SmartCanvas: A Gesture-Driven Intelligent Drawing Desk System

Zhenyao Mo
CGIT Lab, USC
3740 McClintock Ave, EEB131
Los Angeles, CA 90089 USA
+1 213 740 4250
zmo@graphics.usc.edu

J. P. Lewis
CGIT Lab, USC
3740 McClintock Ave, EEB131
Los Angeles, CA 90089 USA
+1 213 740 9619
zilla@computer.org

Ulrich Neumann
CGIT Lab, USC
3740 McClintock Ave, EEB131
Los Angeles, CA 90089 USA
+1 213 740 0877
uneumann@usc.edu

ABSTRACT

This paper describes SmartCanvas, an intelligent desk system that allows a user to perform freehand drawing on a desk or similar surface with gestures. Our system requires one camera and no touch sensors. The key underlying technique is a vision-based method that distinguishes drawing gestures and transitional gestures in real time, avoiding the need for "artificial" gestures to mark the beginning and end of a drawing stroke. The method achieves an average classification accuracy of 92.17%. Pie-shaped menus and a "rotate-to-and-select" approach eliminate the need for a fixed menu display, resulting in an "invisible" interface.

Categories and Subject Descriptors

H.5.2 [**Information Systems Applications**]: User Interfaces—*Input devices and strategies, Interaction styles*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation, Pattern analysis*

General Terms

Algorithms, Human Factors

Keywords

Intelligent user interface, gesture recognition, Support Vector Machine

1. INTRODUCTION

Many vision-based desk systems [1, 3, 5, 13] allow users to create and manipulate graphical objects (circle, rectangle, etc.) with gestures. However, to allow freehand drawing without touch sensors, a key problem is that a mechanism is required to discriminate drawing strokes and transitional strokes, i.e., whether a user is drawing on the desk or is just relocating the fingertip to the starting position of the next stroke.

IUI'05, January 9–12, 2005, San Diego, California, USA.

One simple solution is to mount a camera (as camera 1 in Fig 1) to monitor whether a finger touches the surface or not. This approach requires two cameras, however, and the placement of camera 1 is crucial to the system operation.

In another approach, as shown in a demo video of a drawing board system [1], a user's thumb is extended for switching from transition mode to draw mode (see Fig 2). With such a mechanism, a user's drawing speed is limited due to the frequent mode switch.

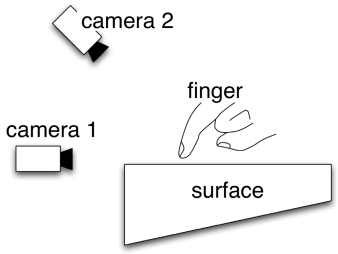

Figure 1: a vision system with two cameras, one to detect whether a finger touches the surface, the other to track the finger's trajectory.

Figure 2: left, transition mode; right, extending the thumb, draw mode.

A third mechanism requires a user's fingertip to stay still for a few seconds both in the beginning and at the end of a drawing gesture so that the system can recognize the stroke [3].

The latter two mechanisms require that extra gestures be inserted into drawing sequence, and the user must change his or her drawing behaviors to fit the system.

In one experiment, we showed two videos of gesture sequences to 5 persons, one delineating the character "Z", and the other delineating "=". Although the fingertip trajectories of the two sequences are almost the same (see Fig 3),

all 5 persons were able to correctly correlate sequences to characters.

An analysis of the two sequences shows that differences are visible in temporal dimensions. As in Fig 3 lower left, the velocity patterns for three drawing strokes are similar whereas in Fig 3 lower right, a transitional stroke (the middle one) is different from the two drawing strokes.

The observation leads to a relatively simple and efficient (real-time) method for classifying transitional gestures and drawing strokes, based on combining Support Vector Machines and a Finite State Machine.

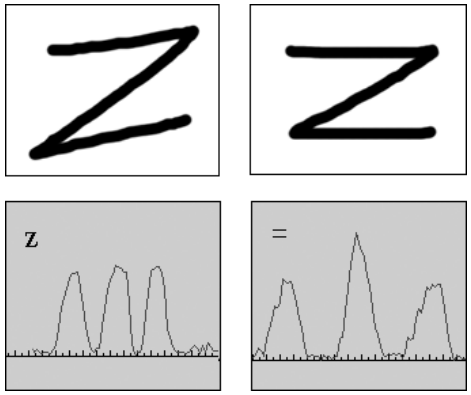

Figure 3: upper, spatial fingertip trajectories for drawing the characters "Z" and "="; lower, velocities along time dimension for drawing "Z" and "=".

This gesture classification method forms the foundation of a vision-based augmented desk system called SmartCanvas, which allows a user to perform freehand drawing using gestures.

The remainder of the paper is organized as follows: after reviewing related work, section 3 describes our method to recognize and classify gestures; section 4 considers menu design of the SmartCanvas system; section 5 discusses future research and concludes the paper.

2. RELATED WORK

Several augmented digital desk systems have been proposed that allow users to draw on a desk. As in [5] and [13], drawing is performed on a physical basis using pen and paper. With a camera tracking the desk, these systems provide a set of functions that greatly enhance the user's ability to interact with the content on paper, thus improving a user's efficiency and productivity.

In [3] a gesture-driven desk system is described. It is reported that to draw simple images using two hands, users achieve better performance than using a traditional mouse and keyboard based system (Adobe Illustrator). Users are able to draw in two ways: either using predefined shapes (circle, rectangle, etc.), or using freehand strokes.

An algorithm using Hidden Markov Models [8] is proposed for automatic handwriting gesture recognition. A Uni-stroke alphabet set is used so that no transitional gestures exist within a character's stroke sequence. Recognition rates between 88% and 100% are achieved.

Many gesture tracking and recognition algorithms have been proposed, from simple 2-D algorithms to sophisticated 3-D view and pose recovery. The gesture tracking algorithm used in our system is similar to the ones proposed in [4], [6], and [11].

3. HANDWRITING GESTURE RECOGNITION

In this section, a simple but effective algorithm for 2-D hand and fingertip tracking is first explained. Then we introduce our method to classify transitional strokes and drawing strokes using Support Vector and Finite State Machines.

3.1 Tracking Hands and Fingertips

Real time hand recognition and fingertip tracking is achieved by a vision-based method, which is summarized as follows.

It is assumed that hands are the only naked skin regions in the view of a camera (we require that users wear long-sleeve shirts, and the camera is adjusted so that a user's face is not in the view).

Hands are segmented from the remainder of a scene using a method proposed in [7]. In an environment of frequently changing luminance (as in our lab), instead of RGB color space, the I1I2I3 color space is used [9] and histograms of skin and non-skin color distribution are built upon I2I3 plane.

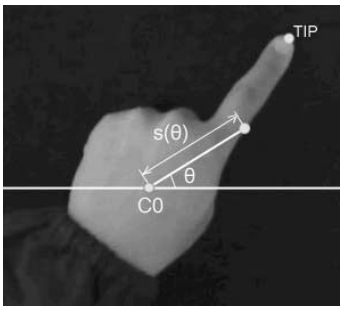

Figure 4: function $S(\theta)$ is defined as the distance of the farthest skin pixel from palm center C_0 at angle θ, $\theta \in [0, 2\pi)$.

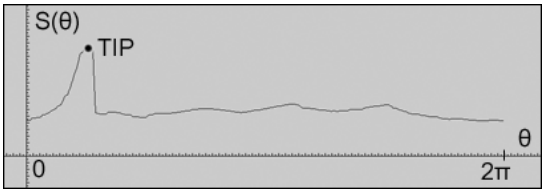

Figure 5: fingertips are located as local maxima of $S(\theta)$.

The center C_0 of a hand is defined as the point on the hand that maximizes its distance to the closest hand region boundary. C_0 is located by applying a morphological erosion operation.

For each hand pose, a function $S(\theta)$ is constructed as the distance of the farthest skin pixel from C_0 at angle θ (see Fig 4). Fingertips are located as the local maxima of $S(\theta)$ (see Fig 5). To reduce skin region segmentation noise, a median filter is applied to $S(\theta)$ before locating fingertips.

A Kalman filter is applied to track a 2-D fingertip trajectory. An observation $\vec{p}$ and a system state $\vec{s}$ are defined as:

$$\vec{p} = (x, y)^T$$

$$\vec{s} = (x, y, v_x, v_y)^T$$

where (x, y) is the position of a fingertip, and (v_x, v_y) is the velocity.

The system is described as:

$$\vec{s}_{t+1} = \mathbf{F} \cdot \vec{s}_t + \mathbf{G} \cdot \vec{w}_t$$

$$\vec{p}_t = \mathbf{H} \cdot \vec{s}_t + \vec{v}_t$$

where $\vec{p}_t$ and $\vec{s}_t$ refer to the observation and the system state at frame t, $\mathbf{F}$ is the state transition matrix, $\mathbf{G}$ is the driven matrix, $\mathbf{H}$ is the observation matrix, $\vec{w}_t$ is the system noise, and $\vec{v}_t$ is the observation noise. Here the velocity is assumed constant, which is compensated for by adding system noise $\vec{w}_t$. For detailed formulation of Kalman filter, see [11] and [12].

3.2 Stroke Classification Using Support Vector Machine

This subsection explains our method of using a Support Vector Machine (SVM) to classify strokes into two categories: transitional strokes ($\hat{T}$) and drawing strokes ($\hat{D}$).

A stroke is defined as a segment of fingertip motion that is consistent both spatially and temporally. The end of a stroke is identified by:

1. a sharp change of orientation in fingertip trajectory; or
2. the fingertip stays still for a few frames (not a few seconds).

Support Vector Machine is well known for its performance on object classification by maximizing margins. The difference of a $\hat{T}$ stroke and a $\hat{D}$ stroke is in temporal dimensions. Thus, SVM classification is performed on the velocity information of strokes. The velocity $v = \sqrt{v_x^2 + v_y^2}$ in each frame is learned as the output of the fingertip tracking algorithm and Kalman filtering. A stroke (k frames) is associated with a velocity k-size vector $\vec{v}$.

The state vector of a stroke is defined as:

$$\vec{s} = [n(\vec{v}), \alpha(\vec{v}), \beta(\vec{v})]$$

where $n(\vec{v})$ is a size-n vector, which is obtained by resampling $\vec{v}$ along the time dimension and then normalizing it so that $\|n(\vec{v})\|_2 = 1$; $\alpha(\vec{v})$ is defined as the average velocity of the stroke; $\beta(\vec{v})$ is the smooth-ness of the stroke, which is computed as in Appendix A.

The input vector for the SVM is $\mathbf{W} \cdot \vec{s}$, where $\mathbf{W}$ is a weighting matrix:

$$\mathbf{W}_{(n+2)\times(n+2)} = diagonal([w_n, ..., w_n, w_\alpha, w_\beta]).$$

The training set for the SVM consists of 231 strokes (101 transitional strokes and 130 drawing strokes). The 231 strokes are segmented and manually labeled from a gesture sequence drawing 26 English characters twice (by person A).

The trained SVM was applied on 4 gesture sequences performed by person A (sequences are different from the training data), and achieved an average rate of correct classification 80.87%. The same SVM was applied to 2 gesture sequences performed by person B, and achieved 74.36%.

The input video size is 640x480 pixels. The processing (tracking and SVM classification) is done in real time (20 frames per second) on a PC with a Pentium 4 CPU 2.4GHz.

Clearly, the classification with SVM alone is not accurate enough.

Table 1: Total stroke numbers and mistakenly classified stroke numbers in each experiment. Classification is based on SVM.

	A1	A2	A3	A4	B1	B2
Total Stroke	36	35	21	23	35	43
Misclassified	4	6	4	8	7	13

3.3 Improved Classification with Finite State Machine

The perfect pattern for a stroke sequence would be:

$$\hat{T} - \hat{D} - \hat{T} - \hat{D} - \hat{T} - \hat{D} - \hat{T} - \hat{D} - ...$$

However, this does not always occur. For example, one drawing sequence for the character "B" is (see Fig 6):

$$\hat{D} - \hat{T} - \hat{D} - \hat{D}.$$

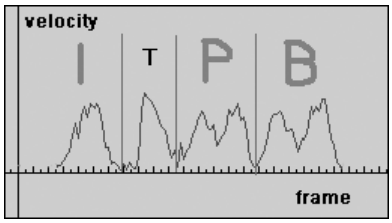

Figure 6: the character "B" consists of 4 strokes, of which the 1^{st}, 3^{rd}, and 4^{th} are drawing strokes, and the 2^{nd} is a transitional stroke.

To model stroke sequences, we defined a Finite State Machine (FSM) of two states ($\hat{D}$ as draw and $\hat{T}$ as transition) (see Fig 7). The probabilities of state transfer from $\hat{D}$ to $\hat{T}$ (72.8%) and from $\hat{T}$ to $\hat{D}$ (100.0%) are much higher than from $\hat{D}$ to $\hat{D}$ (18.2%) and from $\hat{T}$ to $\hat{T}$ (0.0%). The probabilities are approximated by analyzing the training sequence for the SVM.

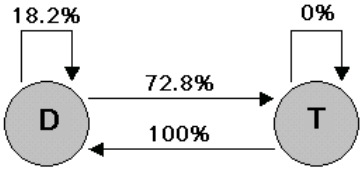

Figure 7: a Finite State Machine with two states. Each state transfer is labeled with a probability.

As seen from Fig 3 lower left and Fig 6, most $\hat{D}$-$\hat{D}$ stroke patterns have the following two properties ("$\hat{D}$-$\hat{D}$" condition):

1. the velocity patterns of the two strokes are similar;

2. few frames exist between two strokes (the fingertip stays still for very short a while).

We can improve the classification accuracy by combining SVM and FSM.

Instead of training one SVM, we train n SVMs using n different sequences (each containing 115 strokes from drawing 26 English characters). A stroke $\hat{S}$ is classified using all n SVMs. $D(\hat{S})$ is the number of SVMs which classifies $\hat{S}$ as a $\hat{D}$ stroke; $T(\hat{S})$ is the number of SVMs which classifies $\hat{S}$ as a $\hat{T}$ stroke. Stroke $\hat{S}$ is classified as $\hat{D}$ with confidence $D(\hat{S})/n$ if $D(\hat{S}) > T(\hat{S})$; $\hat{S}$ is classified as a $\hat{T}$ stroke with confidence $T(\hat{S})/n$ if $T(\hat{S}) > D(\hat{S})$.

The general strategy is: we follow the perfect pattern (a $\hat{T}$ succeeds a $\hat{D}$ and a $\hat{D}$ succeeds a $\hat{T}$ in turn) unless a stroke is classified by SVM with high confidence. The detailed algorithm consists of the following rules:

1. a stroke sequence always starts from a transitional stroke; this requires a user always relocate his or her fingertip before draw a stroke at the beginning of a sequence; thus, we ensure the first stroke always matches the perfect pattern.

2. if a stroke is classified with high confidence by SVMs, then the classification is final;

3. if a stroke is classified with low confidence, and the state transfer is $\hat{D}$ to $\hat{T}$ or $\hat{T}$ to $\hat{D}$, then the classification is final;

4. if a stroke is classified with low confidence, and the state transfer is $\hat{D}$ to $\hat{D}$: if the stroke and the previous stroke satisfy the "$\hat{D}$-$\hat{D}$" condition, then the stroke is classified as a $\hat{D}$, otherwise, the stroke is classified as a $\hat{T}$.

5. if a stroke is classified with low confidence, and the state transfer is $\hat{T}$ to $\hat{T}$, then the stroke is classified as a $\hat{D}$ (we assume no two adjacent $\hat{T}$ strokes, i.e., a user always moves to the starting position of next drawing stroke without "doodling").

Based on the improved algorithm, we perform the 6 experiments again. The same-person classification accuracy is 92.17%. The cross-person classification accuracy is 76.92%. A video showing sequence A1 is available at:

http://graphics.usc.edu/ ˜zmo/SC/demo1.avi.

Table 2: Total stroke numbers and mistakenly classified stroke numbers in each experiment. Classification is based on the improved algorithm.

	A1	A2	A3	A4	B1	B2
Total Stroke	36	35	21	23	35	43
Misclassified	1	1	4	3	4	14

The classification accuracy on sequence B2 is still low. By analyzing the data, we realize that there are several $\hat{T}$ to $\hat{T}$ strokes in sequence B2. After a $\hat{D}$ stroke, person B often moves back to a specific resting position (stroke $\hat{T}_1$) before moving to the starting position of next $\hat{D}$ stroke (stroke $\hat{T}_2$), producing a sequence as "...-$\hat{D}$-$\hat{T}_1$-$\hat{T}_2$-$\hat{D}$-...", which is in contradiction to the assumption of the algorithm.

3.4 Misclassified Strokes Correction

Of all 27 misclassified strokes in Table 2, 16 are $\hat{D}$ strokes misclassified as $\hat{T}$ strokes.

It is observed that when a $\hat{D}$ stroke is misclassified (the stroke is not rendered on the screen as expected), users tend to repeat the $\hat{D}$ stroke immediately, trying to correct the mistake. This pattern is:

$$\hat{S}_1 - \hat{S}_2 - \hat{S}_3,$$

where $\hat{S}_1$ is the misclassified $\hat{D}$ stroke, $\hat{S}_2$ is the $\hat{T}$ stroke that moves back to the staring position of $\hat{S}_1$, and $\hat{S}_3$ is a repeat of $\hat{S}_1$. Thus, the ending point of $\hat{S}_2$ and the starting point of $\hat{S}_1$ should be close, and the trajectories of $\hat{S}_1$ and $\hat{S}_3$ should be similar ("Correction" condition).

When three adjacent strokes $\hat{S}_1$, $\hat{S}_2$, and $\hat{S}_3$ satisfying "Correction" condition are identified, $\hat{S}_1$ and $\hat{S}_3$ are classified as drawing strokes, and $\hat{S}_2$ is classified as a transitional stroke.

Unfortunately, with a misclassified $\hat{T}$ stroke, we have no effective mechanism to correct the error.

4. SMARTCANVAS: THE SYSTEM

SmartCanvas is a gesture-driven system for a virtual drawing desk. A user's index fingertip is used to draw on a regular desk or similar surface. A camera (connected to a PC) is positioned above the desk to track the finger motion and hand gestures. The fingertip trajectory is tracked and segmented into strokes, and strokes are classified as drawing strokes and transitional strokes. Drawing strokes are rendered in real time on the screen (when the end of a $\hat{D}$ stroke is identified), and transitional gestures are ignored.

The system also provides menus that allow a user to select pen color. As reported in [2], pie menus improve over linear menus in both seek time and error rates (mouse as input device). And as reported in [3], in a gesture-driven system, pie menus are also preferable to linear menus. Thus, pie menus are implemented in SmartCanvas.

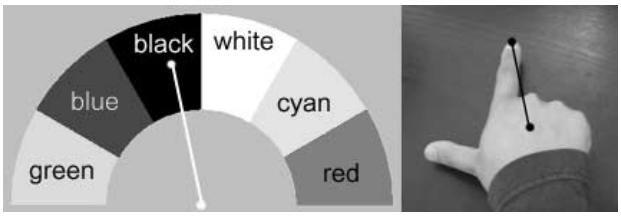

Figure 8: menu items are displayed in a pie shape; the thumb is extended to switch from draw mode to menu mode; the index finger is rotated to locate a menu item.

A user switches from draw mode to menu mode by extending the thumb finger (as in Fig 8). Such mode switch mechanism is reasonable, because menu selection is not frequent.

Because our system does not project menus onto the desk (as in [3]), moving fingers to the location of a menu item for a selection would be inconvenient, because it requires the user to coordinate the fingertip motion on desk with the motion of the pointer on screen.

Thus, instead of "move-to-and-select", we use a "rotate-to-and-select" mechanism. Menu items are displayed in the

upper half of a pie shape (0-π), and each menu item (with total of k submenus) covers an angle range of π/k. A menu item is selected if the index finger's orientation is within the angle range of that menu item for a few seconds. This "rotate-to-and-select" approach takes advantage of a finger's proprioception [10]. A video demonstrating the menu selection along with a drawing sequence is available at:

http://graphics.usc.edu/ ~zmo/SC/demo2.avi.

No eraser function is provided. However, "white" is provided as one of the available pen colors, which is also the background color. Thus, selecting a "white" pen will produce the same effect as an eraser, which allows a user to make corrections.

5. CONCLUSION

Any vision-based drawing system will require an unobtrusive means of distinguishing transitional strokes from drawing strokes. In this paper, we show that transitional strokes can be distinguished from drawing strokes in real time using a combination of Support Vector Machines and a Finite State Machine. Experiments show that our method achieves average classification accuracy of 92.17%.

Our method works best with the drawing behavior as follows:

1. drawing strokes occur with the fingertip touching desk (with a certain degree of strength);
2. transitional strokes occur with the fingertip moving swiftly above the desk;
3. users do not "doodle" (after a drawing stroke, the finger moves directly to the staring position of next drawing stroke).

We believe this behavior is typical of most people.

The method enables us to build a virtual drawing desk with a minimum hardware requirement: a regular desk (no touch sensors) and a camera that connects to a PC. Further, a user is able to draw on a desk fluently with no need of inserting extra "artificial" gestures into drawing sequences.

The menus in SmartCanvas are pie-shape, making use of the reported gain over linear menus. Instead of "move-to-and-select", a "rotate-to-and-select" approach is used for "pie" menu selection.

The major disadvantage of the SmartCanvas system is that strokes are rendered on a screen whereas drawing is performed on a desk. A user's eyes switch between the screen and the desk frequently. Also, fine tuning of drawing is difficult by using fingertips. Further research is needed to address these problems.

6. REFERENCES

[1] L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale color features, hierarchical models and particle filtering. In *Proc. of Face and Gesture 2002*, pages 423–428, 2002. Demo video is available at http://www.nada.kth.se/cvap/gvmdi/DrawBoard2.mpg.

[2] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *Proc. of the ACM Conf. on Human Factors in Computing System (CHI 88)*, pages 95–100, 1988.

[3] X. Chen, H. Koike, Y. Nakanishi, K. Oka, and Y. Sato. Two-handed drawing on augmented desk system. In *Proc. of 2002 International Conference on Advanced Visual Interfaces (AVI 2002)*, May 2002.

[4] J. Davis and M. Shah. Visual gesture recognition. *Vision, Image and Signal Processing*, 141(2), pages 101–106, 1994.

[5] D. Hall, C. L. Gal, J. Martin, O. Chomat, T. Kapuscinski, and J. L. Crowley. Magicboard: a contribution to an intelligent office environment. In *Proc. of the International Symposium on Intelligent Robotic Systems (SIRS 99)*, pages 215–223, 1999.

[6] D. Heckenberg and B. C. Lovell. Mime: a gesture-driven computer interface. In *Proc. of SPIE vol. 4067, Visual Communications and Image Processing*, pages 261–268, 2000.

[7] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *Tech. Rep. CRL 98/11, Compaq Cambridge Research Lab*, 1998.

[8] J. Martin and J.-B. Durand. Automatic handwriting gestures recognition using hidden markov models. In *Proc. of Face and Gesture 2000*, pages 403–409, 2000.

[9] J. B. Martinkauppi, M. N. Soriano, and M. H. Laaksonen. Behavior of skin color under varying illumination seen by different cameras at different color spaces. In *Proc. of SPIE vol. 4301, Machine Vision Applications in Industrial Inspection IX*, pages 102–113, 2001.

[10] M. R. Mine, F. P. Brooks Jr., and C. H. Sequin. Moving objects in space: exploiting proprioception in virtual-environment interaction. In *Proc. of Siggraph 97*, pages 19–26, 1997.

[11] K. Oka, Y. Sato, and H. Koike. Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems. In *Proc. of Face and Gesture 2002*, 2002.

[12] G. Welch and G. Bishop. An introduction to the kalman filter. In *Siggraph 2001 course material*, 2001. Available at http://www.cs.unc.edu/ ~welch/kalman/.

[13] P. Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM*, 36(7), pages 86–97, 1993.

APPENDIX

A. STROKE SMOOTHNESS

```
Input: v⃗[1..k]
Output: β(v⃗)
1:  count = 0;
2:  For i = 2 To k − 1 Do
3:    If (v⃗[i] − v⃗[i − 1]) · (v⃗[i + 1] − v⃗[i]) < 0 Then
4:      count + +;
5:    End-If
6:  End-For
7:  Return: count/(k−2)
```