Preliminary Updates & Scientific Reports: *Multimedia Research Group. MMG – Intelligent User Interface*.
Lehrstuhl Algorithmen & Datenstrukturen. Institut für Informatik. Albert-Ludwigs-Universität Freiburg.

# Identifying Tabletop Corners with Angular- and Velocity-Type Trace-Features in Pen-Gestures

Khaireel A. Mohamed

15 August 2005

**Preamble**

We present an effective corollary from our current works in pen-gesture technologies, originally intended for conventional digital screen environments, that gives significant influence when expanded into the domain of tabletop displays. By virtue of incorporating angular and velocity-type "trace-features" for pen-gesture cognition into current classical linear-classifiers, we are able to place a greater emphasis not on the similarity of the "visual representation" of the gestures, but rather, on determining exactly from which corners of the table the gestures were conceived. In other words, we can find out which one of the four users around the table is interfacing with the tabletop display at any particular time, and do this without the need for additional sensing paraphernalia. We find this especially useful when the hardware requirement is limited to the very basic, and when the digital tabletop displays offer only asynchronous input capabilities; i.e. only one anonymous digital pen input can be considered touching the screen correctly at any one instant. We illustrate this inception by discussing our treatment of "disoriented" pen-gestures with our version of a 'manual' Monopoly board game for the tabletop.

## 1 Introduction

We have yet to find a multi-user interaction model developed for the tabletop displays that handles the notion of "*who*-is-doing-what-and-when-and-where", without having to rely on external sensing devices or imposing hardware-wearables on the users. The support mechanisms in place for current tabletop systems stress considerably more, among others, on ubiquitous table rules for appropriately displaying and orienting screen-items (widgets), effective usage guidelines for practical UI supports, and convenient tools for multiple document management [2, 11, 15]. These supports are for general manipulation purposes; explicitly tailored for tabletop users such as the picking, moving, shoving, and rotating of active widgets on the tabletop, and the defining of users' public and private work areas.

In 1993, Wellner [16] presented and discussed the technique of merging physical papers with digital documents and other onscreen widgets onto a common desktop surface and make them all manoeuvrable directly by hand. This technology of "computerised reality" (or sometimes known as "mixed-reality") has since then caught on, progressing from a simple, single-user-setup that is attached to a single projector, a camera, and a computer, to that of a more complex multi-user-arrangement that is capable of facilitating both collaborative and cooperative activities while aiding users in their quest to solve more important tasks. Most of these interactive tabletops are designed to conform with the computer supported cooperative work (CSCW) standards, which have the main goal of using technologies of computer net-

working and associated hardware, software, services, and techniques to support (and to a certain extent, understand) the way people work in groups [1, 5, 7, 12, 14].

Research for the tabletop displays and the related designs of CSCW environments to encompass multi-user interactions made way for further developments in both hardware innovations and software interfacing models. We see new and hi-tech tabletop equipments presented to the scientific community almost every year since 1994. Many are coupled with properly positioned sensing devices such as cameras and motion detectors, while few others come with human-contact-type sensors [2, 4, 15]. The tabletop displays themselves have recently evolved with built-in sensor architectures smart enough to recognise multiple hand positions and allow synchronous inputs [2, 11, 17]. These are held together by control and cognition algorithms that provide the backbone support for all activities on the tabletop systems. Such assistance form the software programs bridges the interfacing problems between the tabletops' four-corners and the simultaneous interface for multiple users [5, 6, 12].

However, we often note that no significant efforts can be found in the literature to determine which of the user around the table is currently and actively interfacing the tabletop; suffice to know that "someone" is touching and manipulating the onscreen widgets. For even with the help of cameras, techniques to differentiate the users are complex, and subsequently, the cost of such complexities may affect the overall multi-user tabletop-interaction profoundly in terms of reaction times (latency) [18, 19].

Our solution in attempting to correctly recognise which of the users seated around the tabletop display (and as a consequence get the interface program to work or react in favour of the identified user) lie in monitoring closely the input command gestures received by the tabletop operating environment. This is as apposed to placing biometric sensors on the users, or investing on special tabletop sensors that send signals through users upon their touch.

# 2 The *Dis*oriented Pen-Gestures Approach

In the example from Figure 1(a), the "arrow" drawn on the tabletop is an "up-arrow" to a user seated in the South corner, but a "right-arrow" to a user in the East corner. Depending on who drew it, our method will tell apart, whether the gesture is a "*STH*_UP_ARR" or an "*EST*_RT_ARR" accordingly. Because of this attached corner-identity in the recognised gesture-command, the foreground application can simply make use of this information to (perhaps) continue processing and managing the tabletop UI for better communication controls in favour of the user at the identified corner.

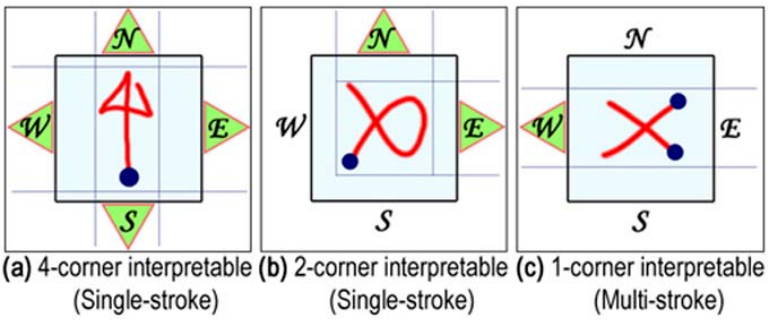


*Figure 1. Similar in appearance but differ in by-corner-interpretation of single and multi-strokes.*

Our approach to identify users seated around the tabletop display without relying on external sensing devices is to bank-in purposefully on the characteristics of the gestures received.

That is, we emphasise our observation on the way users *gesture* commands to the system with their digital pens (or their fingers). We discuss our method in this paper to show that we can relate these *gesturing-techniques* to any of the four corners of the table by virtue of the "orientation properties" contained in every gesture made on the tabletop environment. All pen-gestures are made up of various identifiable "trace-features" in spatial and temporal domains [9]. This set of trace-features is used, in conjunction with a primed set of related trained-weights in classical linear gesture-classification algorithms, to recognise input traces as possible gestures.

By carefully selecting a set of *key* features from the spatial domain and combining them with another set of *key* features from the temporal domain, our method draws strong conclusions of how *dis*-oriented a particular trace is. These are mainly the variance in angular and velocity-type features that we use – within the "dis-orientations" – to relate a trace to one of the corners of the table. This is the corollary of the original linear gesture-classifier theorem [13]; that a gesture is recognisable through features that describe the nature of its trace outline by means of multimodal distribution, meaningful/reusable semantics, relative dependency, and real value variables. Consequently, the classification of the correct gesture type, as well as the identification of the particular table corner it came from, is derived simultaneously using the same algorithm.

# 3 Pen-Gestures as Interaction Protocols

A standard way of communicating to the control system hosting an Ink Environment [10] is to involve the pen-gesture technology as *interaction protocols*. Where in the absence of a keyboard and a mouse, the digital pen is used directly on the digital screen surface to mark the start of the communication, select onscreen items as parameters, and conclude with a final mark of completion. Most of the time, we tend combine all of the three steps above into a single- or multi-stroke gesture-command to achieve the same effect.

The terminology of "gesture-command" insists that the pen-gestures applied on the environment are to be interpreted as system commands, which are to be acted upon immediately. A few or all of the trace-features contained in the gesture can be reused to affect this action visually through appropriate animations on the selected screen item for the users' benefit. The gestures therefore, should be identifiable from anywhere within the predefined environment enclosure.

Similarly, we describe our tabletop environment as an extension of the Ink Environment. And as such, when a gesture-command is drawn on an *active* onscreen widget, its reaction to the gesture is always somewhat related. An active onscreen widget here is one that we can animate expressively on the tabletop environment, as apposed to *passive* widgets which remain anchored to and unmovable at stipulated screen positions.

For instance, we can *gesture* a unidirectional arrow directly on an active document on the tabletop. The entire protocol is realised upon recognition of the gesture-command as, say, "WST_LT_ARR", and the active document as the object parameter to the protocol. In addition, the tabletop environment also knows that the gesture comes from the user seated on its West corner. To complete the visual effect, the document is animated on the tabletop following the properties of the "arrow" gesture – drawing its speed, acceleration, trajectory, and direction from the trace-features contained in the gesture-command. Note that this is not the same as tapping the document, then meticulously dragging it across the table, and letting it go only when we are satisfied with its new position. In effect, there are other means to do this and are explained by several authors [11, 14, 15].

# 4 Significance of Trace-Features

Rubine [13] identified 13 unique features used in determining the gesture-type of an input trace for his gesture trainer and classifier algorithms. These trace-features purportedly describe the measurements of the traces' various lengths, bounding box properties, angles, durations, and speeds. They were selected to accommodate both small and large changes to the feature values that in turn will influence the stability of the trained-weights in the linear discriminator.

## 4.1 The Linear Discriminator

The identification of a gesture-type depends heavily on the goodness of variance on the values of the trained-weights $w_{ci}$ in association to a related trace-feature $f_i$. In a linear gesture-classifier, the set of trace-features $f_1, ..., f_F$ extracted from an input trace $g$ is applied to the linear evaluation function over the values of the trained-weights. The classification of the gesture $G$ is the maximum $v_c$ of all the known gesture values.

$$v_c = w_{co} + \sum_{i=1}^{F} w_{ci} . f_i \text{ ; for } 0 \le c \le C$$

Here, $C$ is the total number of known gesture-types and $F$ is the total number of features extracted from $g$. The training problem then boils down to determining the weights $w_{ci}$ from a sample set of traces that are meant to be interpreted as gestures.

## 4.2 Center of Gravity and Angular Velocity

We implemented Rubine's training algorithm and retained all 13 features in our method. We also added seven more features that measure a trace based upon its angles and speeds, with respect to the trace's center of gravity (COG). By definition and by our implementation, the COG is considered as both spatial and temporal. It tells us a point within the trace's bounding box of where it is most heavy and most balanced when viewed on the 2D plane. Temporally, the concentration of points (which led to the COG itself) is in fact the result of the input sampling rate of the digital screen device (when used as a tabletop), coupled to the users' way of manoeuvring the pen while gesturing; slowing down at some locations (which will pick up more sampled-points per square inch) and hurrying up at others.

Angular and velocity-type features, when observed from the trace's COG, are our *dis*-oriented features. We found them to have the most significant variance for their associated trained-weights $w_{ci}$, when it comes to deciding which corner of the table a gesture was conceived. Our study of analysing how independent an individual feature is with respect to other features (involving the original 13 features) in returning the correct classification of a gesture-type confirms this. Our observations point out that angular (spatial) and velocity-type (temporal) features can supersede all others, and are thus *key* features in their role of identifying gestures from tabletop corners.

## 4.3 The *Dis*oriented Features

Our seven additional features are derived from an input trace as illustrated in Figure 2. The center of gravity $c_0 = (x_{c0}, y_{c0})$ is a point on the 2D plane that is the median of all $\{x_0, \ldots, x_n\}$ and $\{y_0, \ldots, y_n\}$. We list our feature equations as follows, enumerating them after the first 13 originals from Rubine's:

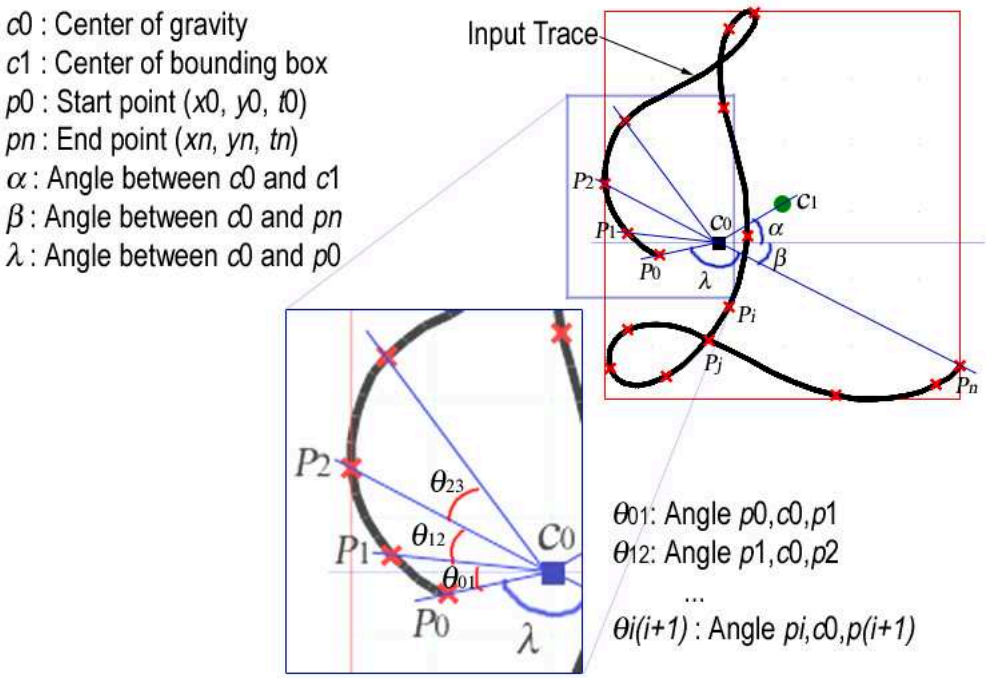


*Figure 2. Deriving angles with respect to the COG and adding time components*

$$f_{14} = \cos\alpha = \frac{(x_{c1} - x_{c0})}{\sqrt{(x_{c1} - x_{c0})^2 + (y_{c1} - y_{c0})^2}}$$

$$f_{15} = \sin\alpha = \frac{(y_{c1} - y_{c0})}{\sqrt{(x_{c1} - x_{c0})^2 + (y_{c1} - y_{c0})^2}}$$

$$f_{16} = \cos\beta = \frac{(x_{pn} - x_{c0})}{\sqrt{(x_{pn} - x_{c0})^2 + (y_{pn} - y_{c0})^2}}$$

$$f_{17} = \sin\beta = \frac{(y_{pn} - y_{c0})}{\sqrt{(x_{pn} - x_{c0})^2 + (y_{pn} - y_{c0})^2}}$$

$$f_{18} = \cos\lambda = \frac{(x_{p0} - x_{c0})}{\sqrt{(x_{p0} - x_{c0})^2 + (y_{p0} - y_{c0})^2}}$$

$$f_{19} = \sin\lambda = \frac{(y_{p0} - y_{c0})}{\sqrt{(x_{p0} - x_{c0})^2 + (y_{p0} - y_{c0})^2}}$$

Let $\Delta x_i = x_{c0} - x_i$, and $\Delta x_j = x_{c0} - x_j$.

Let $\Delta y_i = y_{c0} - y_i$, and $\Delta y_j = y_{c0} - y_j$.

Let $\Delta t_i = t_{i+1} - t_i$.

For $0 \le i \le n$ and $0 \le j \le n$. Then

$$\cos\theta_{ij} = \frac{\Delta x_i \Delta x_j - \Delta y_i \Delta y_j}{\sqrt{(\Delta x_i^{\,2} + \Delta y_i^{\,2})(\Delta x_j^{\,2} + \Delta y_j^{\,2})}}$$

$$f_{20} = \max_{i=0}^{n-1} \frac{\cos\theta_{i(i+1)}}{\Delta t_i^{\,2}}$$

Features $f_{14}$ to $f_{19}$ are a series of related pairs, taking into consideration the cosine and sine of angles between $c_0$ (COG) and $c_1$ (center of bounding box), $c_0$ and $p_n$ (last point on the trace), and $c_0$ and $p_0$ (first point on the trace) respectively. As was pointed out by Rubine, we favoured using the cosine and sine of the angles rather than using the angles themselves to avoid discontinuity as it passes through $2\pi$ and 0. These preliminary features detect the levels of *dis*-orientation a trace has that are centred on its heaviest point, where many of the sampled points within the trace are normally concentrated. These concentrations usually happen at bends and corners around the trace, and whenever users slow down in their gesturing. Our primary observations indicate that these features have a tendency to show a significant variance depending on the corner from which the gesture was drawn, even when the outline of the gesture looks very much alike from all four corners of the table. We need to detect enough of these variances to ensure our derivation of the table-corner, from just an arbitrary input trace, has a high probability of being correct.

# 5 Tabletop Gesture Groups

## 5.1 Family Group

We define a *family* of a gesture-type as a set of gestures whose trace-outlines look very much alike when viewed from more than one corner of the table, but may have different interpretations (of their names and system commands) with respect to the corner they are associated with. Figures 1(a) and 1(b) illustrate this idea. For the same gesture outline in Figure 1(a), we interpret it as STH_UP_ARR, EST_RT_ARR, NTH_DN_ARR, and WST_LT_ARR respectively, going anti-clockwise around the table starting from the South corner. Similarly in Figure 1(b), we interpret the gesture as NTH_ALPHA and EST_TWRL_DN from the North and East corners respectively.

Note that from the way an ‘alpha’-gesture is drawn, we do not include interpretations for the West and South corners as they would prove cognitively unpopular [8, 17]. This is the same reason why Figure 1(c) should only be interpreted from the West corner.

Figure 3 gives a few more examples of the family of gestures that we studied for this paper. A 90-degree rotation of one family gesture results in another family gesture of the same set. A complete *set* of family gesture contains all family gestures of the same type for all four corner-orientations.

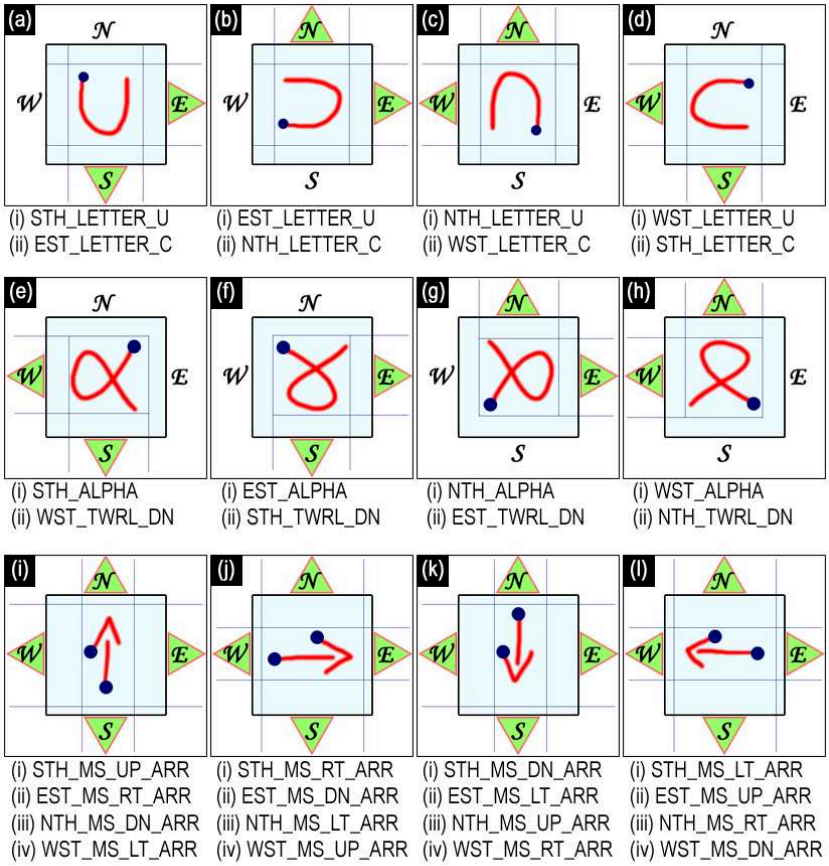


*Figure 3 . Three sets of complete single- and multi-stroke families of gestures: Set 1 – 3(a), (b), (c), (d); Set 2 – 3(e), (f), (g), (h); Set 3 – 3(i), (j), (k), (l)*

## 5.2 Corner Group

A *corner* group refers to a set of gesture-types that has the same command interpretation from more than one table corner. Usually, the outlines of these gestures are not the same when viewed from a single corner of the table. A corner group is a consequence of picking one gesture from each family belonging to a complete family-set, and ensuring that the orientation information in the gesture name is exclusive.

Figure 4 shows an example of an UP_ARROW corner group, which was derived from the complete set of family gestures in Figure 1(a).

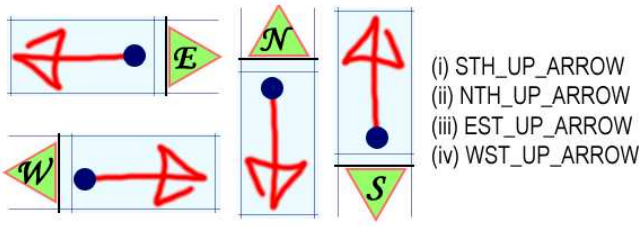


*Figure 4 . The UP_ARROW corner group.*

# 6 Variance Comparison in Trained-Weights $w_{ci}$

We mentioned earlier that we used Rubine's classical training and recognition algorithms in this study. The only differences here are that we are working with a tabletop environment, and with seven additional feature definitions.

The trained-weights $w_{ci}$ of all the known gesture-types are amassed together and then observed categorically by their family and corner groups. Particularly, we are interested in how much each trained-weight of a single gesture-type varies with respect to the same weight in all other gesture-types. The higher this variance, the more significant the trained-weight is, as it will have a bigger influence in the decision making equation of the linear discriminator.

## 6.1 Training by Table Corners for Family

In order that we have a complete set of family gestures, as illustrated in Figure 3, we need to ensure that we create training examples from the correct corners. As a matter of fact, we need to train at least 15 examples from one corner of the table for one particular orientation of a single gesture-type, and then repeat this operation on all four corners of the table to complete the family set. The gestures trained for each corner must coincidentally have the corner information imbued at the start in its name (i.e. NTH_, STH_, EST_, WST_).

We note here that the total number of gesture-types $C$ for the tabletop is at least four times as many as the number of gesture-types for conventional digital screens. Four being the number of corners that our tabletop equipment has.

## 6.2 Observation of the Trained-Weights $w_{ci}$

The results we obtained when observing the variances of the trained weights $w_{ci}$ in association to their trace-features $f_i$ are presented as follows.

Based on a given library of gesture-types $C$, we extract the values of each trained-weight $w_{ci}$ for all $c \in C$ and $1 \leq i \leq 20$, and categorically study the variance of their standard deviations. Single-stroke gestures are viewed separately from the multi-stroke gestures, as the latter is made up of a few more identical trace-features $f_i$ for each stroke-component that make up the full multi-stroke gesture. We do not, however, discuss in this paper our methodologies of multi-stroke gesture recognition that utilises and combines the fundamental trace-features on each gesture-component used in the current single-stroke recognition technique. The reader, however, is invited to read our technical report if he/she wishes further information on this matter []. In this paper though, we believe that it is sufficient to state that multi-stroke gestures require several more trace-features than single-stroke gestures for efficient recognition of the gesture-types as well as identifying the corners they come from. This is evident in the graphs shown in Figures 7 and 8, compared to the graphs shown in Figures 5 and 6.

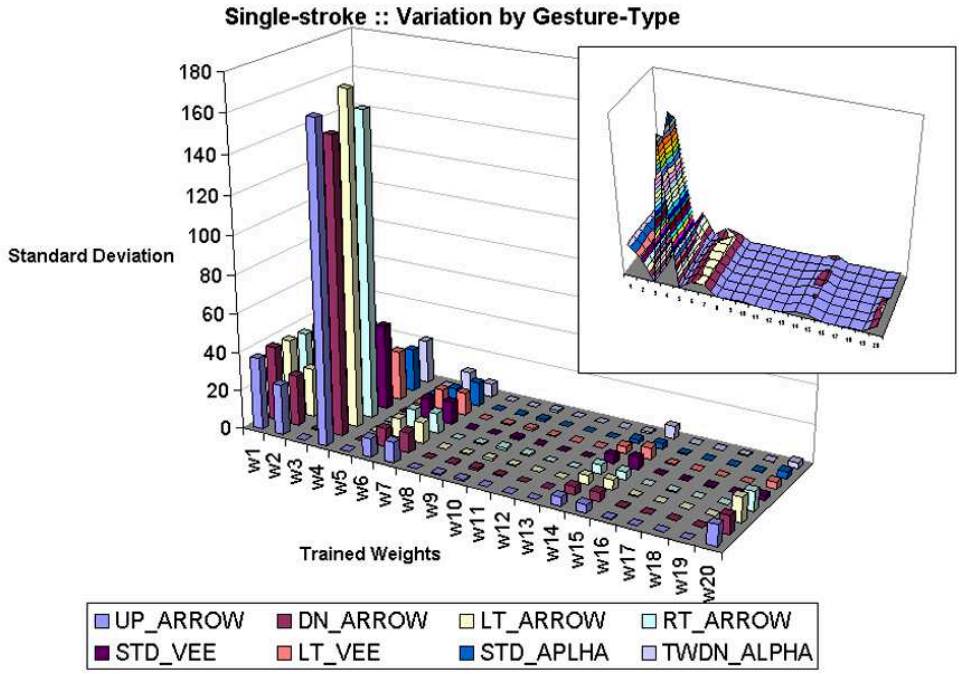


*Figure 5 .Variations of trained-weights $w_i$ for single-stroke gestures arranged by family gesture-types.*

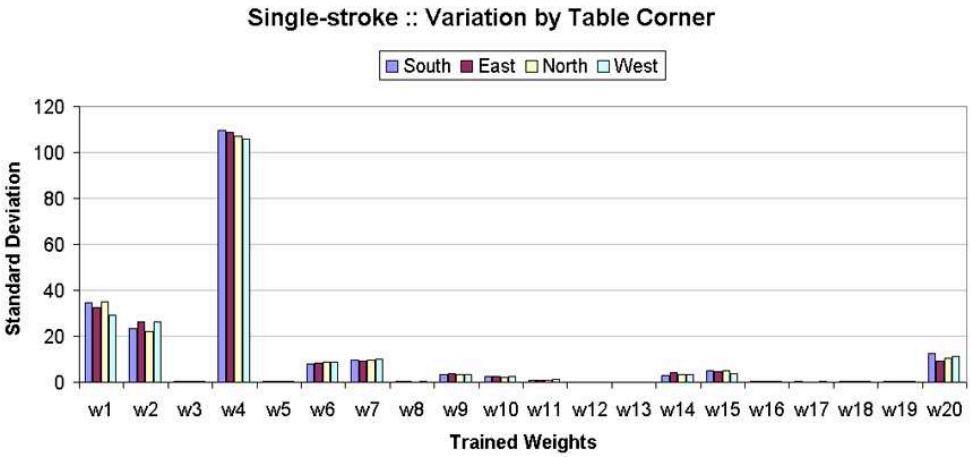


*Figure 6. Variations of trained-weights $w_i$ for single-stroke gestures arranged by corner-group gesture-types.*

We find that the trained-weights $w_{ci}$ that vary the most have associated trace-features $f_i$ that are composed of angular and velocity-type components. This is obvious for the single-stroke gestures as depicted in the peaks of the graphs in Figures 5 and 6; the former arranged by family-group gesture-types, and the latter by the corner-groups.

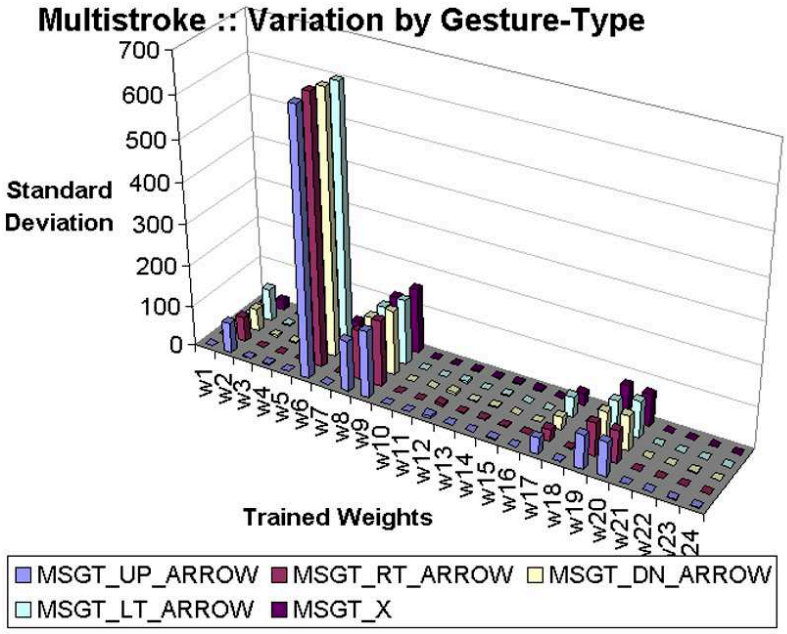


*Figure 7. Variations of trained-weights $w_i$ for multi-stroke gestures arranged by family gesture-types.*

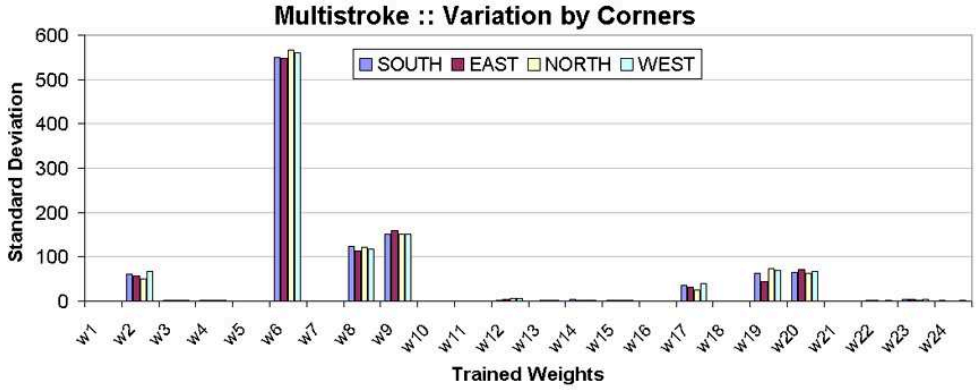


*Figure 8. Variations of trained-weights $w_i$ for multi-stroke gestures arranged by corner-group gesture-types.*

# 7 Implementation: Tabletop Monopoly

We programmed our own prototype of the Monopoly board game to test out the stability of our hypothesis. Our manual and partially-automatic Monopoly game was made to run on the most basic of tabletop environments, and involves a good mixture of active (e.g. dice, cards, money, player pieces) and passive (e.g. street boxes) onscreen widgets.

## 7.1 Handling 'Out-of-Turn' Actions

Monopoly is a turn-based game, and each new turn is preceded by the rolling of the dice. Depending on where on the street boxes the player lands, he/she can decide to make a purchase, auction, or pay rent to the owner of the street box property. Here, we are most inter-

ested in the last bit – the paying of rent. From the rules of the Monopoly game, the owner of the particular street box must demand a payment from his/her opponent who has just landed on the owner's property. This action must be done 'out-of-turn', that is, the owner of the stipulated property needs to ask for rent payment before the next rolling of the dice, or the owner forfeits the amount due. Around the tabletop, the owner of the property may not necessarily sit in the next turn of the current player (Monopoly changes players' turns in clockwise round). And because our Monopoly is programmed for *manual* play, the computer does not know which players own what properties.

This is where the advantage of our method comes in. We made use of up to eight *easy-to-draw* family of gestures in the game, and instructed the players to use them as communication protocols to affect their commands to the game controller. That is for example, a player performs a gesture to roll the dice, then gestures on his/her piece to move it across the board in the number of steps indicated by the dice, and gesture once more to indicate his/her desire to purchase or auction the street box property that he/she has just landed on.

In the same manner, the owner of the street box property *gestures* anywhere on the tabletop to demand payment from his/her opponent. The computer deciphers this gesture and translates it to obtain the 'demand-rent' protocol-command as well as derive which corner of the table the gesture came from. Assuming that the owner of the property does not change places throughout the duration of the game, the computer is able to properly service the owner as per required. Figure 9 details this scenario graphically.

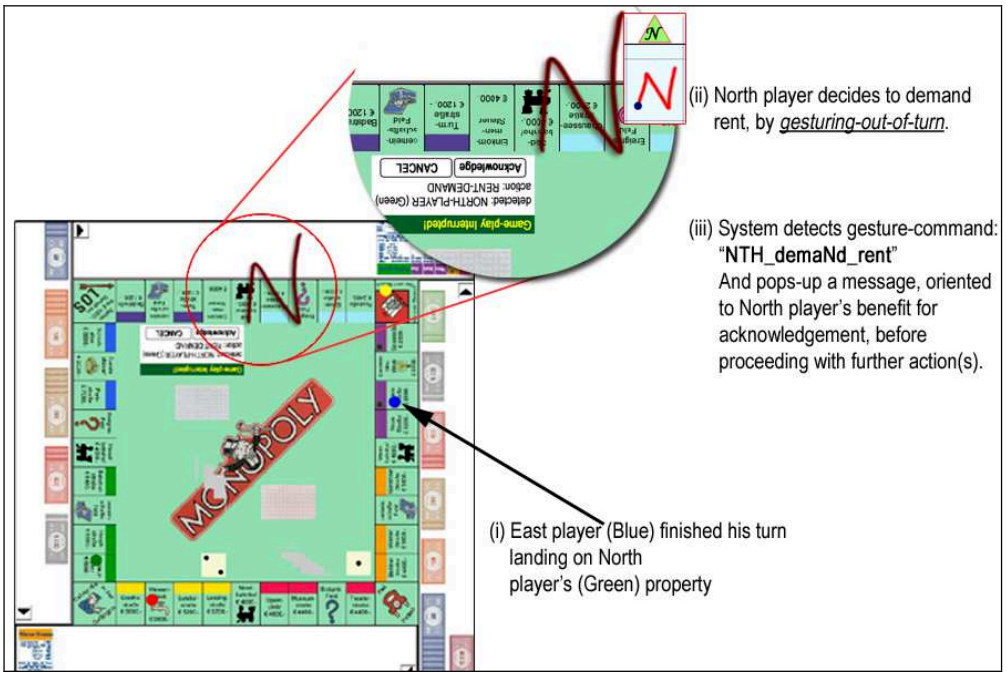


*Figure 9. North player gesturing 'out-of-turn', after East player landed on North player's property.*

Other 'out-of-turn' actions in the Monopoly game that take advantage of the *dis*-oriented gestures include buying houses and hotels, redeeming mortgaged properties, and selling/buying owned properties between any two players.

## 7.2 Correctness of Identification

Rubine, in his original paper, suggested several ways of rejecting false positives that trigger a misclassification of an ambiguous gesture-type [13, 3]. This happens when there is a near tie

for the maximum per-gesture-type evaluation in the linear discriminator. And as the corner information of the gesture-type is tied strongly to the features $f_1$ to $f_{20}$, our implementation of Rubine's method yielded the same results of at least 98% recognition rate of the correct gesture-type as well as the table corner. This is the corollary of the gesture-classification algorithm working.

Now, with respect to our Monopoly table-game, we collected results from a log file that we save at each run of the game. We ran the game 20 times; 14 involving four players, four involving three players, and two involving two players. The log file kept count of the number of times a gesture was made while a player was *in-turn*, and it also noted the number of times *out-of-turn* gestures were made. The results are shown in Table 1.

A typical complete game of Monopoly averages 27% out-of-turn gestures, and out of that, more than 98% were correctly classified (both gesture-type and table-corner).

| Gesture Action | Frequency | Correct | Mis-classified |
|---|---|---|---|
| In-turn | 73.00% | 99.21% | 0.79% |
| Out-of-turn | 27.00% | 98.87% | 1.13% |

*Table 1.Breakdown of gesture-actions for the Monopoly table-game.*

# 8 Generality of Methods

Our method of determining the table-corner from which a gesture comes from is derived directly from our on-going work with the pen-gesture technology that involves conventional digital screen environments. These screens are usually upright, sometimes wall-mounted, but always having only one orientation and only cater for one *anonymous* pen-input at a time. The tabletop display that we are working with makes use of a conventional digital screen, laid out flat on a table, and can only receive asynchronous pen-inputs.

The key point brought forward in this work is that users of our tabletop environment will be required to *gesture* to the system with their digital pens, in order to register their intention and start the protocol for communication. As long as users continue to use gestures as their means of controlling their communication and interfacing on the tabletop, our method should still remain valid. We just need to ensure that proper training of the gesture-types is carried out promptly irregardless of the medium used for gesturing (digital pen, fingers, etc.).

Also, if new tabletop hardwares that can handle multiple inputs simultaneously, can subtly separate all incoming information and pipe complete gestures for serial processing, then the tabletop software controlling the environment can be tweaked so that synchronous inputs can be processed the same way as the current asynchronous ones. Since the raw-gesture information are then intact, we will be able to determine which corners all incoming gestures come from.

# 9 Conclusion

We have shown that it is possible to derive the corner of the tabletop of which a valid pen-gesture comes from and, as a consequence, identify the user seated at that corner. All these

without having to use external tracking devices or attaching biometric sensors on the users. We discussed the strong correlations that the angular and velocity-type features have, contained inside gestures, to the corner of the tabletop of which they were trained and conceived. These features that tell the correct corner need to be related to the COG of the stipulated gesture.

**Acknowledgement**

This research is funded by the Deutschen Forschungsgemeinschaft (DFG) as part of the research project „Algorithmen und Datenstrukturen für ausgewählte diskrete Probleme (DFG-Projekt Ot64/8-3)“.

**References**

[1] M. Billinghurst and H. Kato. Collaborative mixed reality. In *International Symposium on Mixed Reality*, pp 261-284, 1999.

[2] P. Dietz and D. Leigh. Diamond touch: a multi-user touch technology. In *Proceedings of UIST 01*, pp 219-226, 2001, ACM Press.

[3] C. Frankish, R. Hull, and P. Morgan. Recognition accuracy and user acceptance of pen interfaces. In *Proceedings of CHI 95*, 1995.

[4] M. Kobayashi and H. Koike. Enhanced desk: integrating paper documents and digital documents. In *Asia Pacific HCI*, pp 57-62, 1998, ACM Press.

[5] R. Kruger, M. S. T. Carpendale, S. D. Scott, and S. Greenberg. How people use orientation on tables: comprehension, coordination and communication. In *Proceedings of the ACM Group 2003 Conference*, pp 369-378, 2003.

[6] W. Kruger and B. Frohlich. The responsive workbench. In *IEEE Computer Graphics and Applications*, 14(3):12-15, 1994.

[7] I. McCowan, D. G.-Perez, S. Bengio, D. Moore, and H. Boulard. Towards computer understanding of human interactions. In *Proceedings of European Symposium on Ambient Intelligence*, pp 56-75, LNCS.

[8] K. A. Mohamed and Th. Ottmann. Pen-based digital screen interaction. In Claude Ghaoui (ed), *Encyclopedia of HCI*, 2005, Bershire Publishing Grp.

[9] K. A. Mohamed and Th. Ottmann. Increasing the accuracy of anticipation with lead-lag timing analysis of digital freehand writings for the perceptual environment. In T.E. Simos (ed), *Proceedings of the ICCMSE*, pp 387-390, 2004, VSP/Brill Science Citation Index.

[10] K. A. Mohamed and Th. Ottmann. Fast interpretations of pen gestures with competent agents. In *Proceedings of the IEEE Second International Conference of CIRAS*, pp PS09-04, 2003.

[11] J. Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of CHI 01*, pp 113-120, 2001.

[12] Y. Rogers, W. Hazlewood, E. Blevis, and Y.K. Lim. Finger talk: collaborative decisions-making using talk and fingertip interaction around the tabletop display. In *Proceedings of CHI 04 – Extended Abstracts*, pp 1271-1274, 2004.

[13] D. Rubine. Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, pp 329-337, 1991, ACM Press.

[14] K. Ryall, C. Forlines, C. Shen, and M. R. Morris. Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. In *Proceedings of CSCW 04*, 2004, ACM Press.

[15] B. Ullmer, H. Ishi, and D. Glas. The metaDESK: models and prototypes for tangible user interfaces. In *Proceedings of UIST 97*, pp 223-232, 1997.

[16] P. Wellner. Interacting with paper on the digital desk. In *Communications of the ACM*, 36(7):86-96, 1993.

[17] M. Wu and R. Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of UIST 03*, pp 193-202, 2003, ACM Press.

[18] C. von Hardenburg and F. Berard. Bare-hand human computer interaction. In *Proceedings of the 2001 Workshop on Perceptual User Interface*, pp 1-8, 2001.

[19] S. M. Dominguez, T. Keaton, and A. H. Sayed. Robust finger tracking for wearable computer interfacing. In *Proceedings of the 2001 Workshop on Perceptual User Interface*, pp 1-5, 2001.