

# Programming and using stroke shortcuts

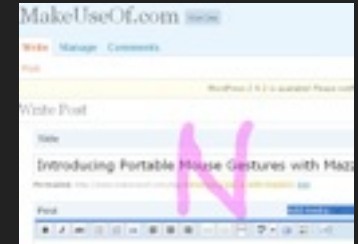
Caroline Appert  
CNRS and Université Paris-Sud XI

# Gestural interaction in this talk

- **Stroke shortcut:** A continuous series of points to trigger a command of the application



*New Tab*



*New File*

- It is an alternative to the conventional “point target then click”. Stroke shortcuts are especially suitable to configurations where targets may be too small (small display) or occluded (direct input).



# Programming stroke shortcuts

# Programming stroke shortcuts is costly

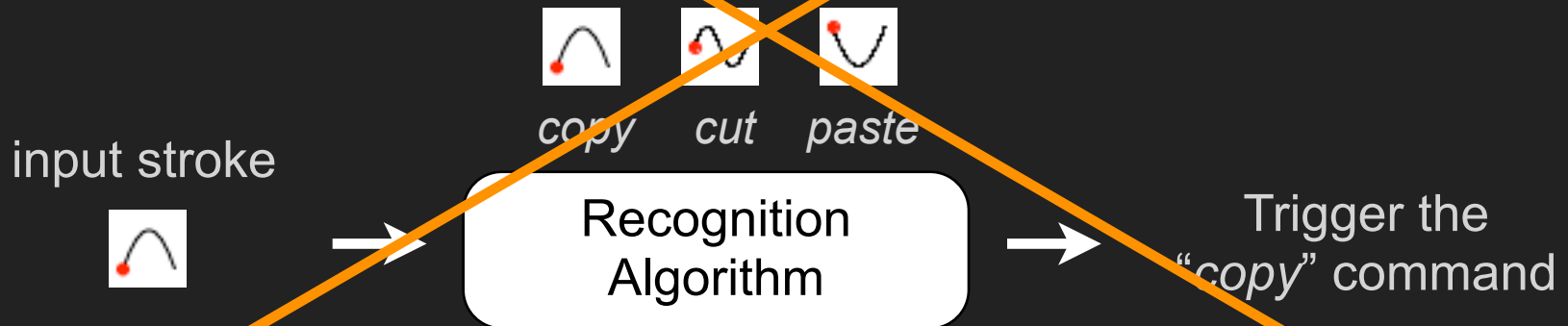
- Programming a recognition algorithm
- Designing a set of *stroke-command* mappings
- Linking recognition to the actual command in the application



- Programming help to allow end users discover and learn shortcuts (2nd part of this talk)

# Usual graphical toolkits (Swing, etc.)

- Programming a recognition algorithm
- Designing a set of *stroke-command* mappings
- Linking recognition to the actual command in the application



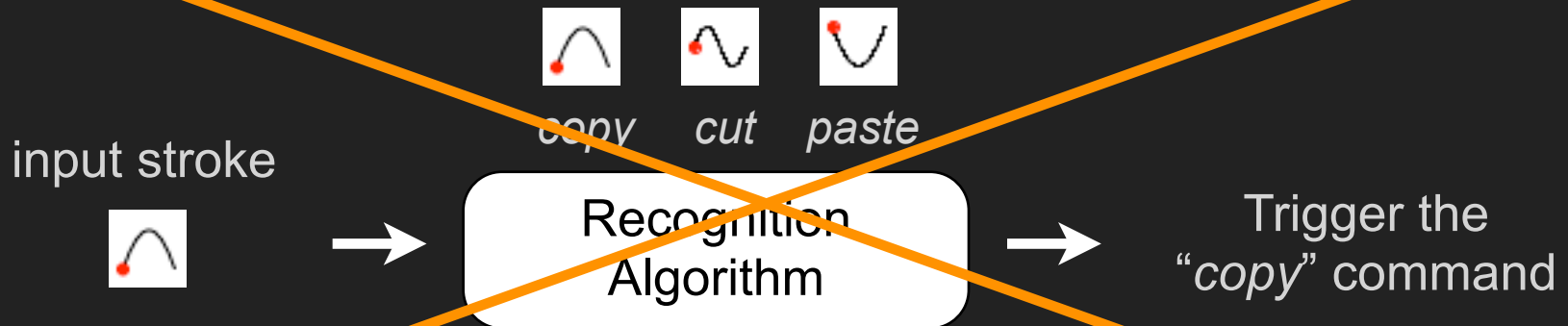
- Programming help to allow end users discover and learn shortcuts (2nd part of this talk)

- Programming a recognition algorithm

## Defining

- ~~Designing~~ a set of *stroke-command* mappings

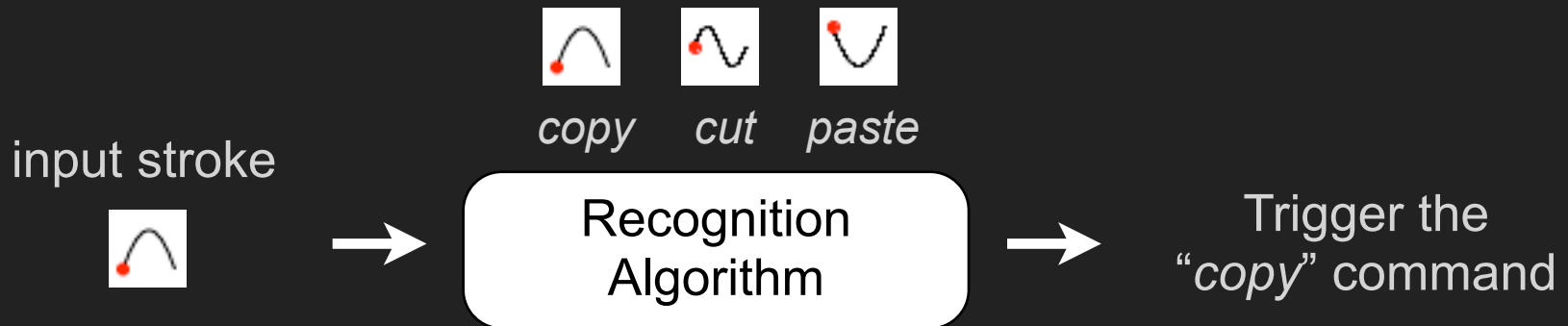
- ~~Linking recognition to the actual command in the application~~



- ~~Programming help to allow end users discover and learn shortcuts (2nd part of this talk)~~

# Stroke Shortcuts Toolkit (SST) [Appert and Zhai, 09]

- Programming a recognition algorithm
- Designing a set of *stroke-command* mappings
- Linking recognition to the actual command in the application



- Programming help to allow end users discover and learn shortcuts (2nd part of this talk)

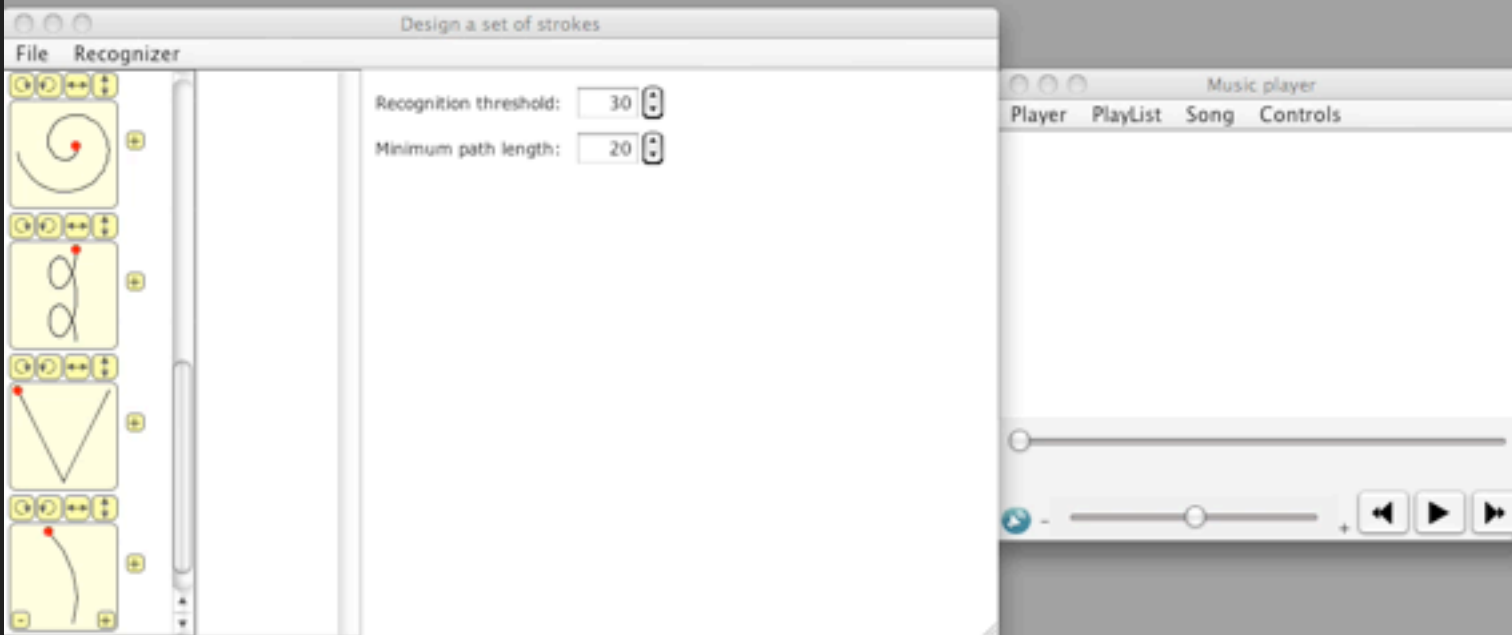
# Stroke Shortcuts Toolkit (SST)

```
StrokeShortcuts manager = new StrokeShortcuts(  
                                playerFrame, playerFrame.dialogAbout);  
manager.launchDesignEnvironment();
```

Programming and using gestural interaction



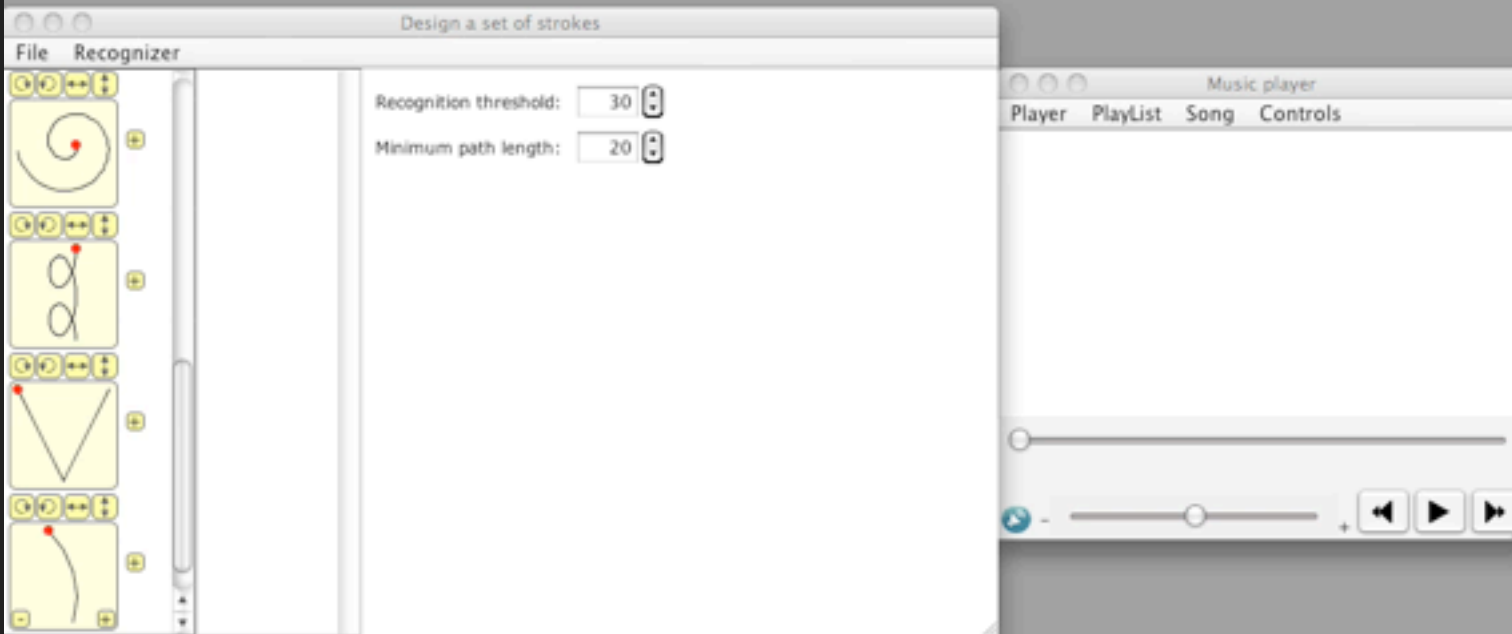
# Stroke Shortcuts Toolkit (SST)



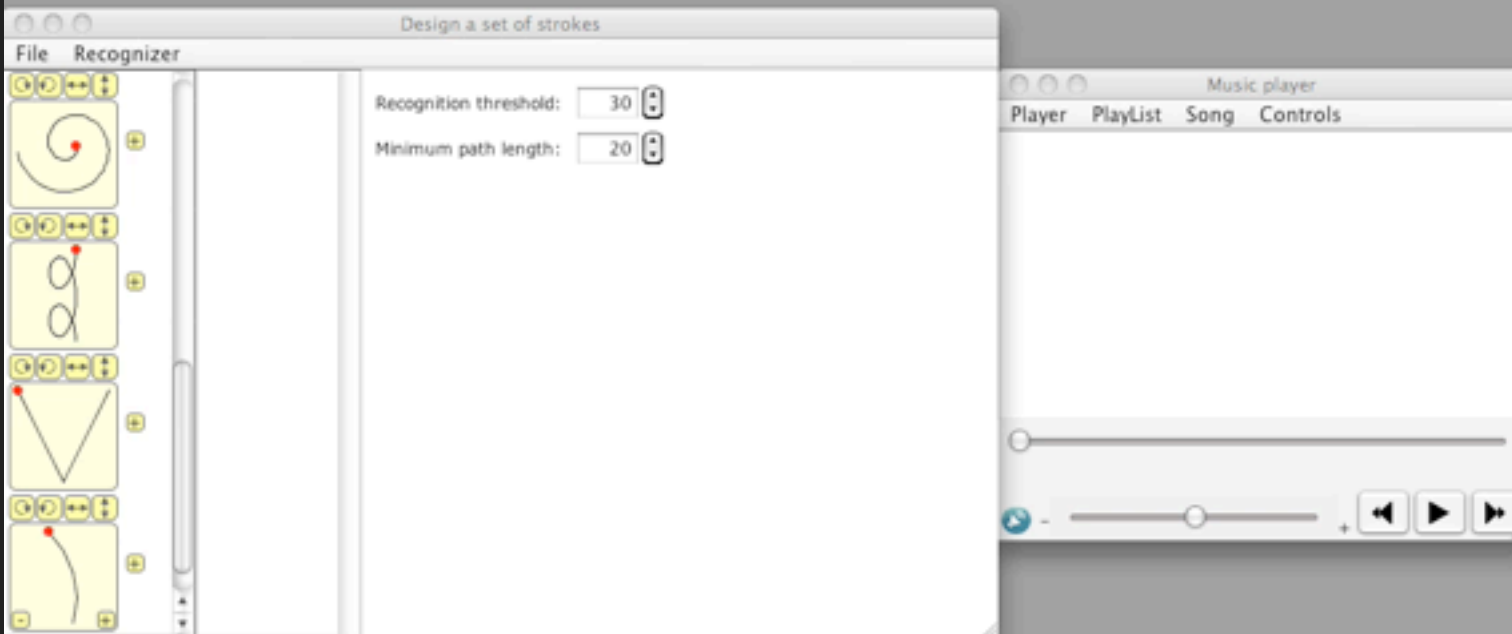
```
StrokeShortcuts manager = new StrokeShortcuts(  
                                playerFrame, playerFrame.dialogAbout);  
manager.launchDesignEnvironment();
```

Programming and using gestural interaction

# Stroke Shortcuts Toolkit (SST)



# Stroke Shortcuts Toolkit (SST)

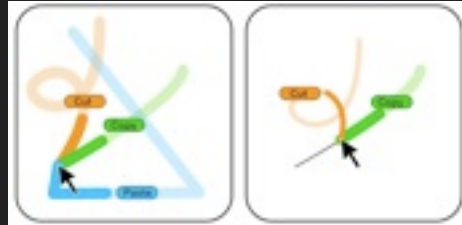


```
manager.addShortcuts("player.strokes", MENU_PREVIEW, TOOLTIP_PREVIEW);
```

# Using stroke shortcuts

# Help for end users

- Available in SST :
  - menu items, tooltips (time costly)
  - cheat sheet (space costly)
- A nicer help... :

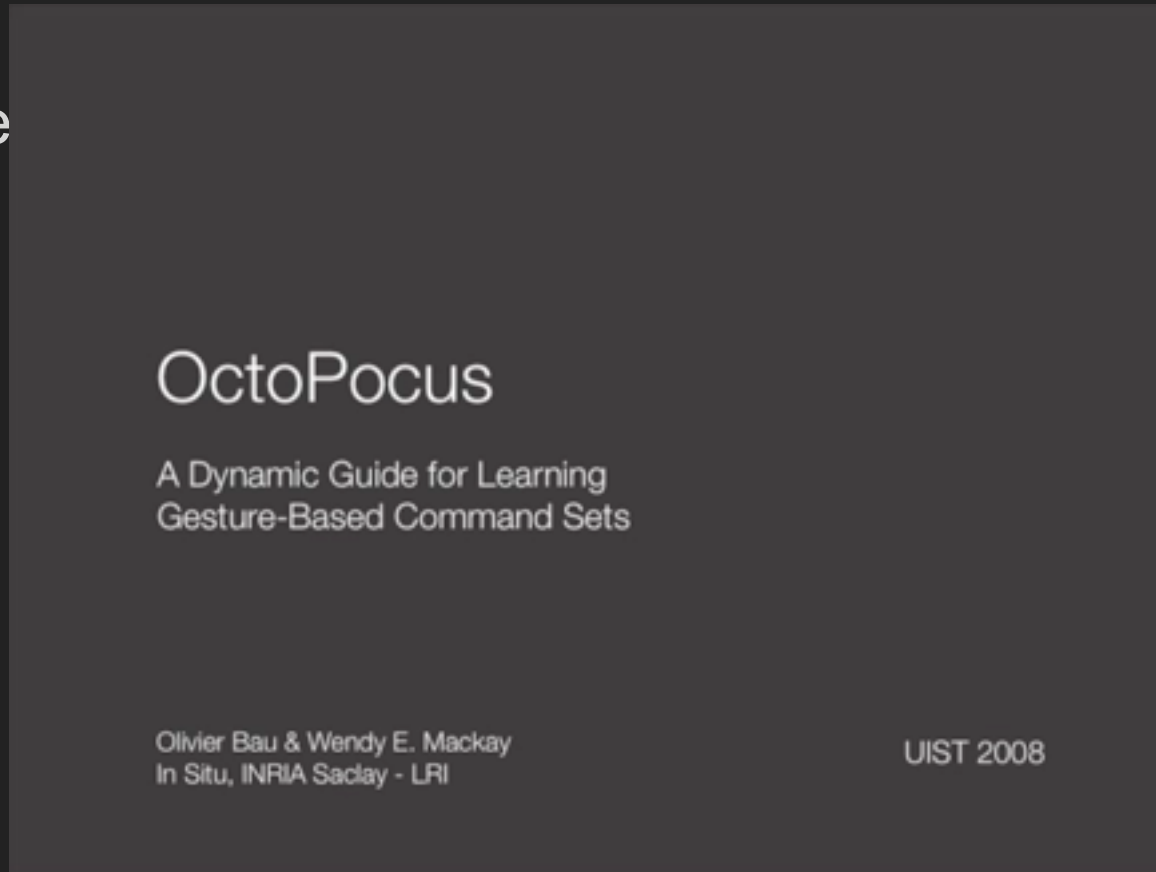


OctoPocus

[Bau & Mackay, 08]

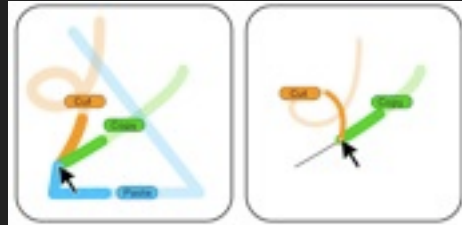
# Help for end users

- Available in SST :
  - menu items, tooltips (time costly)
  - cheat sheet (space costly)
- A nicer he



# Help for end users

- Available in SST :
  - menu items, tooltips (time costly)
  - cheat sheet (space costly)
- A nicer help... :

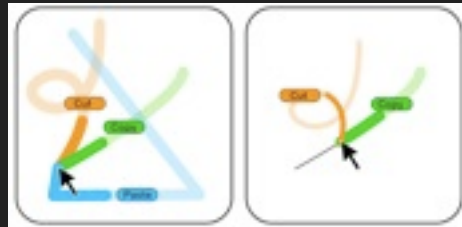


OctoPocus

[Bau & Mackay, 08]

# Help for end users

- Available in SST :
  - menu items, tooltips (time costly)
  - cheat sheet (space costly)
- A nicer help... :



OctoPocus

[Bau & Mackay, 08]

⇒ requires recognizing incomplete stroke input

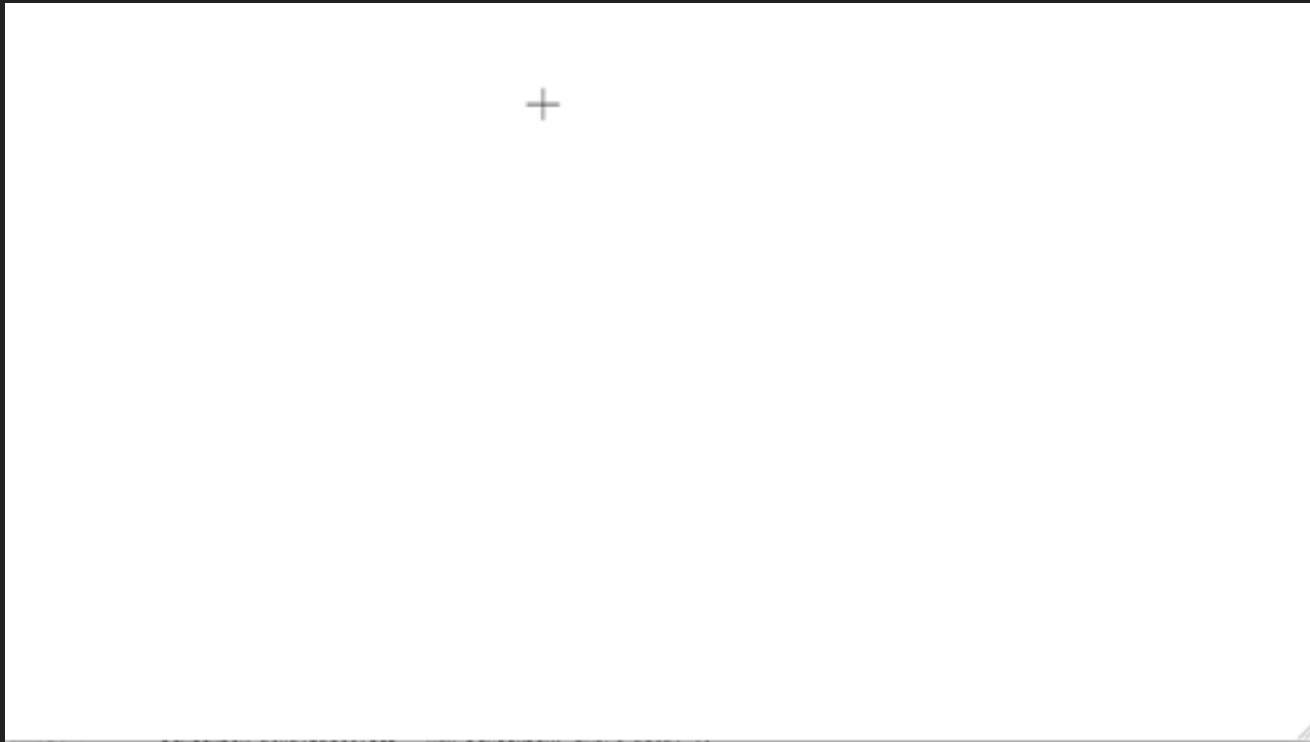


# Stroke shortcut and scale independence

- The same stroke shape at different scales actually invokes the same command

# Stroke shortcut and scale independence

- The same stroke shape at different scales actually invokes the same command

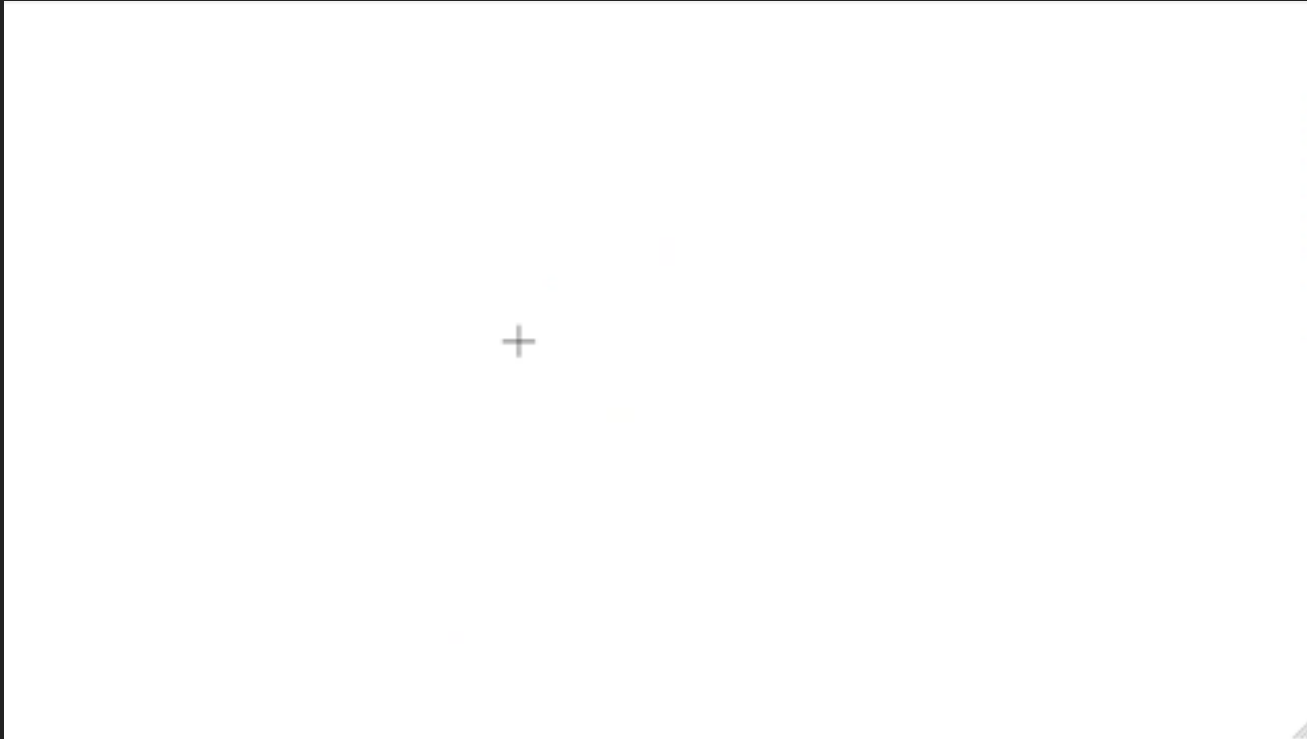


# Scale independence and contextual help

- The first version of OctoPocus was not robust to scale variations...

# Scale independence and contextual help

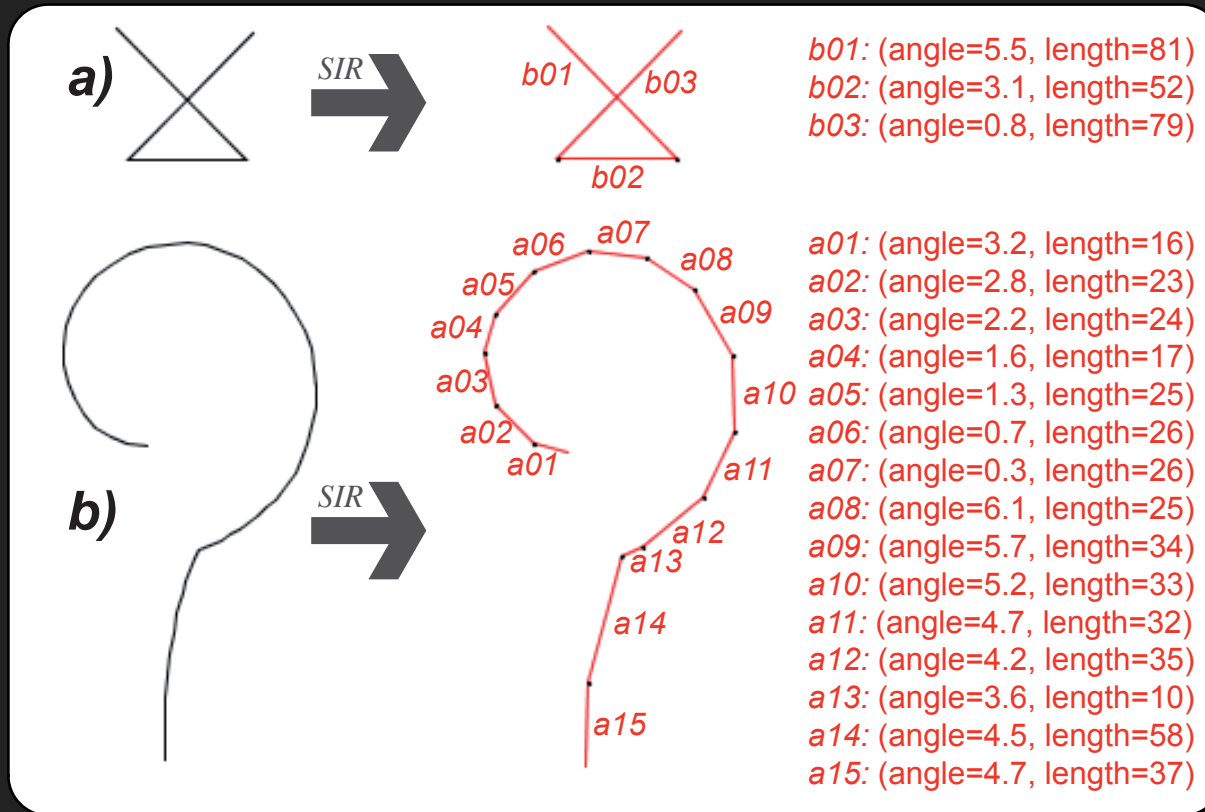
- The first version of OctoPocus was not robust to scale variations...



# Recognition and scale detection for incomplete stroke

[Appert and Bau, 10]

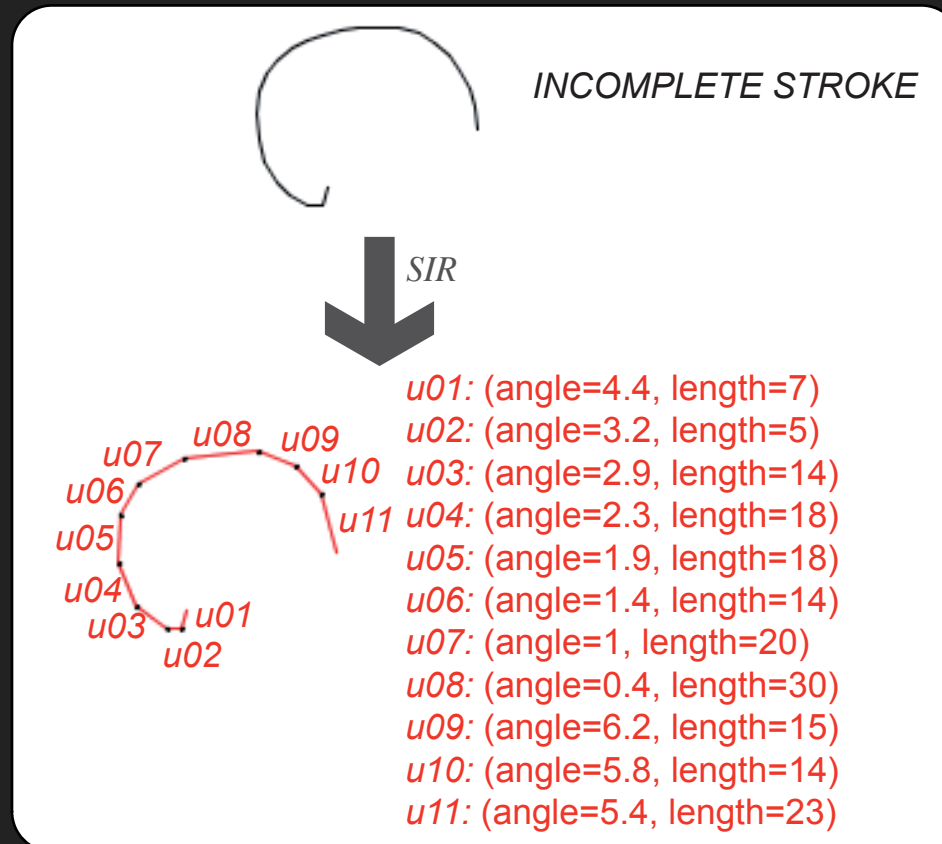
1. For each template, compute a Scale Independent Representation (SIR), i.e. a series of segments (angle, length)



# Recognition and scale detection for incomplete stroke

[Appert and Bau, 10]

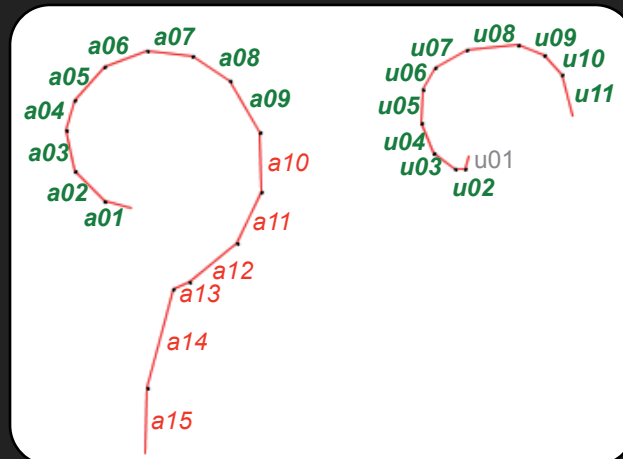
1. For each template, compute a Scale Independent Representation (SIR), i.e. a series of segments (angle, length)
2. Compute the SIR of the incomplete input, SIRinput



# Recognition and scale detection for incomplete stroke

[Appert and Bau, 10]

1. For each template, compute a Scale Independent Representation (SIR), i.e. a series of segments (angle, length)
2. Compute the SIR of the incomplete input, SIR<sub>input</sub>
3. Compare SIR<sub>input</sub> with each SIR template to find candidates:
  - a candidate begins with a similar SIR in *terms of angles* using a tolerance threshold
  - if more than 10% of the length of SIR<sub>input</sub> is not similar to a template, the template is discarded (smaller non matching parts are ignored)

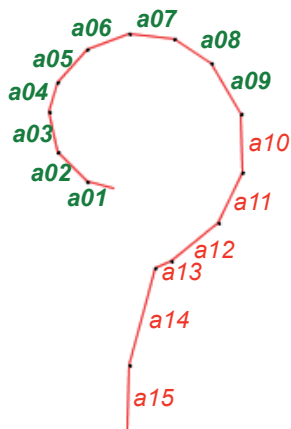


# Recognition and scale detection for incomplete stroke

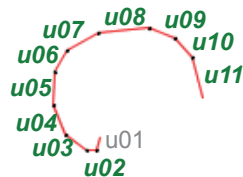
[Appert and Bau, 10]

1. For each template, compute a Scale Independent Representation (SIR), i.e. a series of segments (angle, length)
2. Compute the SIR of the incomplete input, SIRinput
3. Compare SIRinput with each SIR template to find candidates:
4. Compute scales: the mean of the ratios in *terms of lengths*

TEMPLATE



INCOMPLETE STROKE



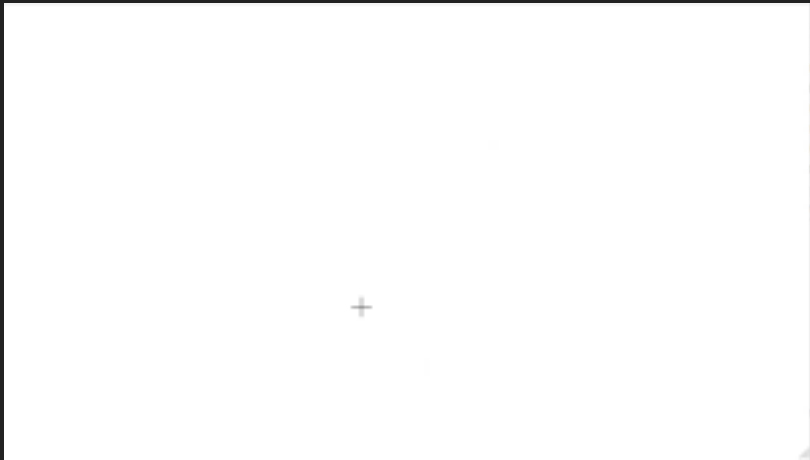
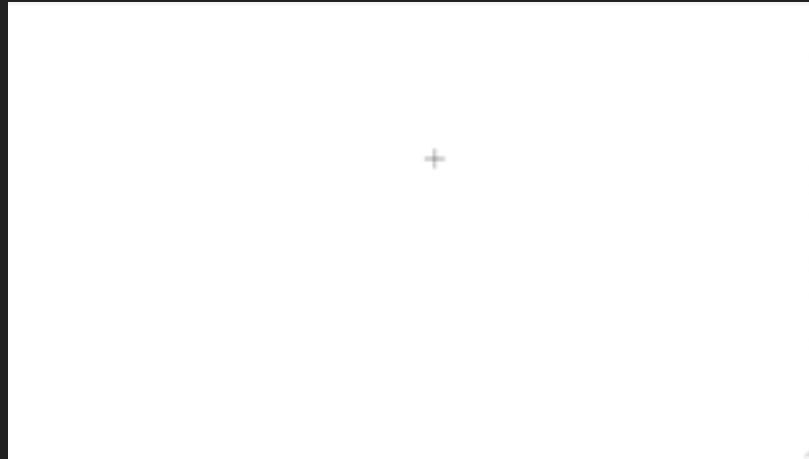
$\text{length}(u01+u02)$	$= 13 = 0.81 * \text{length}(a01)$
$\text{length}(u03)$	$= 14 = 0.62 * \text{length}(a02)$
$\text{length}(u04)$	$= 18 = 0.76 * \text{length}(a03)$
$\text{length}(u05)$	$= 18 = 1.07 * \text{length}(a04)$
$\text{length}(u06)$	$= 14 = 0.56 * \text{length}(a05)$
$\text{length}(u07)$	$= 20 = 0.78 * \text{length}(a06)$
$\text{length}(u08)$	$= 30 = 1.15 * \text{length}(a07)$
$\text{length}(u09)$	$= 15 = 0.60 * \text{length}(a08)$
$\text{length}(u10)$	$= 14 = 0.42 * \text{length}(a09)$

$$\begin{aligned} \text{estimated scale} &= (0.81 + 0.62 + 0.76 + 1.07 + 0.56 + 0.78 + 1.15 + 0.60 + 0.42) / 9 \\ &= 0.75 \text{ of template} \end{aligned}$$

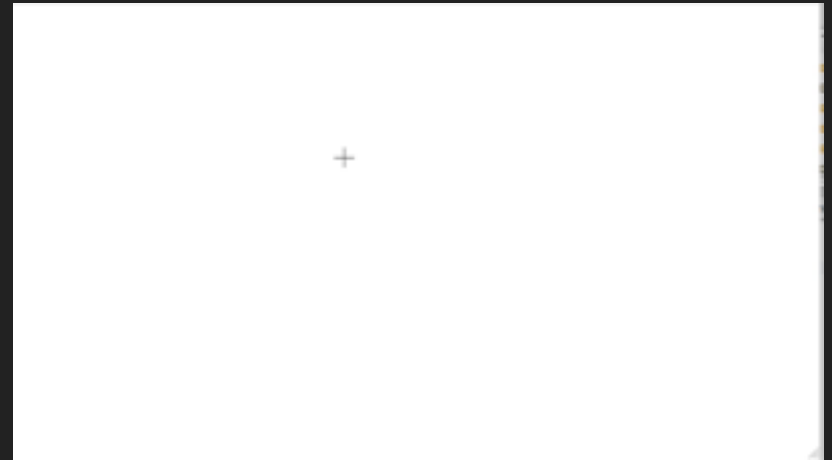
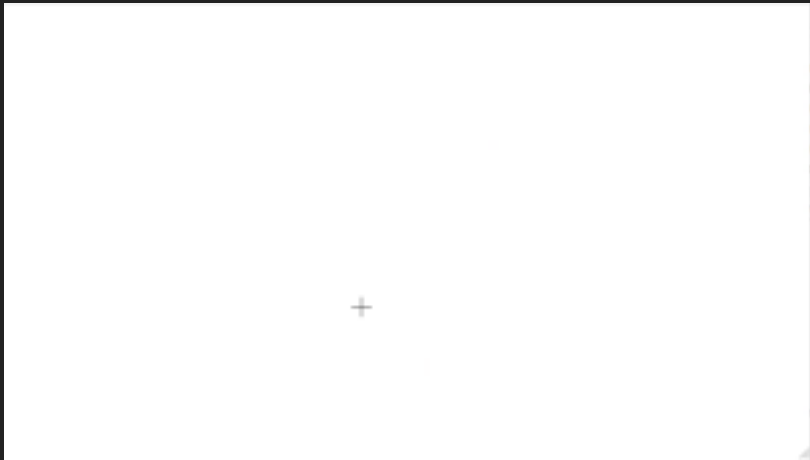
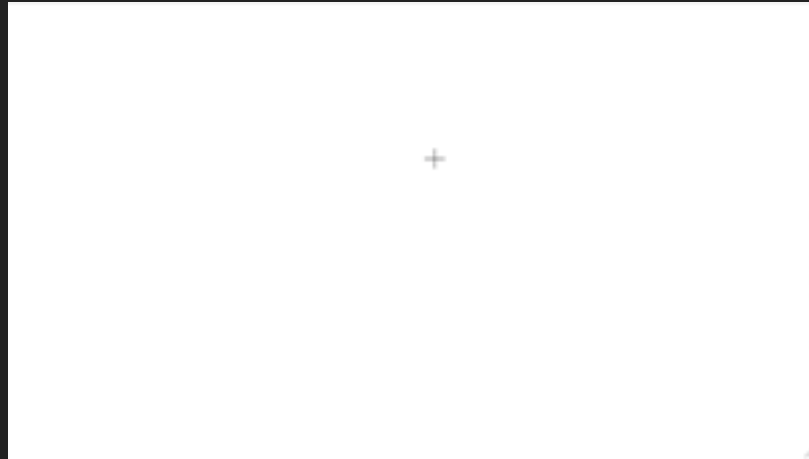


# Illustrations

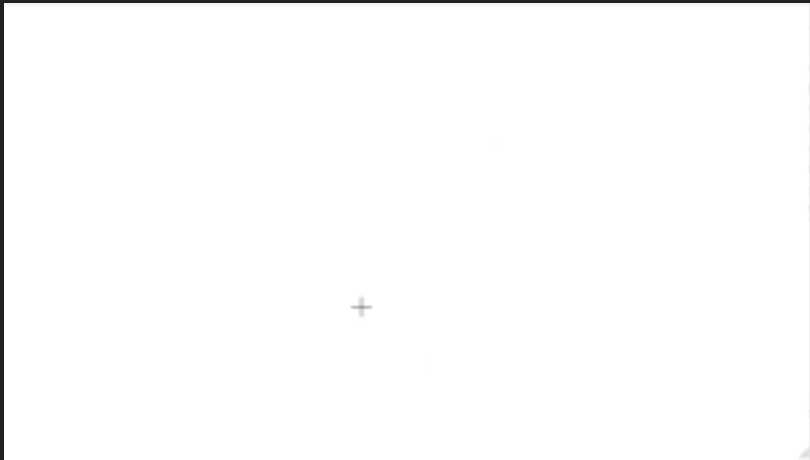
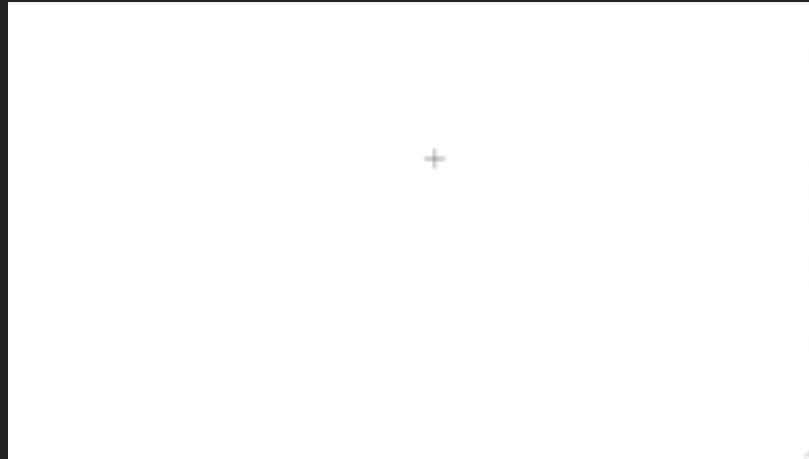
# Illustrations



# Illustrations



# Illustrations



# Summary

- We proposed directions both for programmers and end users to make stroke shortcuts be more widely used:
  - Dedicated tools to enhance existing toolkits
  - Contextual help to discover and learn shortcuts
- While we illustrated them in the context of 2D graphical interfaces, it is worth studying variations of them in a 3D context (e.g. mid-air gestures).