# Motion-Pointing: Target Selection using Elliptical Motions

**Jean-Daniel Fekete**[1] jean-daniel.fekete@inria.fr
**Niklas Elmqvist**[2] elm@purdue.edu
**Yves Guiard**[3] yves.guiard@enst.fr
[1]INRIA Saclay, France
[2]Purdue University West Lafayette, USA
[3]TELECOM ParisTech - CNRS LTCI Paris, France

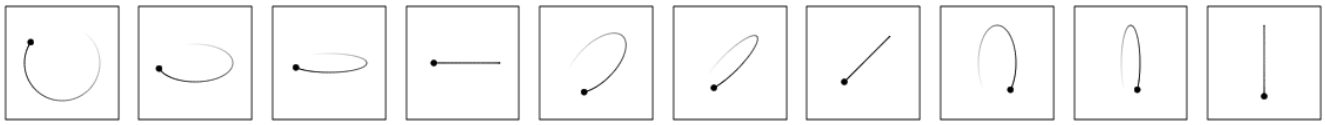


**Figure 1. Sample of some of the cyclic elliptical motions used as the basis for the motion-pointing method.**

## ABSTRACT

We present a novel method called *motion-pointing* for selecting a set of visual items, such as push-buttons or radio-buttons, without actually pointing to them. Instead, each potential target displays an animated point we call the *driver*. To select a specific item, the user only has to imitate the motion of its driver using the input device. Once the motion has been recognized by the system, the user can confirm the selection to trigger the action. We consider cyclic motions on an elliptic trajectory with a specific period, and study the most effective methods for real-time matching such a trajectory, as well as the range of parameters a human can reliably reproduce. We then show how to implement motion-pointing in real applications using an interaction technique we call *move-and-stroke*. Finally, we measure the input throughput and error rate of move-and-stroke in a controlled experiment. We show that the selection time is linearly proportional to the number of input bits conveyed up to 6 bits, confirming that motion-pointing is a practical input method.

### ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces— *Interaction styles*; I.3.6 Computer Graphics: Methodology and Techniques— *Interaction techniques*

### Author Keywords

Oscillatory motion, harmonic motion, alternative input.

## INTRODUCTION

Pointing is currently a central task in modern computer interfaces, and considerable amounts of research have been dedicated to speeding up and improving the accuracy of the 2D pointing task [1]. However, whereas most such techniques are aimed at standard computer systems with high-resolution input and output devices, few of them perform well in settings where there is no pointer or where pointer movement is very costly. Consider a large public screen in an airport displaying a list of flights: how can we select one specific flight with our smart-phone to get details? The same can happen when entering a supermarket with featured items displayed on a public screen: how can we get details about one item using a PDA or a smart-phone when the screen is out of reach?

We propose a technique based on stationary cyclic motion of the input device to select and activate visual items on the display (Figure 1). Humans are apt at reproducing cyclic gestures with reasonable accuracy and ease when trained [4, 15]. Instead of requiring users to reach and acquire the item with a pointer, we provide a grammar of oscillatory motion gestures that users simply mimic with their input device in order to select the desired item. Compared to standard vocabulary-based gesture interfaces [23], our *motion-pointing* method provides more visibility of the available commands and requires no visible cursor, instead relying on the proprioceptive feedback from the user's own body, but at the cost of increased visual load.

Several issues need to be addressed to make motion-pointing practical. First, we discuss the background of computer input and the existing literature. We then experimentally investigate the capabilities of humans to mimic cyclic gestures, and describe algorithms to recognize such gestures. We describe the interaction technique we have designed to apply motion-pointing to real applications, followed by a study of its effectiveness using a controlled experiment. We conclude the article with a discussion and our plans for future work.

## BACKGROUND

Computer input boils down to forced choices, with the user's task typically consisting of selecting one individual element within a finite set of elements. While some of these sets have a constant or nearly constant size, such as the set of alpha-


*CHI 2009*, April 5–9, 2009, Boston, MA, USA.

numeric characters (text entry) or the set of screen pixels (bitmap drawing), others have been inflating over the last two decades, like the alarmingly increasing number of available user commands in modern applications [3]. Still other sets (e.g., all websites on the Internet) have become so huge that they could be considered virtually infinite. This evolution raises a challenge for human-computer interaction: better and more diversified selection tools are needed.

In this paper we focus on another, perhaps more basic aspect of the selection issue: the elemental mechanism by which the system knows that the user is selecting a particular item rather than any other element of the set. Selecting an element is done using the input device to produce certain events that *match* a unique element within the set, presuming some sort of one-to-one mapping. The match can take several forms. When entering a command line on the keyboard, the user is counting on a *semantic* kind of match between the string of characters entered by the user and a certain target command, leveraging the fact that each individual command has a unique name. In the context of a graphical interface, obviously most selections rely on pointing, which amounts to a *spatial* kind of match. Each graphical object, whether a button, a menu item, or a hypertext link, has its unique spatial location. Hence if the coordinates of the screen cursor, which is under user control, coincide with those of one particular graphical object in 1D, 2D, or 3D space, then the user is understood to want to select this object.

### Traditional Input Channels

It is a fact that in today's computer interfaces, virtually all selection techniques resort to the principle of either semantic or spatial coincidence. Therefore, much of the research in HCI has been directed towards improving and extending such techniques. The morphological design space of input devices presented by Card et al. [5] is useful for reasoning about the (mainly spatial) input devices in this category.

For the spatial pointing task, human performance is traditionally modeled by Fitts' law [7, 17], and much research effort has been directed towards "beating" Fitts' law in various ways [1]. These techniques are largely outside the scope of this paper; representative examples include [6, 10, 18].

For keyboard input, the mapping between device events and commands is semantic in nature, yet the input events are created through spatial matches in motor space, i.e. by pressing the appropriate key (as opposed to the mouse, where the spatial match is performed in visual space). Much research has been conducted on keyboards; see any introductory HCI textbook. More recently, researchers have studied the practice of keyboard shortcuts, where the motor spatiality of the keyboard is harnessed to its fullest, and how to speed up the learning of the semantic matches of each key [11].

### Alternative Input Channels

Whereas traditional input channels typically make use of semantic or spatial matchings, there exist techniques based on quite different principles. These alternative input channels are interesting because they often have properties that make them useful in specific platform or user contexts.

Gesture-based interfaces [23] allow for invoking actions using arbitrary motion gestures performed with the input device. This is a semantic mapping to elements because the available gestures comprise a static vocabulary. However, gesture-based interfaces are plagued by low visibility and feedback and thus require learning. These drawbacks are improved by marking menus [16] and the OctoPocus technique [2], yet these techniques (and related ones) are designed to be complementary to the standard spatial pointing task, limiting their applicability outside this context.

One example of a particularly innovative modality is the rhythmic menu [19], which is based on temporal (or, rather, phase) coincidence with the target element. With this technique, a set of items are successively shown at the same location, each with its unique phase in the periodic display. To select an item, the user performs a pointing action in time rather than in space. One interesting property of this technique is that it requires no screen cursor because of the proprioceptive feedback from the user's own hand: the user *feels* the motion and needs no visual feedback.

More recently, Williamson and Murray-Smith [26, 25] introduced a fascinating new selection principle based on reproducing motion using the input device. In their main demonstration of this principle, the authors caused a number of graphical objects to erratically drift, at slow velocities, across a visual display. Since every single object has a unique velocity vector at any single instant, the user can select a specific object by simply mimicking the motion of that object using the input device. With this selection technique, just as with rhythmic menus (and for the same reason), the screen cursor becomes irrelevant. Note also that this method, unlike the aforementioned ones, is continuous, with the information gained by the system gradually accumulating. But the most interesting aspect of this technique is that it introduces an entirely novel principle of selection based on spatio-temporal, or *kinematic* coincidence.

## MOTION-POINTING

The present study capitalizes on the work of Williamson and Murray-Smith. We explore one special case of selection by motion coincidence that we believe has a special potential for input in HCI, namely selection by *periodical* motion coincidence, or *oscillatory coupling*. Unlike a rhythmic menu, which displays its items periodically at the same location and with a rhythm characteristic of the display, our proposed selection principle consists of having each of a set of graphical objects display an oscillating pixel with a unique frequency so that users can specify one element of the set by rhythmically coupling their input device with that pixel. In this paper, we present theoretical arguments and experimental data to suggest that selection based on periodical motion coincidence is not only workable, but promising, meaning that it is a candidate for incorporation in the toolbox of HCI design.

More specifically, the basic idea of motion-pointing is that the user selects a particular item out of a larger set of potential items by using the pointer to mimic the unique cyclic motion of a small *driver*. Each potential item has a unique driver attached to it. The motion of this driver follows an el-

liptic trajectory described by the following five parameters:

1. a period $T$ (inverse of a frequency $F = 1/T$);
2. an angle $\alpha$ from the horizontal to the ellipse major axis;
3. an amplitude $A$ on the major axis;
4. a ratio $R$ in the range $[0,1]$ between the amplitude on the major axis and the amplitude on the minor axis; and
5. a direction $D$ (clockwise or counter-clockwise).

Many drivers, each with different parameters, can be active at any moment on the screen. Recognizing the target driver consists of finding the best match between the user motion and the active drivers as quickly and accurately as possible.

### Why Oscillatory Motion?

Most physical materials have at least some elasticity, and so virtually all matter is subject to vibration, if only over small amplitudes [9]. Living things are no exception, biological movement more often than not taking the form of oscillatory motion. The ubiquity of oscillatory forms of human motor behavior, from fairly automatic movements like breathing through the more voluntary movements that make up the gait, up to the most sophisticated cases of speech and handwriting, has been often emphasized [15].

Not only do humans exhibit an impressively diverse repertoire of oscillations in their behavior, but simple harmonic motion—the form of periodical motion that minimizes the dissipation of mechanical energy by recycling potential energy into kinetic energy and vice versa—is something people do most naturally. It has been experimentally shown, in the context of a reciprocal (i.e. cyclical) pointing task, that the easier the targets (and hence, by Fitts' law [7], the shorter the movement period), the closer the oscillation of the pointer to a pure sine wave [12]. In fact, the recorded kinematic traces of participants do not depart from simple harmonic motion *unless* severe enough spatial constraints (i.e., narrow enough targets) are imposed by the experimenter on the pointing movement. The lesson here is that simple harmonic movement of the hand is both the easiest to perform and the most economical in terms of effort. From this evidence we can conjecture that a method like ours has a future in HCI.

Another relevant fact regarding periodic movement in humans is the ability of people to couple themselves with an external oscillator [4]. Interesting resemblances in this regard can be found between humans and simple mechanical systems. It is well known that two pendular clocks attached to the same wall tend to synchronize in anti-phase [22]. Many similar phenomena, and most notably phase and frequency locking effects, have been observed in humans asked to produce hand oscillations in a variety of situations where the driver is a metronome [4], the participant's other hand [13, 15], or even the hand of another participant [20, 24].

## FORMATIVE USER STUDY

The motivation for our initial formative user study was to determine the validity of reproducing elliptical motions as a general mechanism for selection. A second motivation was to find a suitable matching function for distinguishing different motion trajectories from each other.

Assuming the method to be valid, we were also interested in exploring its limits, such as the ranges of comfortable amplitudes, frequencies and angles, and the time needed to reach a steady state. This data can then be used for designing an actual interaction technique based on motion-pointing.

### Apparatus

The test was conducted on a standard desktop PC running the Microsoft Windows XP operating system. The display was an LCD monitor with a resolution of $1280 \times 1024$ pixels and a refresh rate of 64Hz. We used a standard Microsoft optical wheel mouse as the motion-pointing input device. The motion recording software was set to full-screen mode.

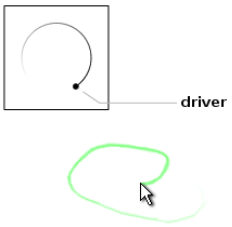



**Figure 2. The motion recorder software showing a driver and the pointer trace (this trace was not visible during the experiment).**

### Participants

We enlisted the help of six participants (2 female, 4 male) for performing this experiment. All participants were experienced computer users. Ages ranged from 23 to 50 years. Participants all had normal or corrected-to-normal vision, were not color-blind, and were right-handed.

### Task and Procedure

Participants were asked to use the mouse to move the pointer in such a way that it mimicked the motion of a single driver centered on the screen (Figure 2). They were to follow the motion of the driver as faithfully as possible, including amplitude (size of the movement), frequency, phase, slope, and shape of the driver's trajectory. Participants were told not to superimpose the mouse cursor with the driver; if they approached the bounding box of the driver, the cursor became invisible to further discourage this practice. The mouse cursor was visible at all other times.

Each trial lasted exactly ten seconds, and was preceded by an intermission screen where users were allowed to rest from the previous trial and prepare for the next. Participants were asked to click twice on a red square (the first click turned it green) located halfway between the perimeter and center of the screen to proceed to the next trial. This ensured that the mouse pointer was in a known position far away from the edges of the screen and the bounding box around the driver.

Prior to conducting the actual experiment, users were given a demonstration of the recording software and of the trial to perform. They were allowed to practice with sample trials until they indicated that they felt ready to proceed with real

trials. They were told that they could rest at any time between trials, and also that they could interrupt a session and resume at a later time (even in a different day) when they became fatigued. The experiment administrator also looked for signs of fatigue, and suggested that the participant take a break in case such were spotted. Despite this, all participants chose to record their performance in one session of approximately 1 hour and 20 minutes (breaks included).

### Experimental Conditions and Design

In accordance to the theoretical background presented above, we chose to investigate the following factors:

- **Frequency** (*F*) Speed of the oscillatory motion of the driver. [1 Hz, $4/3$ Hz, 2 Hz, 4 Hz]
- **Amplitude** (*A*) Size (radius) of the driver's trajectory. [5 pixels, 10 pixels, 20 pixels]
- **Aspect ratio** (*R*) Ratio between the minor and major axes of the motion ellipse. [0, 0.3, 0.6, 1.0]
- **Angle** ($\alpha$) Angle between the major axis and the horizontal axis. [0, 45, 90, 135]
- **Direction** (*D*) Direction of the driver's motion. [clockwise, counter-clockwise]

We built our experiment as a full-factorial within-subject design. Given this, we ended up with 384 trials per session. With six participants, we recorded a total of 2304 trials. Note that when the aspect ratio is 1, the trajectory becomes a circle and the angle becomes irrelevant, and when the aspect ratio is 0, the direction is irrelevant, so the number of visibly different motions is reduced to 264.

For each trial, we collected time-stamped samples of driver and pointer positions at the screen refresh rate (64 Hz). This means that each trial was recorded as a table with 5 columns (Time, Driver X, Driver Y, Mouse X, Mouse Y) and 640 lines of observations for each of these variables.

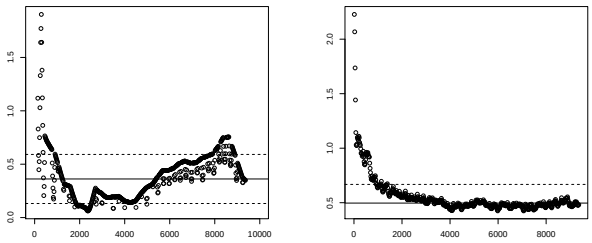
### Results

We performed two kinds of analyses on our collected data: spatial and temporal. For the spatial analysis, we considered the trajectories as just clouds of points, ignoring the time. For the temporal analysis, we simplified the spatial information to disregard shape effects as much as possible. All participants showed consistent trends in the analyzed data.

#### *Spatial Analysis: Moving-Window Ellipse Fitting*

For each trial, we fitted ellipses on the trajectory with a moving window of 100 points [8]. The ellipse fit returns the amplitude, aspect ratio, angle $\alpha$, and ellipse center. It does not provide information regarding the frequency or the direction. We also computed the standard deviation from the fitted ellipse to the trajectory, i.e. for each point, we calculated the shortest distance to the fitted ellipse [14], and from these distances derived the standard deviation (Figure 3(a)).

Finally, we computed the standard deviation in angle (Figure 3(b)), amplitude (Figure 3(c)) and aspect ratio (Figure 3(d)) for each trial and averaged for all trials (Figure 3).




(a) Phase error for one trial. (b) Average phase error.

**Figure 4. Phase error between driver and pointer over time (ms).**

#### *Temporal Analysis: Moving-Window Phase Error*

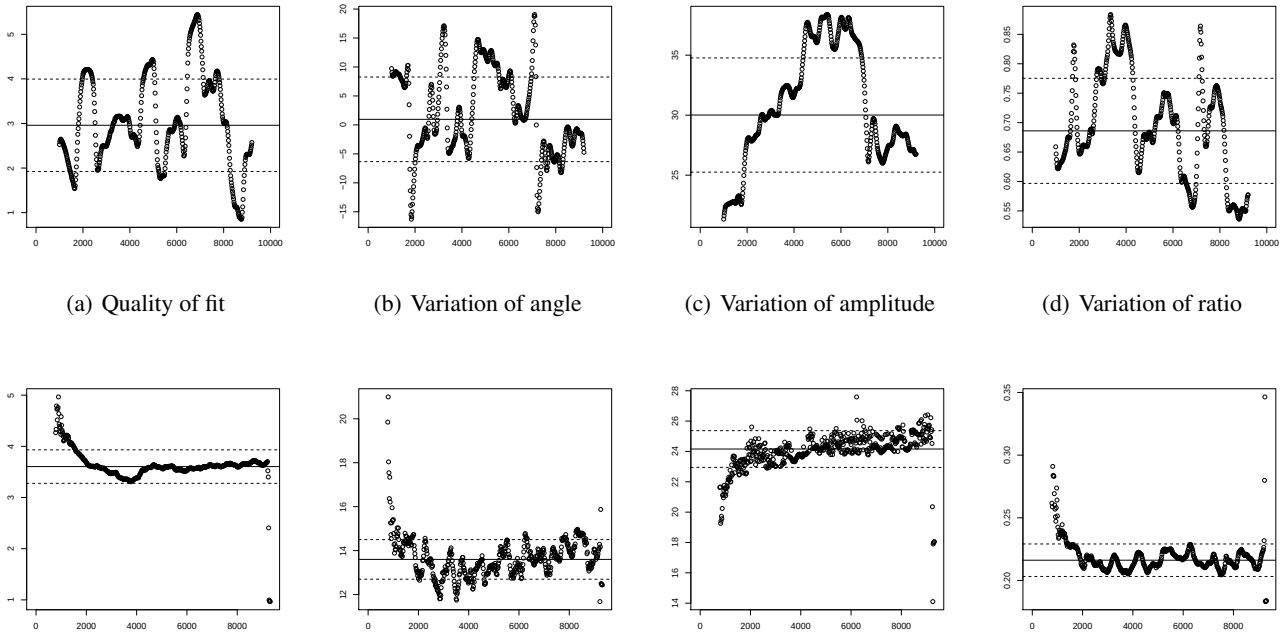
For each trial, we computed a phase difference between the coordinates of the driver and the pointer. The phase of a 1D signal (such as the X coordinate of the mouse) is $+1$ at each local maximum, $-1$ at each local minimum, and a linear interpolation in-between. This serves as a useful abstraction for the signals. The phase difference between the mouse and the driver is therefore the distance between the X/Y phase of the driver and the X/Y phase of the pointer.

#### *Implications for Design*

We notice three very clear phases in the data: an expected initial phase when the user is observing the motion and trying to synchronize with it (about 1-1.5 second), a second phase when the user is able to synchronize, or become *coupled* with the motion (about 1-2 seconds in duration), and a third phase when the synchronization is lost and sometimes, but not always, recovered. This third phase may be due to fatigue or to the lack of feedback. Moreover, the coupling phase is consistent with research in psychology [4] and constitutes the foundation upon which motion-pointing stands.

Figure 3(e) shows that the quality of the fit varies from 1 to 10 pixels and on average improves with time. There is an optimal "quality" after 2 seconds where the *steady state* is reached. Figure 3(f) shows that the angle becomes stable with a standard deviation of around 15 degrees for all users, so angles separated by 30 degrees should be recognizable.

Figure 3(g) shows that the standard deviation of the amplitude is much larger than the amplitude itself, so this parameter is not usable for matching. Figure 3(h) shows that the standard deviation of the aspect ratio is about 0.25, so three ratios could be distinguished in principle: 0, 0.5 and 1. However, with a ratio of 1, the angle cannot be used so using 0 and 0.5 is likely to yield better accuracy for the matcher. Moreover, when the ratio is 0, the angle becomes twice more accurate, around 7 degrees. Therefore, there is a tension between using ratios of 0 and 0.5, or only 0. The former provides more potential motions, but decreases the resolution of the angle. The second limits the motions to straight lines and the direction parameter becomes meaningless. During the post-study interview, subjects consistently expressed that lines were much easier to mimic than the rounded shapes.



(a) Quality of fit (b) Variation of angle (c) Variation of amplitude (d) Variation of ratio

(e) Quality of fit averaged (f) Variation of angle av. (g) Variation of amplitude av. (h) Variation of ratio av.

**Figure 3. Results of fitting an ellipse on a moving time-window (time in milliseconds on the horizontal axis).**

*Matching Motions*

From these observations, we are looking for a matching function that is fast and improves with time but is also robust against the decrease of accuracy visible after 2-4 seconds. Without loss of generality, this matching function can be expressed as a distance or dissimilarity function $f(u,d) = s$ where $u$ is a user motion and $d$ is a driver motion. The parameter $d$ can be expressed as a trajectory or as a parametric function yielding timed positions. When computing matches, we should take relative movement into account since absolute positions of the pointer is by design not interesting. Therefore, we regard the subject trajectory as a list of $(\delta x, \delta y, t)$ that are synchronized with the driver motions.

The simplest possible method of using the matching function is to study the integral of distances during a time window $w$. We experimented with the following distance functions:

Let $u$ be a user motion, $ux$ the $x$ delta component of the user motion (analogously, $uy$ for the $y$ delta component), $d$ a driver motion (where $dx$ and $dy$ are the $x$ and $y$ delta components for the driver), $t$ the current time, and $w$ the size of the moving time window.

**Euclidean Distance**

$$f(u,d) = \sum_{i \in [t-w,t]} ((ux_i - dx_i)^2 + (uy_i - dy_i)^2) \quad (1)$$

**Normalized Euclidean Distance**

$$f(u,d) = \sum_{i \in [t-w,t]} (\frac{ux_i}{||u_i||} - \frac{dx_i}{||d_i||})^2 + (\frac{uy_i}{||u_i||} - \frac{dy_i}{||d_u||})^2 \quad (2)$$

**Correlation** (Not a distance but a similarity, so large values are better.)

$$f(u,d) = \sum_{i \in [t-w,t]} ((ux_i \times dx_i) + (uy_i \times dy_i)) \quad (3)$$

We applied these functions to the recorded motions by computing the distance between each pointer and driver motion for all trials ($384 \times 384$ distances computed per participant). In principle, with a good distance function, the driver associated with a motion should be at a shorter distance from that motion than to any other driver motion. So for each user motion, we sorted the driver motions by increasing distance (decreasing for the correlation) and looked at the rank of the expected driver. A rank of 0 is a perfect match, a rank of 1 is a good match, etc. The objective function we used to assess the quality of the distance function was the count of motions ranked below 4. For all the users, the normalized distance function was almost always better than the Euclidean distance and the correlation came last. Therefore, we chose the normalized distance matcher with a 2-second time window.

## THE MOVE-AND-STROKE TECHNIQUE

Move-and-stroke is our proposed interaction technique that makes use of motion-pointing as the underlying mechanism for selecting items. It displays a driver for each clickable

item and overlays semi-translucent matching information on top of the items. Because motion matching is typically not 100% accurate, as the previous experiment showed, move-and-stroke displays the top four matched motions and lets the user perform a mouse stroke to discriminate between these four using a dynamic pie menu.

### Motivation and Requirements

The main underlying motivation for designing an interaction technique based on motion-pointing in the first place is the case for supporting pointing and selection of items in settings where there is no pointer, or where pointing is very costly, such as for area-limited input devices or out-of-reach displays. Given this background, we summarize the guiding requirements for the design of the move-and-stroke technique:

R1 **Fixed gaze.** Unlike for normal pointing techniques, the new technique should allow users to fix their gaze on the desired target and not require them to split their attention between the target and the pointer. We also want to avoid relying on the presence of a pointer. While the proprioceptive feedback of motion-pointing is sufficient for accurate motion reproduction, this also means that we must display all information in the vicinity of the potential targets.

R2 **Visibility and learnability.** Gesture-based interfaces have the intrinsic weakness of low visibility and often require considerable training. To sidestep this issue, we want to explicitly show the available motions and to bypass the need for training entirely.

R3 **Matcher robustness.** Motion matching is an imprecise process, so the new technique should be robust against matching inaccuracy. More specifically, it should support the desired target motion being matched among the top candidates, not necessarily the highest-ranked one.

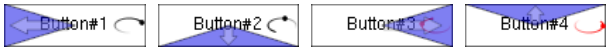
R4 **Simple and intuitive activation.** Motion matching cannot be constantly active lest it causes false triggering of undesired items. Therefore, the technique needs to be modal, and it should be activated using a simple action that conforms to the spirit of its design.

R5 **Cancellation.** Due to the possibility of false triggering or inaccurate matching, users should be able to easily cancel an initiated move-and-stroke operation.

R6 **Low visual clutter.** Cluttering the display with too much extra information or animation will make the screen busy and may be distracting and detrimental to performance.

### Design Rationale

Move-and-stroke displays animated drivers, each describing a unique motion, overlaid in the corner of each potential target, eliminating the need to shift the gaze from the desired target (R1). Graphical lines for each driver show parameter traces of the motion. These visual drivers give visibility and eliminate the need for practice when using the technique (R2), but increase the visual load of the display (R6). To further help visibility and predictability, static (i.e. not animated) icons of the drivers may be shown when the technique is in idle mode.




(a) West (b) South (c) East (d) North

**Figure 5. Semi-translucent triangle quarters indicating the top four motion matches. These quarters become a four-item pie menu when the user transitions to the next mode (by clicking).**

At any point in time, the inaccuracy of our motion matcher as well as the motoric skills of the user may result in the user's desired target not being ranked as the top candidate. To provide some robustness against this situation (R3), move-and-stroke indicates the current **four** top-ranked drivers by overlaying semi-translucent highlights on top of their respective targets. The highlights are drawn as triangular quarters in one of the four compass directions (Figure 5).

These quarter highlights, along with a white arrow, indicate the stroke direction for activating the specific target. Clicking the mouse button at any time stops the motion-pointing process and turns the four target candidates into a pie menu (a split pie-menu with four centers, one for each candidate). Releasing the button again within a certain dead zone area, or clicking a different button, cancels the activation (R5). Performing a mouse stroke, on the other hand, activates the candidate target in the compass direction of the stroke.

Finally, move-and-stroke is modal to avoid false triggering. Activating the motion matching is done by performing a quick, circular motion with the mouse. Note that no mouse click is required to trigger the motion matching, so the technique should not interfere with other interaction techniques. This circular motion is also a simple action that it is typically not used in natural settings, thus avoiding activation by mistake (R4). At the same time, it is in line with the operation of the technique since it is relatively easy to transition from the circular activation into reproducing a driver's oscillatory motion. Furthermore, because we show static icons of the drivers even when the technique is in idle model, the user can predict the motion of the desired driver and initiate the active mode accordingly.

### Interaction Example

Figure 6 gives an example of using move-and-stroke to select the ninth button out of a set of 16 buttons. Figure 6(a) shows the initial situation with move-and-stroke in idle mode. Note that static motion icons are shown for each potential target, giving the user an indication of the motion to reproduce in order to acquire it. Initiating the tracking mode using a quick circular motion results in the situation depicted in Figure 6(b). As the user attempts to replicate Button 9's driver motion, the blue triangles highlight the current top matches. Stopping the motion at this point will cause the technique to go back into idle mode after half a second of inactivity.

Satisfied with the intended target being in the top four, in Figure 6(c) the user has clicked the mouse button and progressed into pie selection mode. All that remains is for the user to drag the mouse in the compass direction indicated by the arrows, turning Button 9 green, and then releasing the mouse button to activate the selection. Arrows originating

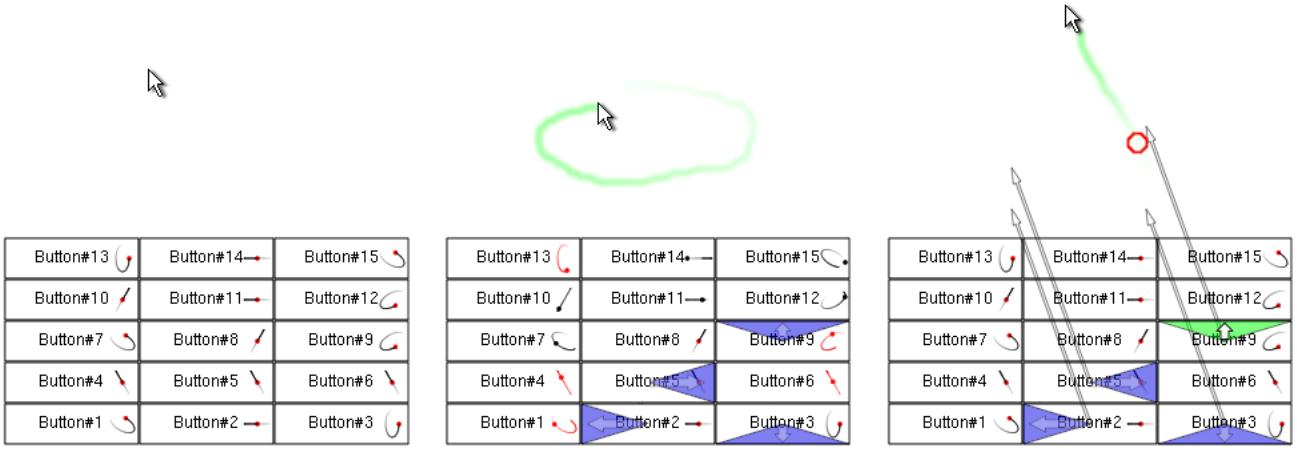



**Figure 6. Sequence of screenshots showing move-and-stroke in action: (a) Idle mode with static motion icons for each potential target; (b) Tracking mode after the user has initiated the technique, the blue triangles indicate the top four matching motions; (c) Pie selection mode with visible mouse strokes after clicking (the red circle indicates the click position and the arrows the directions). Note that the green mouse traces do not appear in the real technique.**

from the center of each of the four candidates give visual feedback on the current direction of the mouse stroke.

### Instantiating the Parameter Space

Actually implementing move-and-stroke requires one additional step: the creation of the motions that the user needs to reproduce to select a visual item. It is clear that each individual item must have a unique motion associated with it in order for this to be feasible. Furthermore, the motions should be as different from each other as possible to facilitate easy motion reproduction and correspondingly easy matching.

We used the results from our formative motion-pointing study to inform the motion creation process. Our approach relies on evenly dispersing candidate motions across the entire parameter space; adding or removing a set of motions thus requires recomputation of the whole set of candidate motions. This is acceptable because we always show visual indications of the motions to reproduce and thus do not require users to learn the motions mapped to individual items.

The information derived from our formative study, such as the maximum and minimum comfortable (and practical) frequencies, gives us the limits of the parameter space. The analysis of the variance for each parameter in the study allows us to rank the discriminating power of each parameter accordingly; for instance, angle is more accurate than the ratio of the motion, and amplitude is too inaccurate to be used, so is disregarded entirely.

### Implementation Notes

We implemented move-and-stroke in a Java using the JOGL OpenGL bindings for efficient rendering because Java/Swing rendering was not fast enough. The design of JOGL enables OpenGL buffers to be overlaid on top of standard Java windows, thus allowing to integrate move-and-stroke with existing Java applications.

## SUMMATIVE USER STUDY

We designed this experiment to evaluate the performance of the move-and-stroke interaction technique when selecting a particular target from a set of potential items. We were particularly interested in experimentally deriving the throughput [17] of the technique, as well as finding the error rate and pinpointing potential problems with the technique and with the use of motion-pointing for selection in general.

### Apparatus

The test was conducted on a laptop PC running the Microsoft Windows XP operating system. The display was an LCD monitor with a resolution of $1920 \times 1200$ pixels and a refresh rate of 64Hz. We used a standard Microsoft optical wheel mouse as the motion-pointing input device.

### Participants

We enlisted the help of six participants (1 female, 5 male) for performing this experiment (all different than for the formative experiment). All participants were experienced computer users. Ages ranged from 23 to 45 years. Participants all had normal or corrected-to-normal vision, were not color-blind, and were right-handed.

### Task and Procedure

The task consisted of using the move-and-stroke technique to select a target button out of a set of buttons displayed on screen. Buttons were numbered (Figure 6 shows the setup used in the experiment), and the button to select was referred to both using its number prior to each trial as well as by highlighting the button in yellow. Trials were interleaved by intermission screens where the participant could rest and prepare for the next trial. During this time, the participant also returned the mouse pointer to a neutral position.

Selecting a target button consisted of initiating the move-and-stroke technique, acquiring the correct target by repro-

ducing its driver motion, and then performing a pie-menu stroke to select it. Participants were required to repeat this process until they selected the correct button. The pointer was visible at all times, but participants were instructed to focus their gaze on the target.

Because we were interested in measuring the input throughput of the move-and-stroke technique, we varied the number of buttons simultaneously displayed on the screen. This number served as a measurement for the number of bits the participant was forced to communicate to the technique in order to select the correct target.

Prior to conducting the actual experiment, participants were allowed to practice the use of the move-and-stroke technique. The training phase lasted approximately ten minutes and involved selection of targets in 2-button as well as 8-button scenarios. Participants were allowed to repeat the training until they felt comfortable with the technique.

#### Experimental Conditions and Design

Beyond the motion parameters for the target driver, we involved only one factor in our evaluation: the number of bits ($B$) of information to convey so that $2^B$ gives the number of buttons on the screen (we studied 1, 2, 3, 4, 5, and 6 bits).

Dependent variables were time and error measurements for each phase of the interaction: Idle→Tracking (IT), Tracking→Selection (TS), Selection→Idle (SI).

Trials were arranged in blocks given by the number of bits. Participants selected each target once for each block. Given the above set of bits, each participant performed 126 trials. With 6 participants, we collected time and error measurements for 756 individual trials in total.

Blocks were presented sequentially in order of increasing number of bits (i.e. starting with 2 buttons, then 4, 8, etc). This was done intentionally to allow learning effects to help participants prepare for the next and more difficult block. We did this because we were interested in finding the maximum performance of reasonably trained users, not in comparing the technique to competing techniques.

#### Results

Performance results from the experiment are summarized in Figure 7. We performed a repeated-measures ANOVA and found a number of significant main effects.

Figure 7(a) gives the distribution of the tracking time (TS) as a function of the number of bits. We found a significant effect of bits on tracking time ($F_{5,5} = 150.33, p < .001$). IT (average 0.91 s) and TS (average 0.05 s) showed no significant effect of bits. We also found significant main effects of angle ($F_{4,5} = 19.44, p < .001$) and ratio ($F_{1,5} = 55.61, p < .001$), but no effects of direction or frequency on TS.

Figure 7(b) shows the average tracking time as a function of the number of bits (error bars show standard deviations). Disregarding the first two conditions (two and four buttons), where the task reduces to pie-menu selection, it appears that the time progression for bits 3 through 6 is linear.

Finally, Figure 7(c) shows the mean correctness for all trials as a function of the number of bits. Again disregarding the first two bits, it appears we have a slowly, perhaps linearly, decreasing rate of correctness. We explain this by the imperfections in our matcher for larger number of potential target motions. Interestingly enough, ratio had a significant impact on tracking performance, with lines ($R = 0$) being significantly easier to match than ellipses ($R = 0.5$). Again, we believe this could be fixed by improving the motion matcher.

## DISCUSSION

We have performed two empirical studies, one formative and one summative, to evaluate the motion-pointing method and the move-and-stroke technique. Here are our main findings:

- Humans are apt at reproducing oscillatory motions using a mouse, sufficiently so for motion-pointing to be a viable alternative input channel; and
- The move-and-stroke technique supports a target acquisition time proportional to the number of bits to convey.

#### Explaining the Results

Our results from both studies confirm earlier findings from psychology [4] on humans being apt at coupling to harmonic motions. In particular, this coupling can done with no visual feedback, relying solely on the proprioceptive feedback from the user's own hand; none of participants reported that they felt a need to focus on the pointer during the experiments.

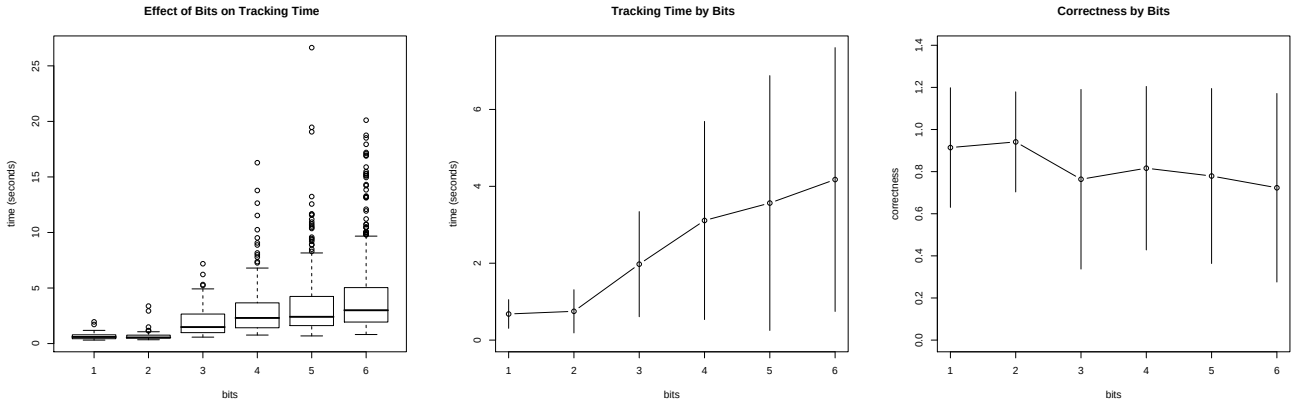
As discussed earlier, we found that lines were more easily matched than ellipses in the summative study. While this may be due to imperfections in the matcher, some of our participants reported that they found it difficult to match the shape of the elliptical driver trajectories using a mouse. They noted that the use of a stylus or a touchpad might have made such paths easier to reproduce. This may also explain the difference in performance for the different ratios and angles.

One concern we had regarding move-and-stroke was that having several animated motions simultaneously visible on the screen would distract the user and cause deteriorated performance. However, while we did see performance drops, participants did not indicate distraction to be the cause; rather, we explain the decreasing performance for higher target numbers with the limited parameter space and motion matcher. People seem to be apt at replicating the motion of individual drivers even when the driver in question is surrounded by other drivers exhibiting other motions.

#### Designing Motion-Pointing Techniques

A number of tradeoffs and implications regarding the design of motion-pointing apply. In this subsection, we discuss some of the design decisions that were made and their implications for the motion-pointing method as a whole.

*Visual vs. acoustic driver:* It has been experimentally shown that synchronizing oneself with an external oscillator is easier if the rhythm is auditory rather than visual [21]. However, choosing between multiple potential targets requires many rhythms to be presented simultaneously. Such a parallelism is impossible in the auditory modality because a set of acoustical sources would be perceived as confusing noise.




(a) Tracking time (seconds). (b) Average tracking time (seconds). (c) Average correctness (ratio).

**Figure 7. Time and correctness performance for matching the target as a function of the number of bits $B$ (error bars show standard deviations).**

*Continuous vs. discrete driver:* One interesting open question concerns the difficulty of synchronizing one's hand with a continuous harmonic oscillation in comparison with the case where the rhythm consists of discrete periodic event like, say, a flash or light or some abrupt positional shift in the visual display. More research is needed to determine how our technique may be optimized in this regard. However, we believe that the continuous oscillatory motion patterns chosen for our work serve a useful vehicle for promoting stability and aiding the rhytm reproduction, as well as increasing the distinguishing power of the motion matching algorithms.

*Between-person variability:* It is common experience (e.g., from dancing halls) that not everyone is capable of stable synchronization with an external rhythm. Obviously there is some between-individual variability in the picture, which means that the technique we are proposing may be problematic to a subset of the population. However, rhythmic ability is subject to improvement by training [27], and so our technique remains, in principle, accessible to anyone.

*Managing errors:* Move-and-stroke exhibits a relatively high error rate, which is a problem for a selection technique like ours. In general, our observations show that motion reproduction, for obvious reasons, will remain an imprecise input method. Further research is needed to investigate how to counteract this by, e.g., improving the matcher or the interaction mechanics of the selection technique.

### Applying Motion-Pointing Techniques

So far, we have only briefly touched on some of the applications where motion-pointing can come in useful: in specific contexts where there is no pointer, or where pointing is costly. It is often not technically difficult to introduce a pointer, even for a low-resolution input device. In such cases, relative pointing can come in useful (cf. joysticks or touchpads and touchpoints on laptops). However, there exist situations where displaying a pointer is not desirable. Consider a large display, such as in a museum or another public space, where several users are expected to interact with the display simultaneously (for example using their PDAs or iPhones). Displaying a pointer for each actor will cause a confused and cluttered display and will make interaction difficult. Alternately, in similar public spaces, displaying the pointer may even be undesirable due to privacy concerns.

Our move-and-stroke technique utilizes the mouse (or another spatial 2D input device), but we certainly believe that alternate input devices can be used for the same purpose. For instance, motion-pointing should allow for accurate selection even for low-cost gaze or marker-based tracking.

Furthermore, another strength of motion-pointing is that it does not interfere with existing input techniques or devices. This means that it can combined with other techniques and just add another modality to the interaction.

## CONCLUSIONS AND FUTURE WORK

We have presented a new mechanism for selection called motion-pointing that uses the alternative input channel of reproducing cyclic elliptical motions to discriminate a target item out of set of potential items. This method is useful for situations where direct pointing and acquistion is impossible or costly. The literature suggests that humans are apt at reproducing harmonic, rhytmical motions, and we have performed a formative study that confirms this fact. Building on this basic premise, we have designed an interaction technique called move-and-stroke that utilizes motion-pointing to allow participants to select items in real user interface settings. Results from a summative study evaluating the performance of move-and-stroke shows a linear increase in pointing time depending on the number of input bits to convey. We have also presented a number of applications where motion-pointing can come in useful.

This is our first attempt at harnessing the power of motion-pointing in a user interface context, but we believe that there are many potential uses beyond those outlined in this paper. In the future, we intend to improve the motion matching algorithms as well as the interaction mechanics in the move-and-stroke technique. We are also interested in exploring the use of motion-pointing for particular settings or user groups,

such as interaction with large, public displays, or for disabled users with limited physical reach or accuracy.


## ACKNOWLEDGMENTS

Thank you to the members of the AVIZ team for their feedback and comments on this work and for Évelyne Lutton for her help in analyzing the data. This work was partly funded by the Microsoft-Research/INRIA ReActivity project.


## REFERENCES


1. R. Balakrishnan. 'Beating' Fitts' law: virtual enhancements for pointing facilitation. *International Journal of Human Computer Studies*, 61(6):857–874, 2004.
2. O. Bau and W. Mackay. OctoPocus: A dynamic guide for learning gesture-based command sets. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2008.
3. M. Beaudouin-Lafon. Designing interaction, not interfaces. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 15–22, 2004.
4. P. J. Beek. The science of juggling. *Scientific American*, 273(5):92–97, November 1995.
5. S. K. Card, J. D. Mackinlay, and G. G. Robertson. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems*, 9(2):99–122, Apr. 1991.
6. A. Cockburn and A. Firth. Improving the acquisition of small targets. In *Proceedings of the British HCI Conference*, pages 181–196, 2003.
7. P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381–391, 1954.
8. A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least-squares fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, May 1999.
9. A. P. French. *Vibration and Waves*. CRC Press, 1971.
10. T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the ACM CHI 2005 Conference on Human Factors in Computing Systems*, pages 281–290, 2005.
11. T. Grossman, P. Dragicevic, and R. Balakrishnan. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the ACM CHI 2007 Conference on Human Factors in Computing Systems*, pages 1591–1600, 2007.
12. Y. Guiard. On Fitts' and Hooke's laws: simple harmonic movement in upper-limb cyclical aiming. *Acta Psychologica*, 82(1-3):139–159, 1993.
13. H. Haken, J. A. S. Kelso, and H. Bunz. A theoretical model of phase transitions in human hand movements. *Biological Cybernetics*, 51(5):347–356, February 1985.
14. J. C. Hart. *Distance to an ellipsoid*, pages 113–119. Academic Press Professional, 1994.
15. P. N. Kugler and M. T. Turvey. *Information, natural law, and the self-assembly of rhythmic movement*. Lawrence Erlbaum Associates, 1987.
16. G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *Conference Companion for the ACM CHI'94 Conference on Human Factors in Computing Systems*, page 218, 1994.
17. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139, 1992.
18. S. MacKenzie and S. Riddersma. Effects of output display and control-display gain on human performance in interactive systems. *Behaviour and Information Technology*, 13(5):328–337, 1994.
19. S. Maury, S. Athenes, and S. Chatty. Rhythmic menus: toward interaction based on rhythm. In *Extended Abstracts of the ACM CHI'99 Conference on Human Factors in Computing Systems*, volume 2, pages 254–255, 1999.
20. D. Mottet, Y. Guiard, T. Ferrand, and R. J. Bootsma. Two-handed performance of a rhythmical Fitts task by individuals and dyads. *Journal of Experimental Psychology: Human Perception and Performance*, 27(6):1275–1286, 2001.
21. A. D. Patel, J. R. Iversen, Y. Chen, and B. H. Repp. The influence of metricality and modality on synchronization with a beat. *Experimental Brain Research*, 163(2):226–238, May 2005.
22. K. M. F. Romero, S. G. Mokarzel, M. O. T. Cunha, and M. C. Nemes. Realistic decoherence free subspaces. *arXiv:quant-ph/0304018v2*, 2003.
23. D. Rubine. Specifying gestures by example. *Computer Graphics (SIGGRAPH 1991)*, 25(4):29–337, 1991.
24. R. C. Schmidt, M. Bienvenu, P. A. Fitzpatrick, and P. G. Amazeen. A comparison of intra- and interpersonal interlimb coordination: Coordination breakdowns and coupling strength. *Journal of Experimental Psychology: Human Perception and Performance*, 24(3):884–900, 1998.
25. J. Williamson. *Continuous Uncertain Interaction*. PhD thesis, Department of Computing Science, University of Glasgow, 2006.
26. J. Williamson and R. Murray-Smith. Pointing without a pointer. In *Extended Abstracts of the ACM CHI 2004 Conference on Human Factors in Computing Systems*, pages 1407–1410, 2004.
27. P. G. Zanone and J. A. S. Kelso. Coordination dynamics of learning and transfer: collective and component levels. *Journal of Experimental Psychology*, 23(5):1454–1480, October 1997.