INTERACTION GESTUELLE

SOMMAIRE

1. Qu'est ce qu'un geste ?
2. La place du geste dans les systèmes interactifs
3. Comment capter un geste ?
4. Comment reconnaitre un geste ?

QU'EST CE QU'UN GESTE ?

Le Canal Gestuel

- Permet d'agir sur le monde physique
  - Canal d'information
  - Moyen d'émission
  - et de réception d'informations

Trois fonctions souvent liées [Cadoz 94]

- *Epistémique*
- *Ergotique*
- *Sémiotique*

C. Cadoz (1994) *Le geste canal de communication homme/machine – la communication « instrumentale ».* Techniques et science informatiques, vol. 13 pp 31-61

QU'EST CE QU'UN GESTE ? FONCTION EPISTÉMIQUE

Le geste du toucher : **connaître**

La main joue le rôle d'organe de perception

- Perception haptique ou tactilo-proprio-kinesthésique
  - Système perceptif lié au toucher et à la kinesthésie
    - Tactile : texture, température (capteurs de la peau)
    - Kinesthésie : perception des positions, trajectoire, poids (capteurs des articulations et de l'oreille)
  - Mouvements d'exploration
- Perception proprioceptive
  - positions du corps dans l'espace
  - des parties du corps les unes par rapport aux autres

QU'EST CE QU'UN GESTE ? FONCTION ERGOTIQUE

Le geste moteur : **agir**

La main joue le rôle d'organe moteur et agit sur le monde physique pour le transformer

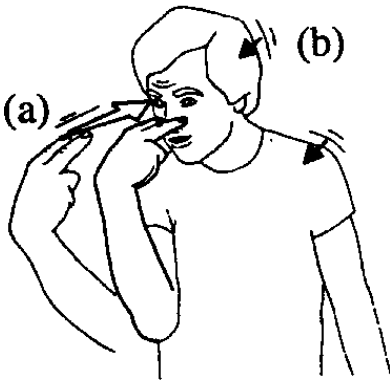
Elle applique à un objet des forces qui vont provoquer une déformation ou un déplacement

QU'EST CE QU'UN GESTE ? FONCTION SÉMIOTIQUE

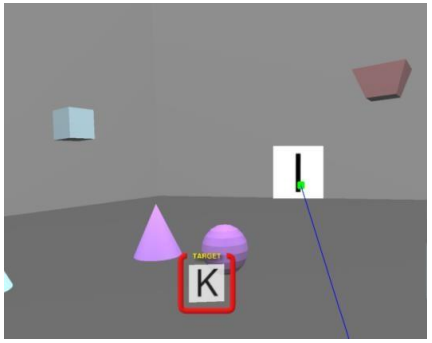
La communication gestuelle : **faire connaître**

La main joue le rôle d'organe d'émission d'information

- à destination de l'environnement
  - Diversité des gestes et différents niveaux de communication
  - Vocabulaire réduit (gestes des plongeurs, grutiers, courtiers)
- Geste co-verbal (simultané à la parole, illustre ou complète le message verbal)
- Langage des signes



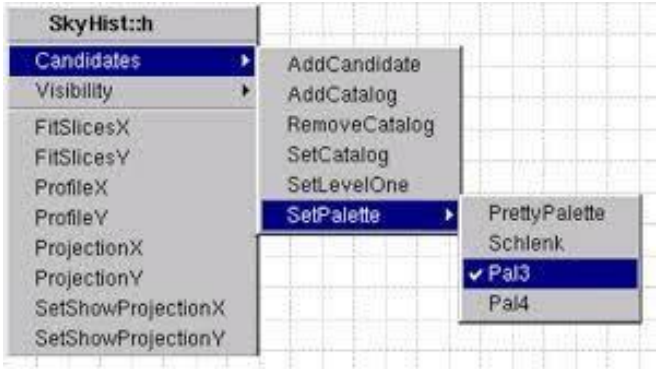
LA PLACE DU GESTE DANS LES SYSTÈMES INTERACTIFS



Le geste est **partout** !

En tant qu'entrée

- Pointage ou manipulation d'objets

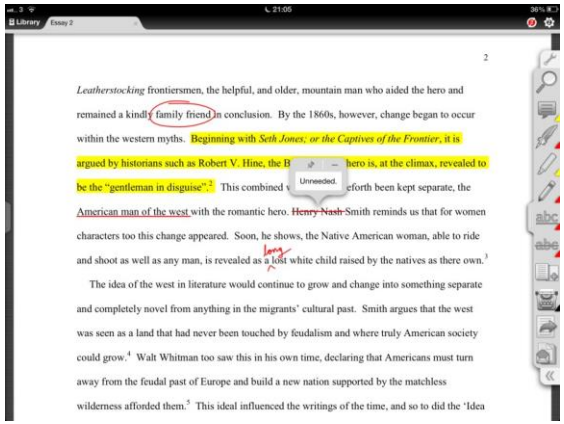


LA PLACE DU GESTE DANS LES SYSTÈMES INTERACTIFS

Le geste est **partout** !

En tant qu'entrée

- Dessins
- Des marques et des signes (souligné; flèches, ..)

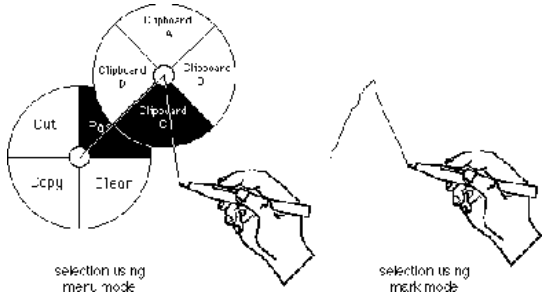


Laisse une **trace**

Mais sans <u>interprétation</u>

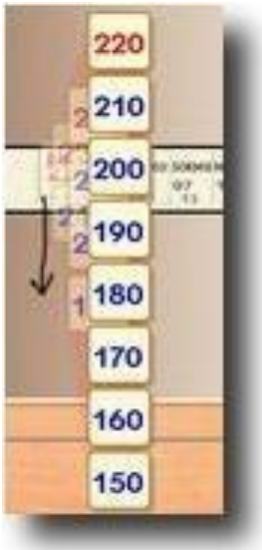
LA PLACE DU GESTE DANS LES SYSTÈMES INTERACTIFS

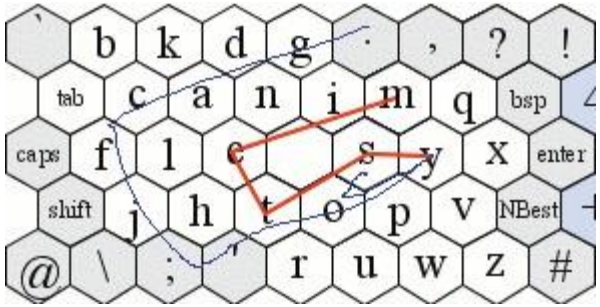
Le geste est **partout** !

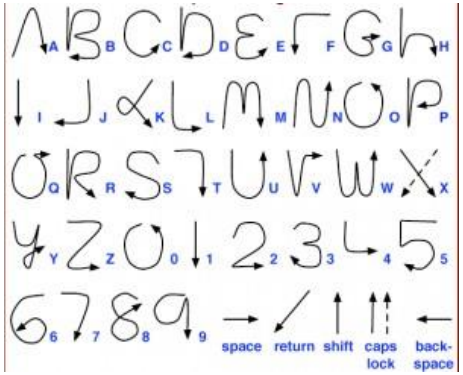


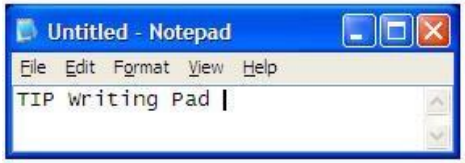
En tant qu'entrée

- Commandes gestuelles
- Reconnaissance d'alphabet
- Reconnaissance d'écriture









LA PLACE DU GESTE DANS LES SYSTÈMES INTERACTIFS

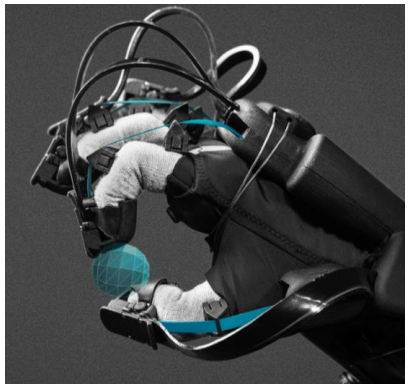
Le geste est **partout** !

En tant que sortie

- Mouvements générés (synthèse de geste)
- Par des informations accessibles au toucher

LA PLACE DU GESTE DANS LES SYSTÈMES INTERACTIFS

Le geste est **partout** !



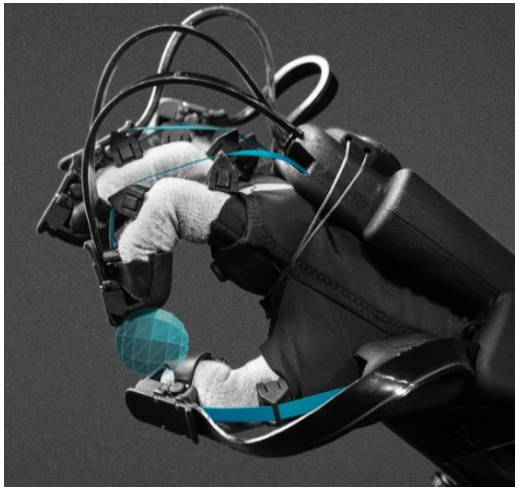
En tant qu'entrée **ET** sortie

- Dispositifs à retours d'effort
  - Souris
  - Stylos
  - Gants
- souris tactiles

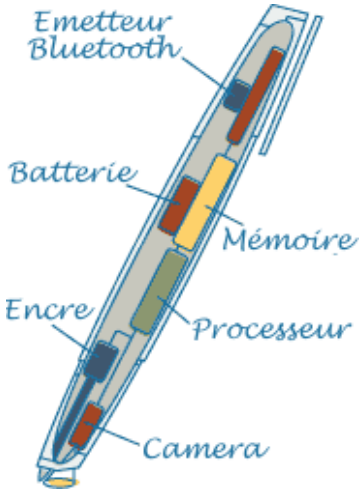
COMMENT CAPTER UN GESTE ?

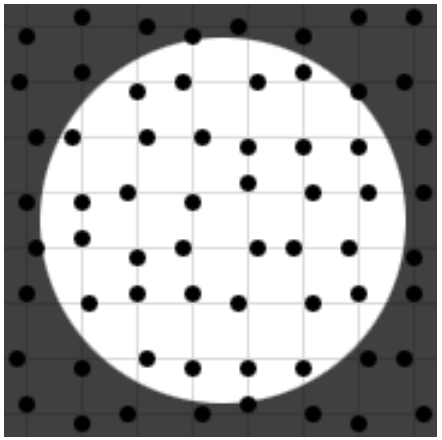
Grande diversité des dispositifs de capture ?

- À base de vision (caméra)
- Gants numériques
- Capteurs
- stylo
- Écran tactile

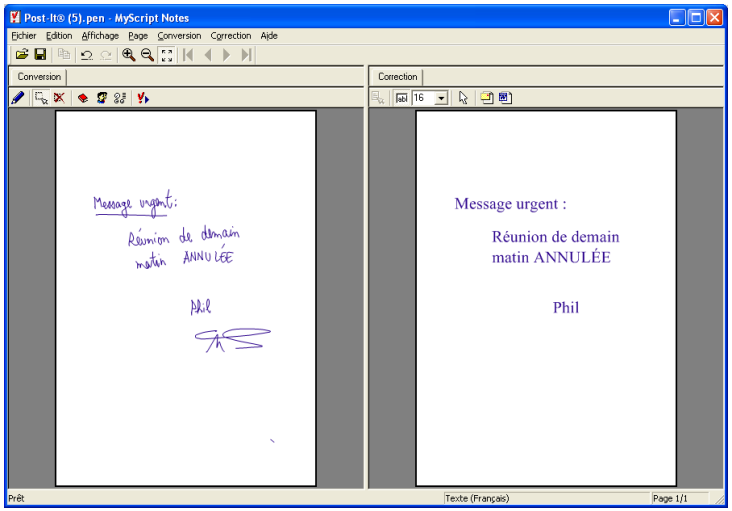


(FOCUS SUR LES STYLOS ANOTO)





```
- <Seg>
    <Fs stk="15" pt="0" />
    <Ls stk="28" pt="2" />
    <Cand txt="réunion de demain" score="0.9741058" />
  - <Seg>
      <Fs stk="15" pt="0" />
      <Ls stk="22" pt="13" />
      <Cand txt="réunion" score="0.9614716" />
      <Cand txt="Réunion" score="0.9103851" />
      <Cand txt="réunions" score="0.8316498" />
    </Seg>
  - <Seg>
      <Fs stk="23" pt="0" />
      <Ls stk="24" pt="20" />
      <Cand txt="de" score="1" />
      <Cand txt="dl" score="0.6507874" />
      <Cand txt="des" score="0.5481415" />
    </Seg>
```



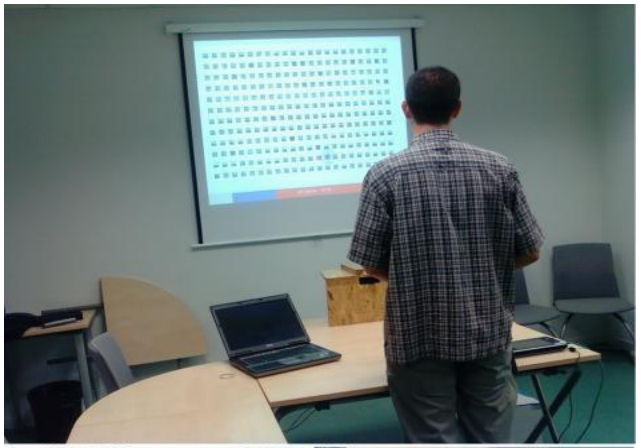
INTERACTION DIRECTE ET INDIRECTE

Interaction directe

- Ecran tactile
  - Pas besoin de pointeur
  - Interaction co-localisée

Interaction indirecte

- Espace moteur
- Espace visuel



INTERACTION DIRECTE ET INDIRECTE ESPACES MOTEUR ET VISUEL

Espace physique/moteur

- Degré de liberté
  - Translation
  - Rotation
- Dispositifs
  - Isotonique
    - Valeurs mesurées : Position / Vitesse
  - Isométrique
    - Valeurs mesurées : Force / Déplacement

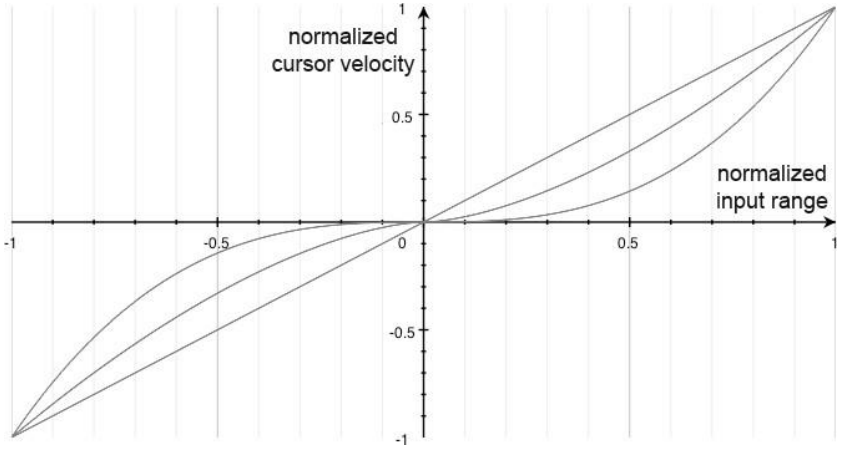
Espace virtuel/visuel

- déplacement d'un pointeur
  - Position
  - Vitesse

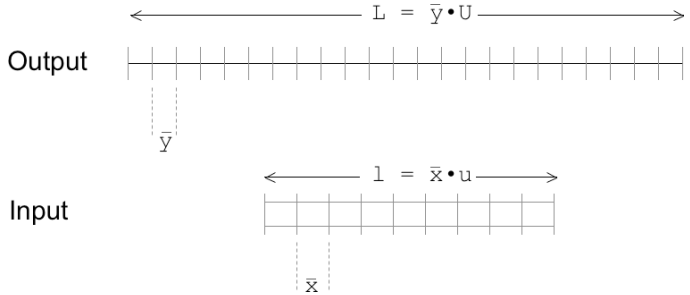
INTERACTION DIRECTE ET INDIRECTE FONCTION DE TRANSFERT

Relations privilégiées entre moteur et visuel [Zhai 95]

- Dispositif isotonique → contrôle de position
- Dispositif isométrique → contrôle de vitesse



Fonction de transfert [Casiez 12]



S. Zhai (1995) Human Performance in Six Degree of Freedom Input Control, Ph.D. Dissertation, University of Toronto.

G. Casiez (2012) *Du mouvement à l'interaction et au geste : études, techniques, outils et périphériques,* Habilitation à diriger des recherches, Université Lille 1

EVALUER ET OPTIMISER UN MOUVEMENT

(FORCÉMENT) LA LOI DE FITTS

$$ID = \log_2\left(\frac{D}{W} + 1\right),$$

Prédire le temps nécessaire pour atteindre une cible !

→ dépend de la distance et la taille de la cible

$$MT_{\text{Predicted}} = a + b \times ID.$$

A été adaptée à plusieurs dimensions

→ Norme ISO pour l'évaluation d'un dispositif de pointage

Soukoreff, R. W., & MacKenzie, I. S. (2004). Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI. International Journal of Human-Computer Studies, 61, 751-789.

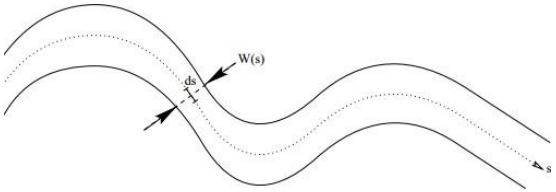
ISO. 2002. 9241–9. 2000. Ergonomics requirements for office work with visual display terminals (VDTs) – Part 9: Requirements for non-keyboard input devices. International Organization for Standardization (2002)

ISO. 2012. 9241–411. 2012. Ergonomics of human-system interaction – Part 411: Evaluation methods for the design of physical input devices. International Organization for Standardization (2012)

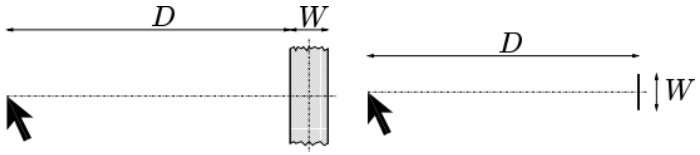
(FORCÉMENT) LA LOI DE FITTS

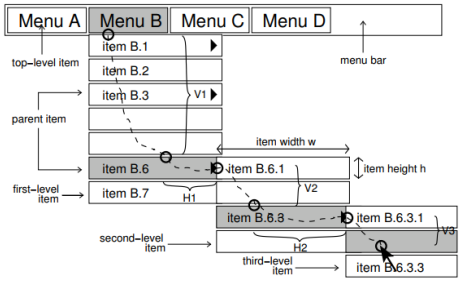
Steering Law : extension à la navigation dans un tunnel

$$T = a + b \int_C \frac{ds}{W(s)}$$



Laws of crossing

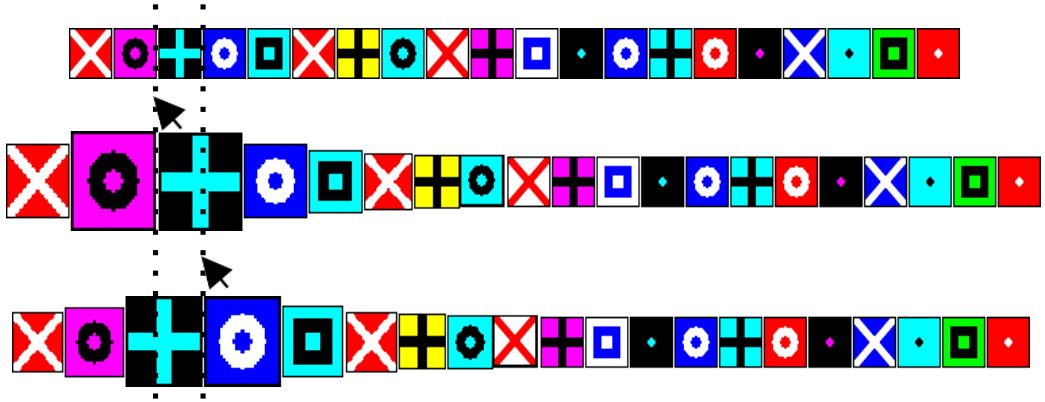


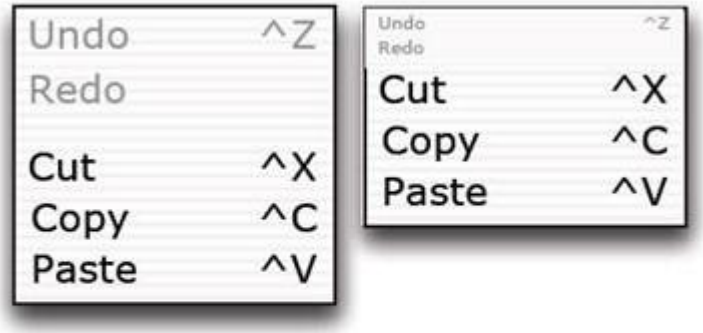


Accot, J., & Zhai, S. (1999). Performance evaluation of input devices in trajectory-based tasks: an application of the steering law. In Proc of CHI'99, pp. 466-472, ACM.

Johnny Accot and Shumin Zhai. 2002. More than dotting the i's --- foundations for crossing-based interfaces. In Proc of CHI '02, pp. 73-80, ACM

DES OPTIMISATIONS





Menu redesign

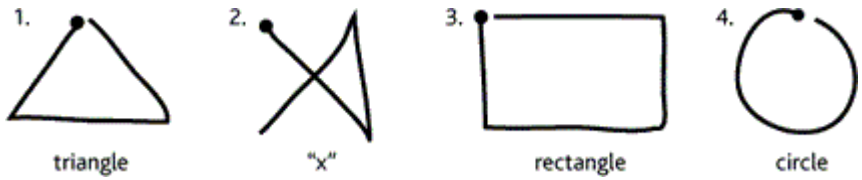
(a) unchanged visual version (b) motor space version

COMMENT RECONNAÎTRE UN GESTE ?

DÉPEND DES CARACTÉRISTIQUES

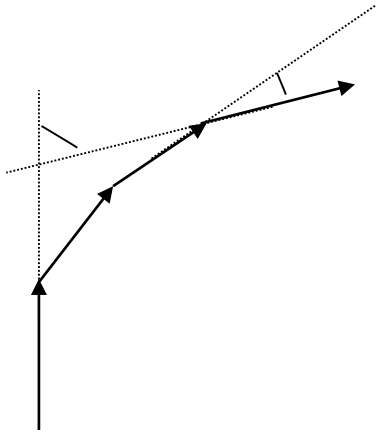
Geste

- Unistroke
- multistroke



Tracé

- Points caractéristiques : départ / arrivée
- Ensemble de points ordonnés
- Fréquence d'échantillonnage
- Vitesse d'exécution du geste

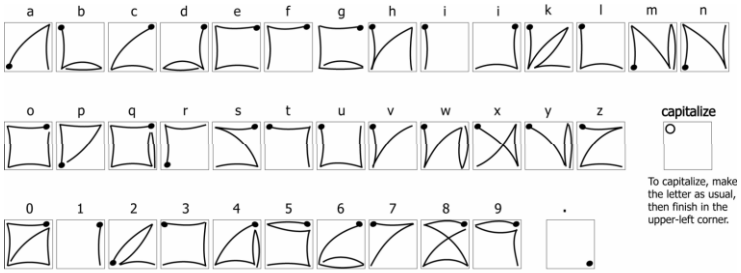


Chaque geste est associé à une commande

QUELQUES TECHNIQUES DE RECONNAISSANCE

Utilisation de « *régions géométriques* »

- Gestes décrits par des séquences de chiffres
  - Libstroke
  - EdgeWrite



Classifieurs statistiques (Ex: Rubine)

Modèles de markov cachés

Réseaux de neurones

Méthodes ad-hoc

Les deux techniques les plus utilisées : Rubine et Dynamic Time Warping (DTW – Déformation Temporelle Dynamique)

R. Plamondon, S.N. Srihari (2000) *On-line and off-line handwriting recognition: A comprehensive survey.* IEEE Trans. Pattern Analysis & Machine Int., Vol. 22 (1), pp. 63-84

(FOCUS) ALGORITHME DE RUBINE

Chaque geste entré (ou exemple de geste) est réduit à un vecteur de caractéristiques ("***feature vector***") et correspond donc à un point multidimensionnel.

Il s'agit alors de classer ces points parmi les catégories de gestes.

Taux de reconnaissance > 95%

Rubine, D. 1991. Specifying gestures by example. In Proceedings of the 18th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '91. ACM, New York, NY, 329-337. DOI= http://doi.acm.org/10.1145/122718.122753

(FOCUS) ALGORITHME DE RUBINE

Liste de points en entrée

Élimination des points trop proches.

Calcul d'un vecteur de caractéristiques statistiques (13 fonctions)

Comparaison aux gestes de référence

→ **le geste avec le score le plus élevé est renvoyé**

pressing of a mouse button, while the end is indicated by releasing the button or ceasing to move the mouse.

Each gesture is an array g of P time-stamped sample points:

$$g_p = (x_p, y_p, t_p) \qquad 0 \le p < P$$

Some simple preprocessing is done to eliminate jiggle: an input point within 3 pixels of the previous input point is discarded.

The gesture recognition problem is stated as follows: There is a set of C gesture classes, numbered 0 through $C-1$. Each class is specified by example gestures. Given an input gesture g, determine the class to which g belongs (*i.e.* the class whose members are most like g).

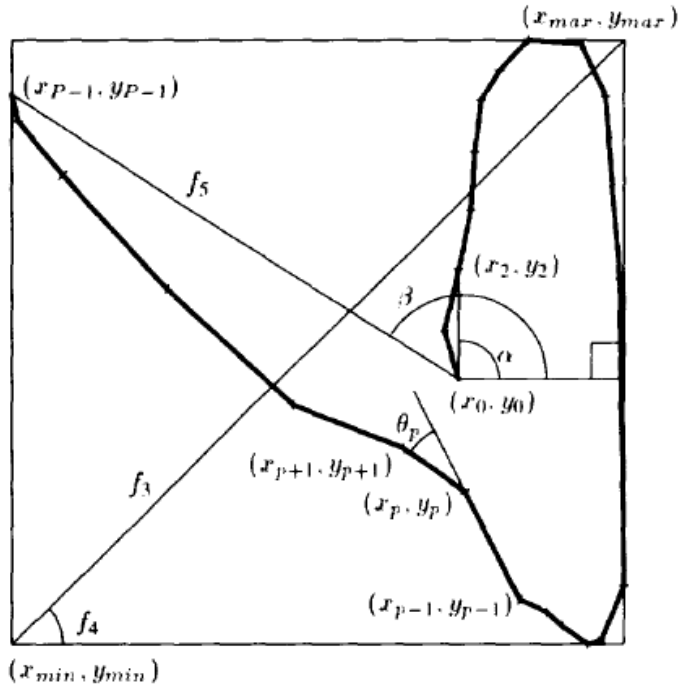
Statistical gesture recognition is done in two steps. First, a vector of features, $\mathbf{f} = [f_1, \ldots, f_F]$, is extracted from the input gesture. Then, the feature vector is classified as one of the C possible gestures via a linear machine.

4.1 The Features

Features were chosen according to the following criteria. Each feature should be incrementally computable in constant time per input point, which allows arbitrarily large gestures to be handled as efficiently as small ones. Since the classification algorithm performs poorly when a class has a feature with a multimodal distribution, a small change in the input should result in a correspondingly small change in each feature. Each feature should be meaningful so that is can be used in gesture semantics as well as for recognition. Finally, there should be enough features to provide differentiation between all gestures that might reasonably be expected, but, for efficiency reasons, there should not be too many.

Figure 6 shows the actual features used, both geometrically and algebraically. The features are the cosine (f_1) and the sine (f_2) of the initial angle of the gesture, the length (f_3) and the angle (f_4) of the bounding box diagonal, the distance (f_5) between the first and the last point, the cosine (f_6) and the sine (f_7) of the angle between the first and last point, the total gesture length (f_8), the total angle traversed (f_9), the sum of the absolute value of the angle at each mouse point (f_{10}), the sum of the squared value of those angles (f_{11}), the maximum speed (squared) of the gesture (f_{12}), and the duration of the gesture (f_{13}).

An angle's cosine and sine are used as features rather than the angle itself to avoid a discontinuity as the angle passes through 2π and wraps to 0. The "sharpness" feature, f_{11}, is needed to distinguish between smooth gestures and those with sharp angles, e.g. "U" and "V." Features f_{12} and f_{13} add a dynamic component so that gestures are not simply static pictures. Some applications may wish to disable these two features. The initial angle features, f_1 and f_2, are computed from the first and third mouse point because the result is generally less noisy than when computed from the first two points.



$$f_1 = \cos\alpha = (x_2 - x_0)/\sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}$$

$$f_2 = \sin\alpha = (y_2 - y_0)/\sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}$$

$$f_3 = \sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2}$$

$$f_4 = \arctan\frac{y_{max} - y_{min}}{x_{max} - x_{min}}$$

$$f_5 = \sqrt{(x_{P-1} - x_0)^2 + (y_{P-1} - y_0)^2}$$

$$f_6 = \cos\beta = (x_{P-1} - x_0)/f_5$$

$$f_7 = \sin\beta = (y_{P-1} - y_0)/f_5$$

Let $\Delta x_p = x_{p+1} - x_p \qquad \Delta y_p = y_{p+1} - y_p$

$$f_8 = \sum_{p=0}^{P-2} \sqrt{\Delta x_p^2 + \Delta y_p^2}$$

Let $\theta_p = \arctan\dfrac{\Delta x_p \Delta y_{p-1} - \Delta x_{p-1}\Delta y_p}{\Delta x_p \Delta x_{p-1} + \Delta y_p \Delta y_{p-1}}$

$$f_9 = \sum_{p=1}^{P-2} \theta_p$$

$$f_{10} = \sum_{p=1}^{P-2} |\theta_p|$$

$$f_{11} = \sum_{p=1}^{P-2} \theta_p^2$$

Let $\Delta t_p = t_{p+1} - t_p$

$$f_{12} = \max_{p=0}^{P-2} \frac{\Delta x_p^2 + \Delta y_p^2}{\Delta t_p^2}$$

$$f_{13} = t_{P-1} - t_0$$

Figure 6: Features used to identify strokes

(FOCUS) $1 RECOGNIZER

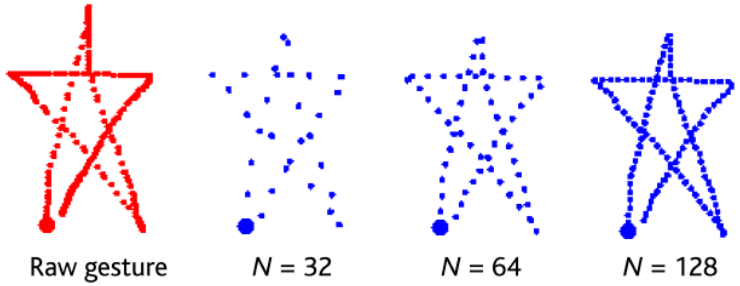


Figure 4. A star gesture resampled to *N*=32, 64, and 128 points.

Utilise seulement des opérations mathématiques de base

Simple à implémenter, sans libraries

Rapide

Bon pour prototyper des interfaces gestuelles, même sur des plateformes ou langages moins performants (comme JavaScript)

Une des étapes clés : rééchantillonage du geste

Taux de reconnaissance supérieur à Rubine

J.O. Wobbrock, A.D. Wilson, Y. Li (2007). *Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes.* UIST'07, pp. 159-168.

(FOCUS) $1 RECOGNIZER

1) L'utilisateur réalise un geste

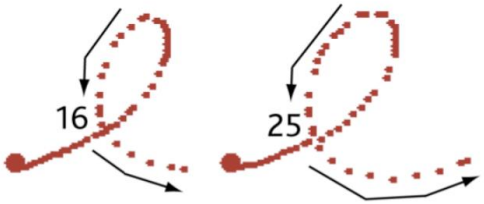
- Le geste est représenté par une liste ordonnée de points

2) Ce geste est comparé à un ensemble de gestes de référence (*templates*) en utilisant une mesure de distance euclidienne

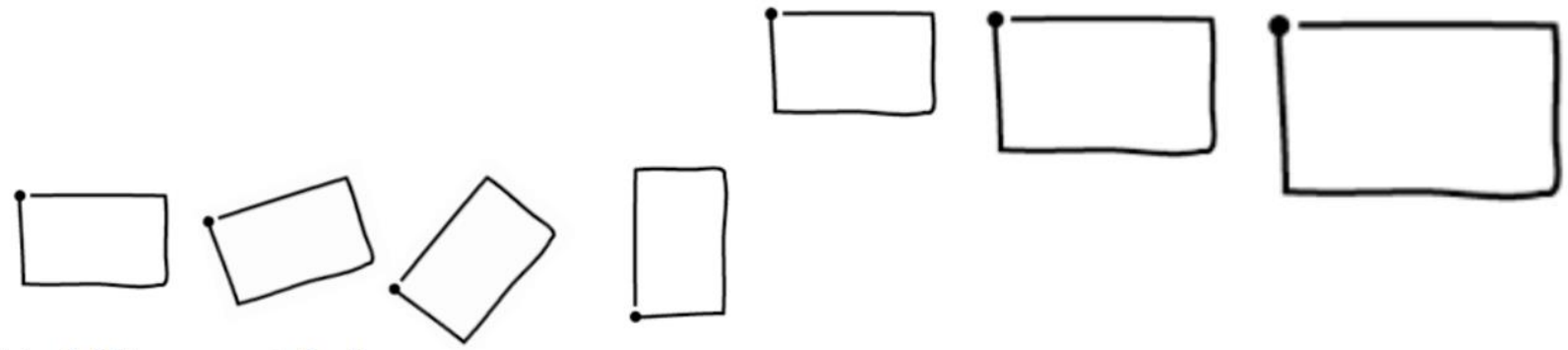
3) Le geste reconnu est celui pour lequel cette distance est minimale

(FOCUS) $1 RECOGNIZER - PROBLÈMES

Le nombre de points pour définir un geste va dépendre de la vitesse d'exécution, de la fréquence d'échantillonnage du périphérique, …

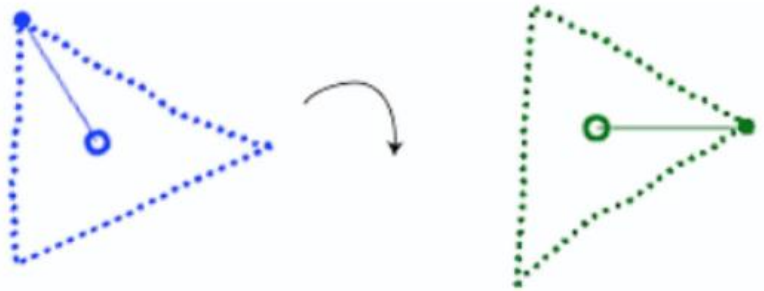


Un geste peut être réalisé à différentes positions, suivant différentes orientations et différentes échelles



(FOCUS) $1 RECOGNIZER - ETAPES

1. **Ré-échantillonnage** du geste pour être invariant à la fréquence d'acquisition et la vitesse d'exécution du geste
2. **Ré-orientation** du geste pour être invariant à l'orientation suivant laquelle il est exécuté
3. **Mise à l'échelle et translation** pour être invariant à l'échelle de réalisation du geste et la position à laquelle il est exécuté
4. **Reconnaissance** du geste



handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Analysis & Machine Int. 22* (1), 63-84.
22. Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992) *Numerical Recipes in C.* Cambridge Univ. Press.
23. Rubine, D. (1991) Specifying gestures by example. *Proc. SIGGRAPH '91.* New York: ACM Press, 329-337.
24. Salvador, S. and Chan, P. (2004) FastDTW: Toward accurate dynamic time warping in linear time and space. *3rd Wkshp. on Mining Temporal and Sequential Data, ACM KDD '04.* Seattle, Washington (August 22-25, 2004).
25. Sezgin, T.M. and Davis, R. (2005) HMM-based efficient sketch recognition. *Proc. IUI '05.* New York: ACM Press, 281-283.
26. Stojmenović, M., Nayak, A. and Zunic, J. (2006) Measuring linearity of a finite set of points. *Proc. CIS '06.* Los Alamitos, CA: IEEE Press, 1-6.
27. Swigart, S. (2005) Easily write custom gesture recognizers for your Tablet PC applications. *Tablet PC Technical Articles.*
28. Tappert, C.C. (1982) Cursive script recognition by elastic matching. *IBM J. of Research & Development 26* (6), 765-771.
29. Tappert, C.C., Suen, C.Y. and Wakahara, T. (1990) The state of the art in online handwriting recognition. *IEEE Trans. Pattern Analysis & Machine Int. 12* (8), 787-808.
30. Vermunt, J.K. (1997) *Log-linear Models for Event Histories.* Thousand Oaks, CA: Sage Publications.
31. Wilson, A.D. and Shafer, S. (2003) XWand: UI for intelligent spaces. *Proc. CHI '03.* New York: ACM Press, 545-552.
32. Zhai, S. and Kristensson, P. (2003) Shorthand writing on stylus keyboard. *Proc. CHI '03.* New York: ACM Press, 97-104.

APPENDIX A – $1 GESTURE RECOGNIZER

Step 1. Resample a *points* path into *n* evenly spaced points.

```
RESAMPLE(points, n)
1   I ← PATH-LENGTH(points) / (n − 1)
2   D ← 0
3   newPoints ← points_0
4   foreach point p_i for i ≥ 1 in points do
5     d ← DISTANCE(p_{i-1}, p_i)
6     if (D + d) ≥ I then
7       q_x ← p_{i-1}_x + ((I − D) / d) × (p_{i_x} − p_{i-1_x})
8       q_y ← p_{i-1}_y + ((I − D) / d) × (p_{i_y} − p_{i-1_y})
9       APPEND(newPoints, q)
10      INSERT(points, i, q)   // q will be the next p_i
11      D ← 0
12    else D ← D + d
13  return newPoints
PATH-LENGTH(A)
1   d ← 0
2   for i from 1 to |A| step 1 do
3     d ← d + DISTANCE(A_{i-1}, A_i)
4   return d
```

Step 2. Rotate *points* so that their indicative angle is at 0°.

```
ROTATE-TO-ZERO(points)
1   c ← CENTROID(points)   // computes (x̄, ȳ)
2   θ ← ATAN (c_y − points_0_y, c_x − points_0_x)   // for -π ≤ θ ≤ π
3   newPoints ← ROTATE-BY(points, -θ)
4   return newPoints
ROTATE-BY(points, θ)
1   c ← CENTROID(points)
2   foreach point p in points do
3     q_x ← (p_x − c_x) COS θ − (p_y − c_y) SIN θ + c_x
4     q_y ← (p_x − c_x) SIN θ + (p_y − c_y) COS θ + c_y
5     APPEND(newPoints, q)
6   return newPoints
```

Step 3. Scale *points* so that the resulting bounding box will be of $size^2$ dimension; then translate *points* to the origin. BOUNDING-BOX returns a rectangle according to (min_x, min_y), (max_x, max_y). For gestures serving as templates, Steps 1-3 should be carried out once on the raw input points. For candidates, Steps 1-4 should be used just after the candidate is articulated.

```
SCALE-TO-SQUARE(points, size)
1   B ← BOUNDING-BOX(points)
2   foreach point p in points do
3     q_x ← p_x × (size / B_width)
4     q_y ← p_y × (size / B_height)
5     APPEND(newPoints, q)
6   return newPoints
TRANSLATE-TO-ORIGIN(points)
1   c ← CENTROID(points)
2   foreach point p in points do
3     q_x ← p_x − c_x
4     q_y ← p_y − c_y
5     APPEND(newPoints, q)
6   return newPoints
```

Step 4. Match *points* against a set of *templates*. The *size* variable on line 7 of RECOGNIZE refers to the *size* passed to SCALE-TO-SQUARE in Step 3. The symbol φ equals $\frac{1}{2}(-1 + \sqrt{5})$. We use $\theta = \pm 45°$ and $\theta_\Delta = 2°$ on line 3 of RECOGNIZE. Due to using RESAMPLE, we can assume that *A* and *B* in PATH-DISTANCE contain the same number of points, i.e., $|A| = |B|$.

```
RECOGNIZE(points, templates)
1   b ← +∞
2   foreach template T in templates do
3     d ← DISTANCE-AT-BEST-ANGLE(points, T, -θ, θ, θ_Δ)
4     if d < b then
5       b ← d
6       T' ← T
7   score ← 1 − b / 0.5√(size^2 + size^2)
8   return ⟨T', score⟩
DISTANCE-AT-BEST-ANGLE(points, T, θ_a, θ_b, θ_Δ)
1   x_1 ← φθ_a + (1 − φ)θ_b
2   f_1 ← DISTANCE-AT-ANGLE(points, T, x_1)
3   x_2 ← (1 − φ)θ_a + φθ_b
4   f_2 ← DISTANCE-AT-ANGLE(points, T, x_2)
5   while |θ_b − θ_a| > θ_Δ do
6     if f_1 < f_2 then
7       θ_b ← x_2
8       x_2 ← x_1
9       f_2 ← f_1
10      x_1 ← φθ_a + (1 − φ)θ_b
11      f_1 ← DISTANCE-AT-ANGLE(points, T, x_1)
12    else
13      θ_a ← x_1
14      x_1 ← x_2
15      f_1 ← f_2
16      x_2 ← (1 − φ)θ_a + φθ_b
17      f_2 ← DISTANCE-AT-ANGLE(points, T, x_2)
18  return MIN(f_1, f_2)
DISTANCE-AT-ANGLE(points, T, θ)
1   newPoints ← ROTATE-BY(points, θ)
2   d ← PATH-DISTANCE(newPoints, T_points)
3   return d
PATH-DISTANCE(A, B)
1   d ← 0
2   for i from 0 to |A| step 1 do
3     d ← d + DISTANCE(A_i, B_i)
4   return d / |A|
```

(FOCUS) $1 RECOGNIZER - LIMITATIONS

1. Impossible de distinguer un carré d'un rectangle, un ellipse d'un cercle, d'un trait vers la droite ou vers la gauche …