
Stroke abstraction: finding corners



[**T. Sezgin et al.**, *Sketch Based Interfaces: Early Processing for Sketch Understanding*, Workshop on Perceptive User Interfaces, 2001.]

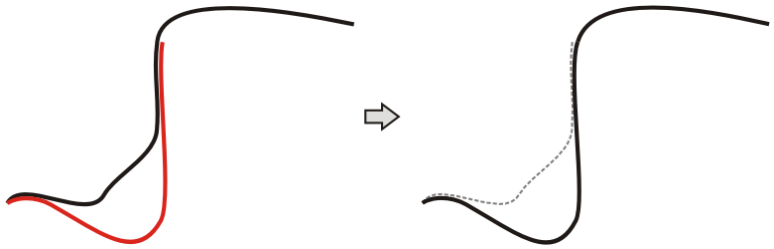
Beautification

- Fit primitive shapes.
- Infer geometric relations and constraints.
  - Parallelism
  - Perpendicularity
  - Symmetry
  - Isometry
  - Snapping



Oversketching

- Interactive sketch correction
  1. Find affected region
  2. Splice in new stroke
  3. Smooth connection



Gestural input

- “Gesture-based interfaces offer an alternative to traditional keyboard, menu, and direct manipulation interfaces.” [Rubine]
- “Pen, finger, and wand gestures are increasingly relevant to many new user interfaces.” [Wobbrock]

Recognizable spatio-temporal strokes.

What can gesture?

mouse, pen, wands, fingers, hands, face, body...

Why support gestures?

- Natural form of communication
- Efficient
  - A single stroke can indicate:
    - The operation
    - The operand
    - Additional parameters
  - A proofreader’s mark indicates [Rubine] :
    - that a move should occur (operation)
    - the text that should be moved (the operand)
    - and the new location of the text (an additional param)

Gesture support

- Ad-hoc or pre-defined: "Recognizers that use heuristics specifically tuned to a pre-defined set of gestures." [Wobbrock 2007]
  - Application specific: shorthand, chinese Brush Painting, musical scores, chemical formulas.
  - Platform specific: gesture libraries.

- Template-based or systematic.
  - Toolkit or framework
  - Simple algorithm

Ad-hoc vs. template-based

- Ad-hoc can recognize more complex gestures.
- Harder to train template-based gestures.
- Better consistency of gestural use in ad-hoc systems.
- Better gesture collision handling in ad-hoc systems.

- Ad-hoc doesn’t allow new gestures and limited customization.

GRANDMA approach

1. Create a new gesture handler and associate it with a class.
2. Draw gesture ~15 times.
3. Define semantics (in Objective C)

Problem Statement

- Gesture is represented as an array g of P sample points:

  $G_p = (x_p, y_p, t_p)$

  $0 \leq p \leq P$

- Problem: Given an input gesture g and set $\{C_1, C_2, \ldots\}$ of gesture classes determine which class g belongs to.

GRANDMA Algorithm

- 13 Features
- A gesture class is a set of weights assigned to each feature
  - Gestures are given a grade by the linear evaluation function resulting from the weights
  - A gesture is assigned to the class with the maximum grade.
- Training assigns weights to the 13 features
- Gestures are rejected if the grade assigned to two classes is similar

$1 recognizer

- Most recognizers are hard to write and involve a certain amount of machine learning.
- Toolkits are not available in every setting.
- i.e. easily implement your own.

$1 goals

- Resilience to sampling.
- Require no advance math.
- Small code.
- Fast.
- 1-gesture training.
- Return an N-best list with scores.

$1 algorithm

- Resample the input
  - N evenly spaced points
- Rotate
  - “Indicative” angle between centroid and start point
- Scale
  - Reference square
- Re-rotate and Score
  - Score built from average distance between candidate and template points

Limitations

- Cannot distinguish between gestures whose identities depend on aspect ratios, orientations.
  - Square from rectangle
  - Up arrow from down arrow
- Cannot be distinguished based on speed.
- Only single strokes.
- Stroke order is important.
- Closed strokes?
- Gestalt gestures!