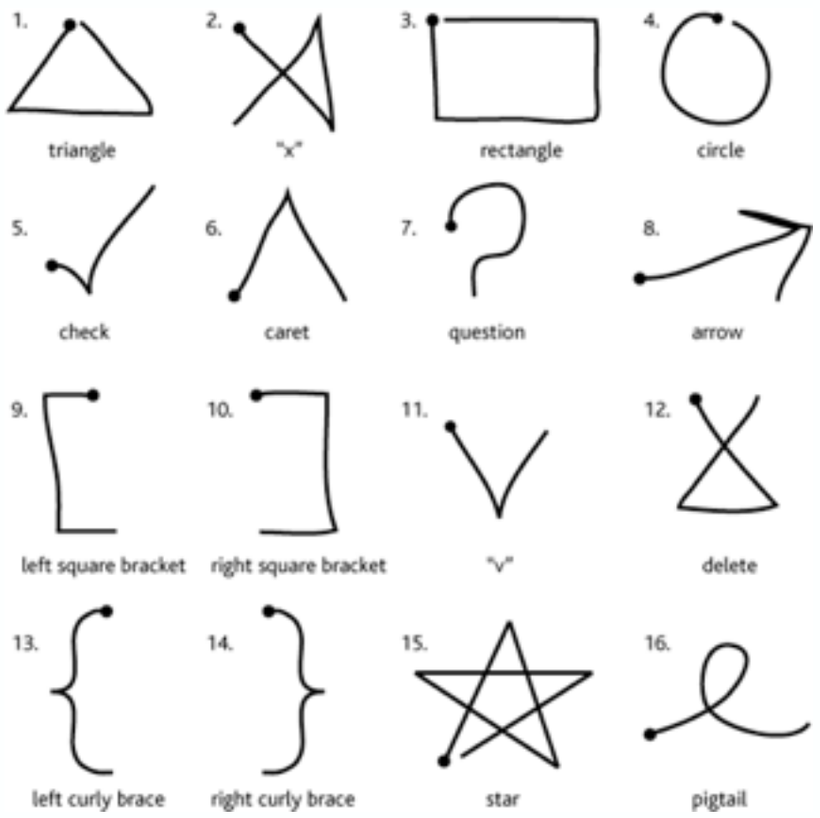
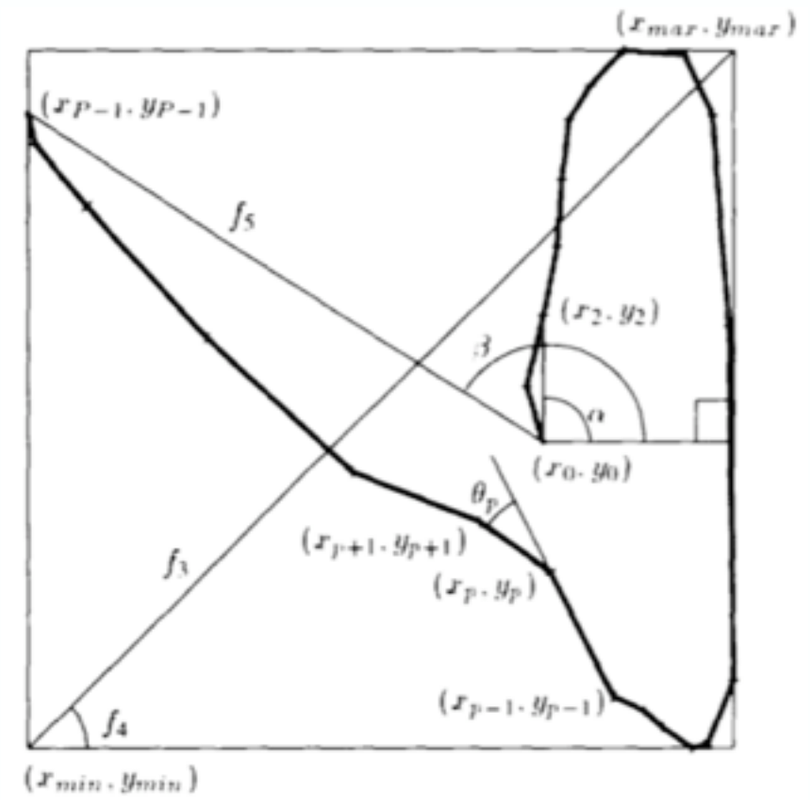
# RECONNAISSANCE DE GESTES

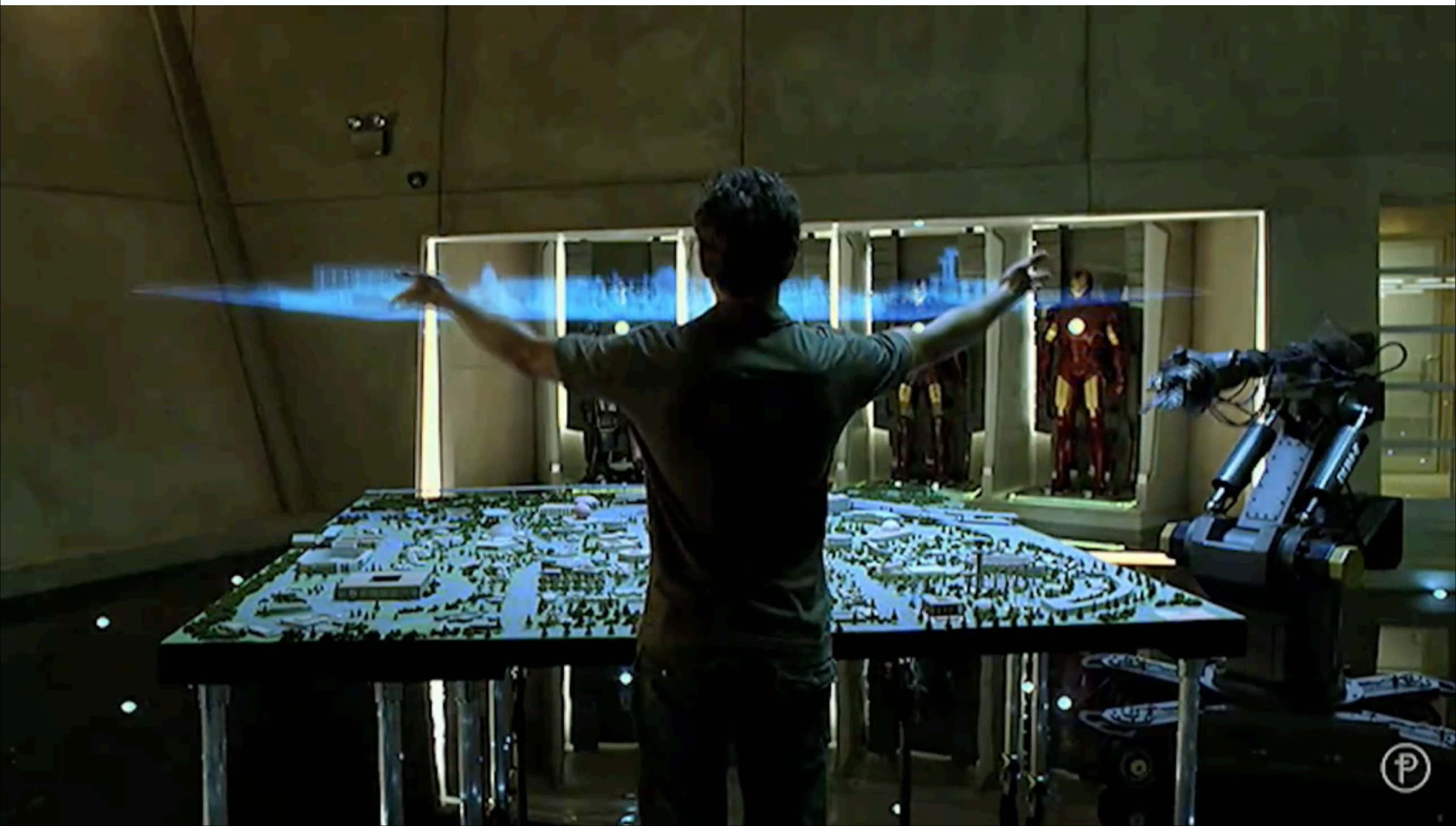


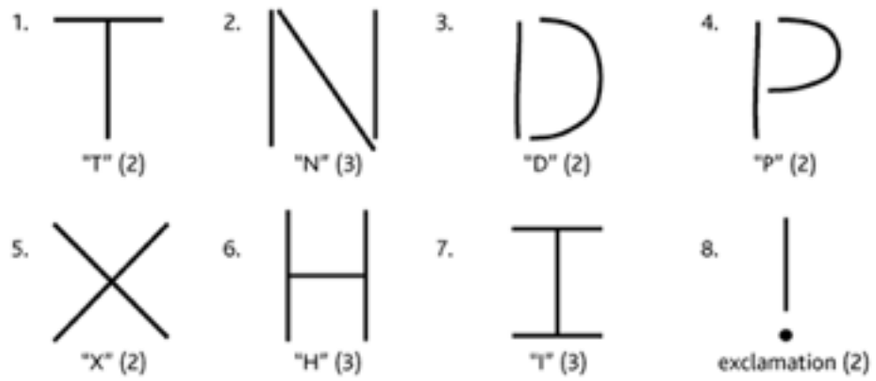

- Thomas Pietrzak www.thomaspietrzak.com
- Supports adaptés de Géry Casiez http://www.lifl.fr/~casiez/
- Telecom Lille 1 - 3D Entertainment Technologies

# MINORITY REPORT

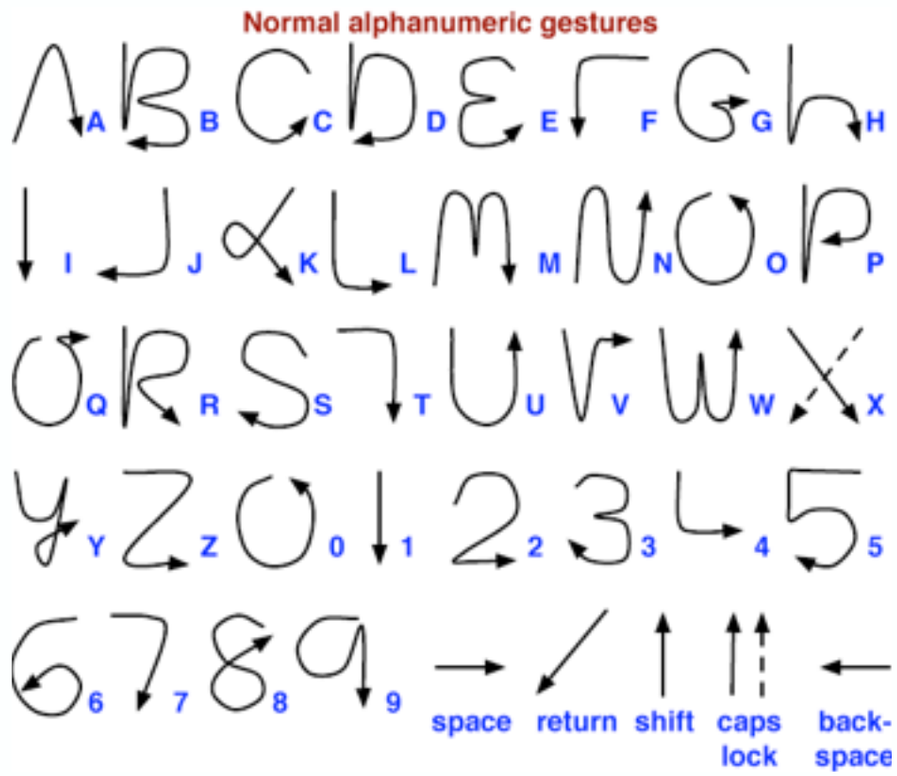
# IRON MAN



# OBJECTIFS




Multistroke




Unistroke - Graffiti (Palm OS)

- Reconnaître des gestes de l'utilisateur
- Les associer à des commandes
- On se limite aux gestes un seul trait (unistroke)

# TECHNIQUE DE RECONNAISSANCE DE GESTES ÉLÉMENTAIRES: LIBSTROKE (1997)

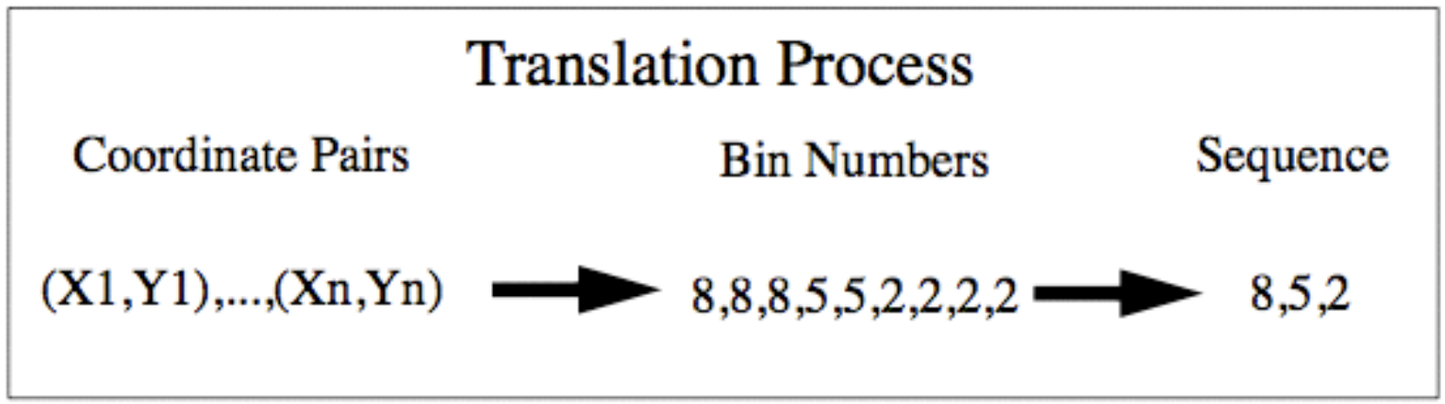
- Librairie disponible dans FVWM http://etla.net/libstroke/
- Les gestes sont décrits par des séquences de chiffres

```
# Strokes
#         num         button context mod. action
Stroke  14789       2      A       N     Exec exec xlock -mode blank&
Stroke  258         2      A       N     Exec exec xterm&
Stroke  563214789   2      A       N     Exec exec exmh&
Stroke  7415963     2      A       N     Exec exec netscape&
Stroke  741236987   2      A       N     Destroy
Stroke  1478963     2      A       N     Popup “Apps”
Stroke  74123       2      A       N     Module “winlist” FvwmWinList
Stroke  74159       2      A       N     Move
Stroke  852         2      A       N     Menu “Apps” Nop
```

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

# LIBSTROKE: ALGORITHME

1. Calcul de la bounding box à partir de minx, maxx,miny, maxy
2. Division de la boite en 9 cellules
3. Étiquetage des points
4. Factorisation
5. Comparaison aux motifs enregistrés




- Extension Firefox: Mouse Gestures Redox

# TECHNIQUES DE RECONNAISSANCE

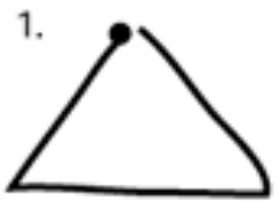
- Classifieurs statistiques
- Modèles de Markov cachés
- Réseaux de neurones
- Méthodes ad-hoc
- 2 techniques fréquemment utilisées:
  - Rubine
  - Dynamic Time Warping (DTW)

# TECHNIQUES DE RECONNAISSANCE

- $1 recognizer
- Simple à implémenter
- Taux de reconnaissance comparables à Rubine et DTW
- 1 seul exemple suffit

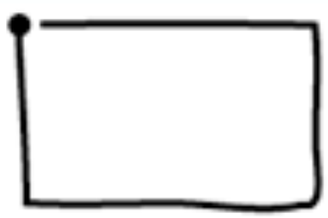
*Jacob O. Wobbrock, Andrew D. Wilson, Yang Li.*
*Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes*
*UIST '07, 159-168.*

# $1 RECOGNIZER



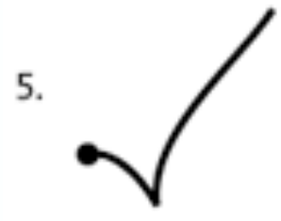

triangle


"x"




rectangle


circle




check


caret


question

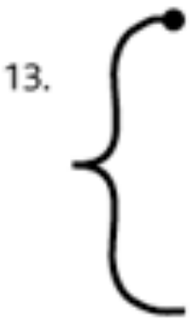

arrow


left square bracket


right square bracket


"v"


delete




left curly brace


right curly brace

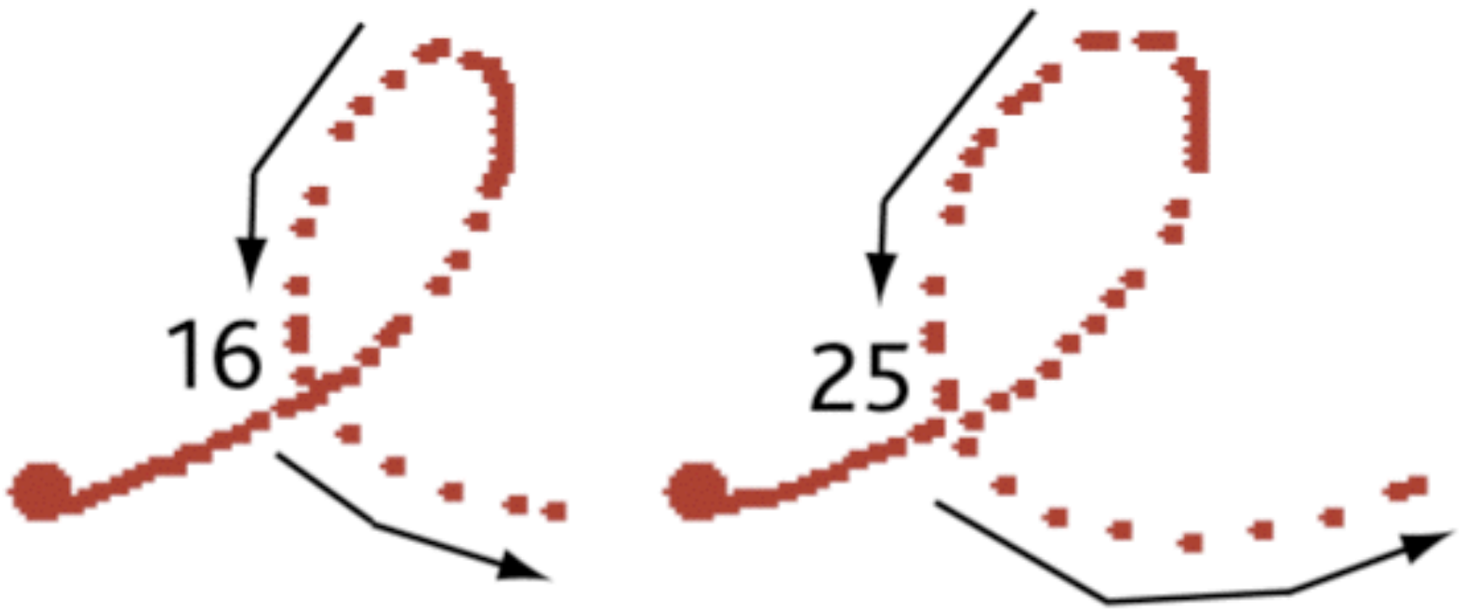

star


pigtail

# ALGORITHME

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
   - Les références sont appelées « templates »
   - On mesure la distance euclidienne
3. Le geste reconnu est celui pour lequel cette distance est minimale

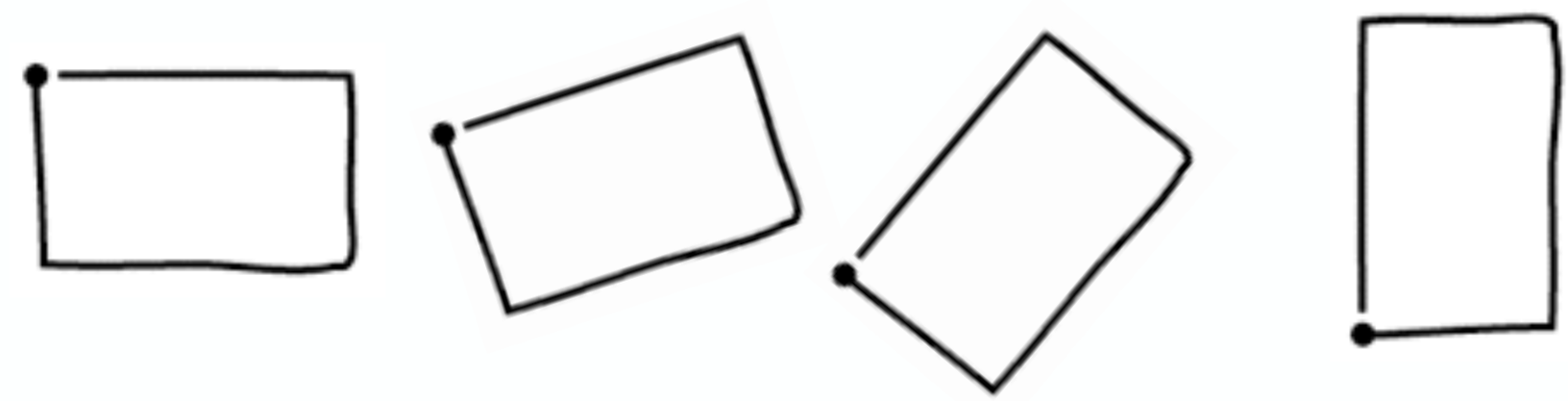
# PROBLÈMES POSÉS




- Le nombre de points d'un geste dépend de :
  - La vitesse d'exécution
  - La fréquence d'échantillonnage du périphérique
  - …

# PROBLÈMES POSÉS

- Un geste peut être réalisé à différentes positions sur l'interface,
- Suivant différentes orientations



- Et différentes échelles

# 4 ÉTAPES

1. Ré-échantillonner le geste
   - ➡Invariant à la fréquence d'acquisition
   - ➡Invariant à la vitesse d'exécution
2. Ré-orientation du geste
   - ➡Invariant à l'orientation
3. Mise à l'échelle et translation
   - ➡Invariant à l'échelle
   - ➡Invariant à la position
4. Reconnaissance du geste

# 1ÈRE ÉTAPE : RÉ-ÉCHANTILLONNAGE

- Le geste est défini par M points ordonnés.
- On veut N points ordonnés équidistants les uns des autres.
- N = 64

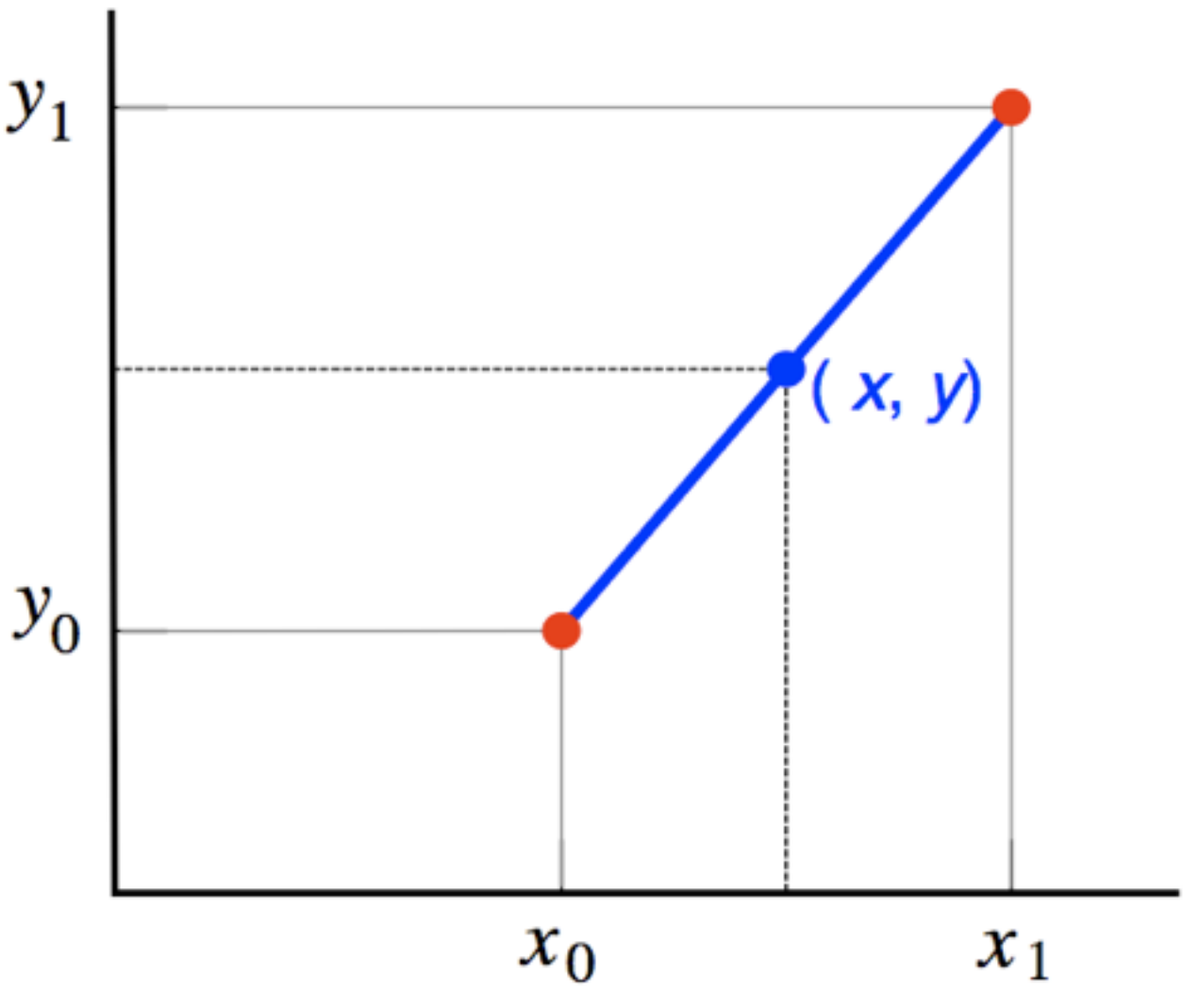
# 1ÈRE ÉTAPE : RÉ-ÉCHANTILLONNAGE

1. Calcul de la distance I entre 2 points :
   - Calcul de la longueur totale du geste.
   - I = longueur / (N-1)
2. Interpolation linéaire sur les points du geste d'origine.

➡ Permet de calculer la distance en prenant les points 2 à 2.

# 1ÈRE ÉTAPE : RÉ-ÉCHANTILLONNAGE

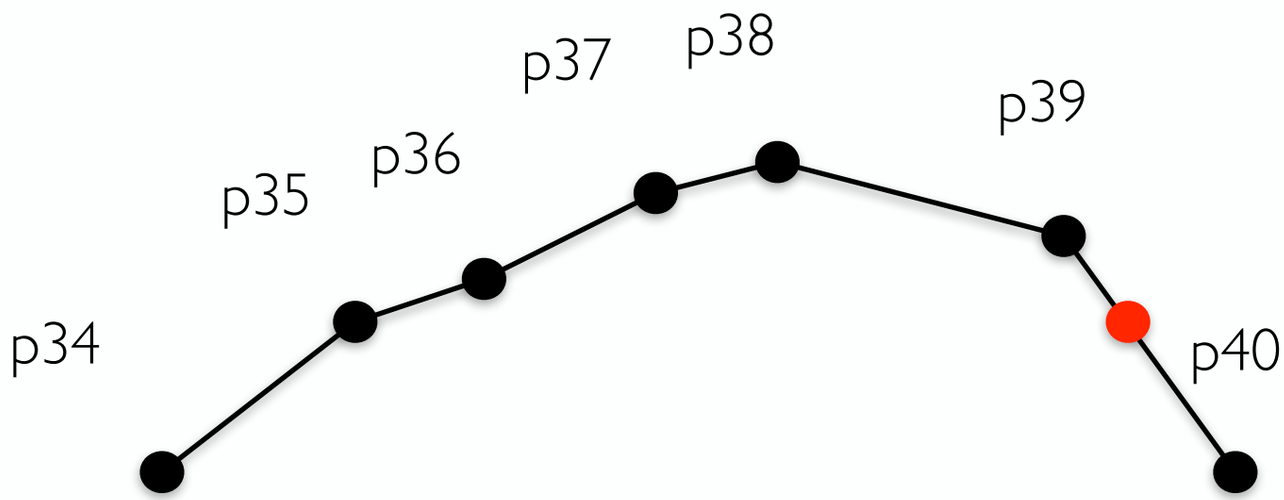
- Interpolation linéaire




$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$$

# 1ÈRE ÉTAPE: RÉ-ÉCHANTILLONNAGE




$$\sum_{i=35}^{40} distance(p_{i-1}, p_i) > I$$

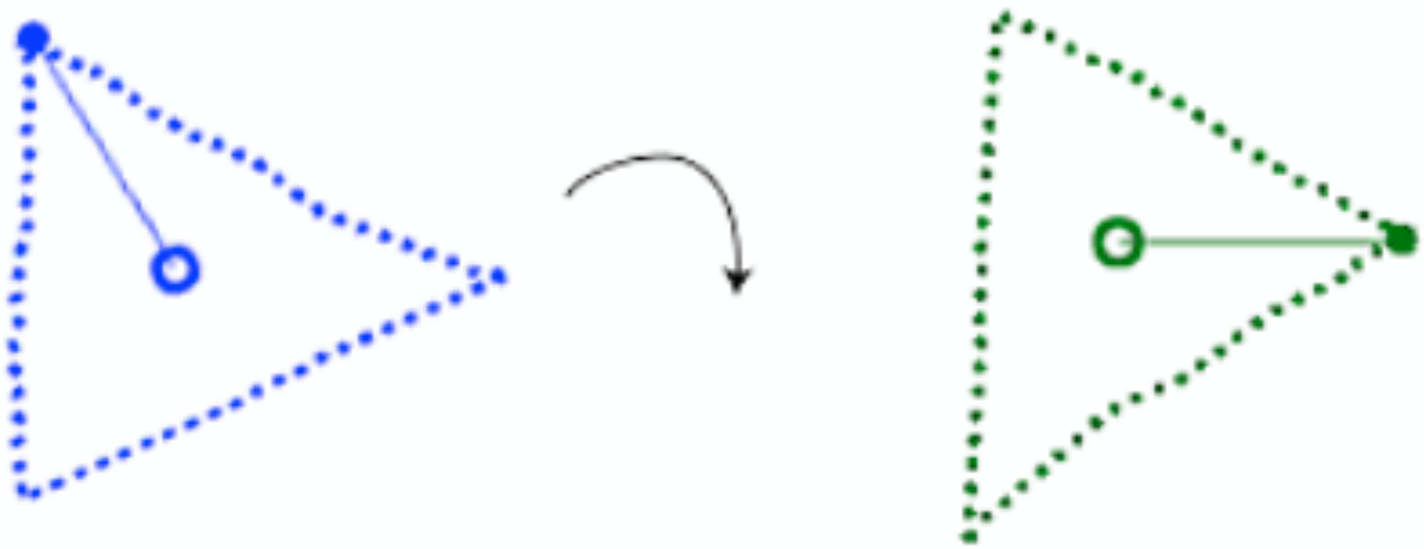
# 1ÈRE ÉTAPE: RÉ-ÉCHANTILLONNAGE

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

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

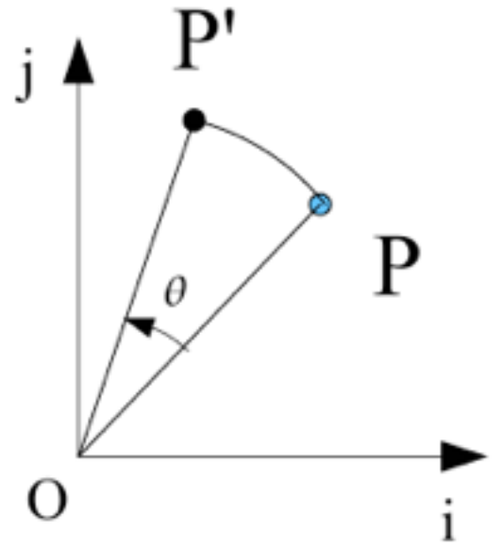
# 2E ÉTAPE : ROTATION « INDICATIVE »

1. Calcul du centre du geste (centroïde)
2. Calcul de l'angle entre :
   - Le centroïde,
   - Le premier point
   - L'horizontale
3. Rotation des points en utilisant cet angle



# 2E ÉTAPE : ROTATION « INDICATIVE »

- Rotation d'un point




$$\begin{cases} x' &= x\cos(\theta) - y\sin(\theta) \\ y' &= x\sin(\theta) + y\cos\theta \end{cases}$$

Comment retrouver ? Soit $\alpha$ l'angle $(i, OP)$ et $r = \|OP'\| = \|OP\|$. Alors

$$\begin{cases} x' &= r\cos(\alpha+\theta) \\ y' &= r\sin(\alpha+\theta) \end{cases}$$

$$\begin{cases} x' &= r\cos(\alpha)\cos(\theta) - r\sin(\alpha)\sin(\theta) \\ y' &= r\cos(\alpha)\sin(\theta) + r\sin(\alpha)\cos(\theta) \end{cases}$$

or $x = r\cos(\alpha)$ et $y = r\sin(\alpha)$

# 2E ÉTAPE : ROTATION « INDICATIVE »

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

ROTATE-TO-ZERO(*points*)

1 $c \leftarrow$ CENTROID(*points*) // computes $(\bar{x}, \bar{y})$
2 $\theta \leftarrow$ ATAN $(c_y - points_{0_y}, c_x - points_{0_x})$ // for $-\pi \leq \theta \leq \pi$
3 *newPoints* $\leftarrow$ ROTATE-BY(*points*, $-\theta$)
4 **return** *newPoints*

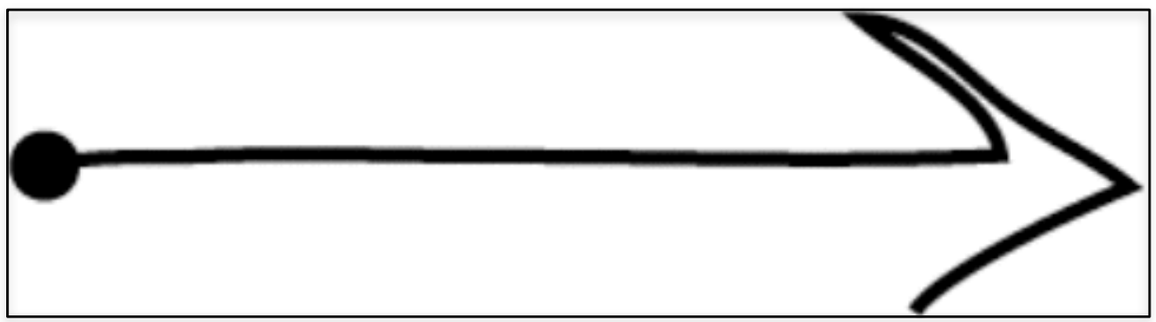
ROTATE-BY(*points*, $\theta$)

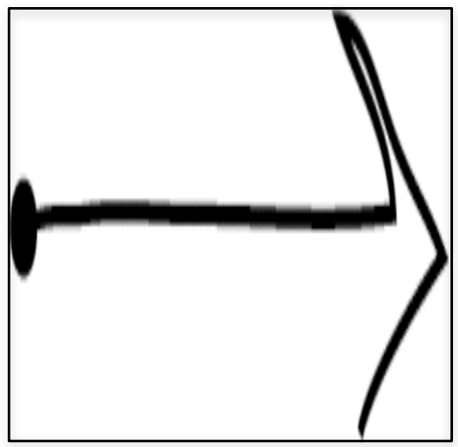
1 $c \leftarrow$ CENTROID(*points*)
2 **foreach** point $p$ in *points* **do**
3 $q_x \leftarrow (p_x - c_x)$ COS $\theta - (p_y - c_y)$ SIN $\theta + c_x$
4 $q_y \leftarrow (p_x - c_x)$ SIN $\theta + (p_y - c_y)$ COS $\theta + c_y$
5 APPEND(*newPoints*, $q$)
6 **return** *newPoints*

# 3E ÉTAPE :
# MISE À L'ÉCHELLE ET TRANSLATION

- Mise à l'échelle non uniforme: on ramène le geste à un carré de référence

1. Calcul de la bounding box
   - Calcul de minx, maxx, miny, maxy
2. Mise à l'échelle
3. Translation à l'origine





# 3E ÉTAPE :
# MISE À L'ÉCHELLE ET TRANSLATION

**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 $B \leftarrow$ BOUNDING-BOX(*points*)
2 **foreach** point $p$ in *points* **do**
3 $q_x \leftarrow p_x \times (size / B_{width})$
4 $q_y \leftarrow p_y \times (size / B_{height})$
5 APPEND(*newPoints*, $q$)
6 **return** *newPoints*

TRANSLATE-TO-ORIGIN(*points*)
1 $c \leftarrow$ CENTROID(*points*)
2 **foreach** point $p$ in *points* **do**
3 $q_x \leftarrow p_x - c_x$
4 $q_y \leftarrow p_y - c_y$
5 APPEND(*newPoints*, $q$)
6 **return** *newPoints*

# 4E ÉTAPE : RECONNAISSANCE

- Un geste candidat *C* est comparé à chaque templates $T_i$
  - Calcul de la distance moyenne $d_i$ entre les points

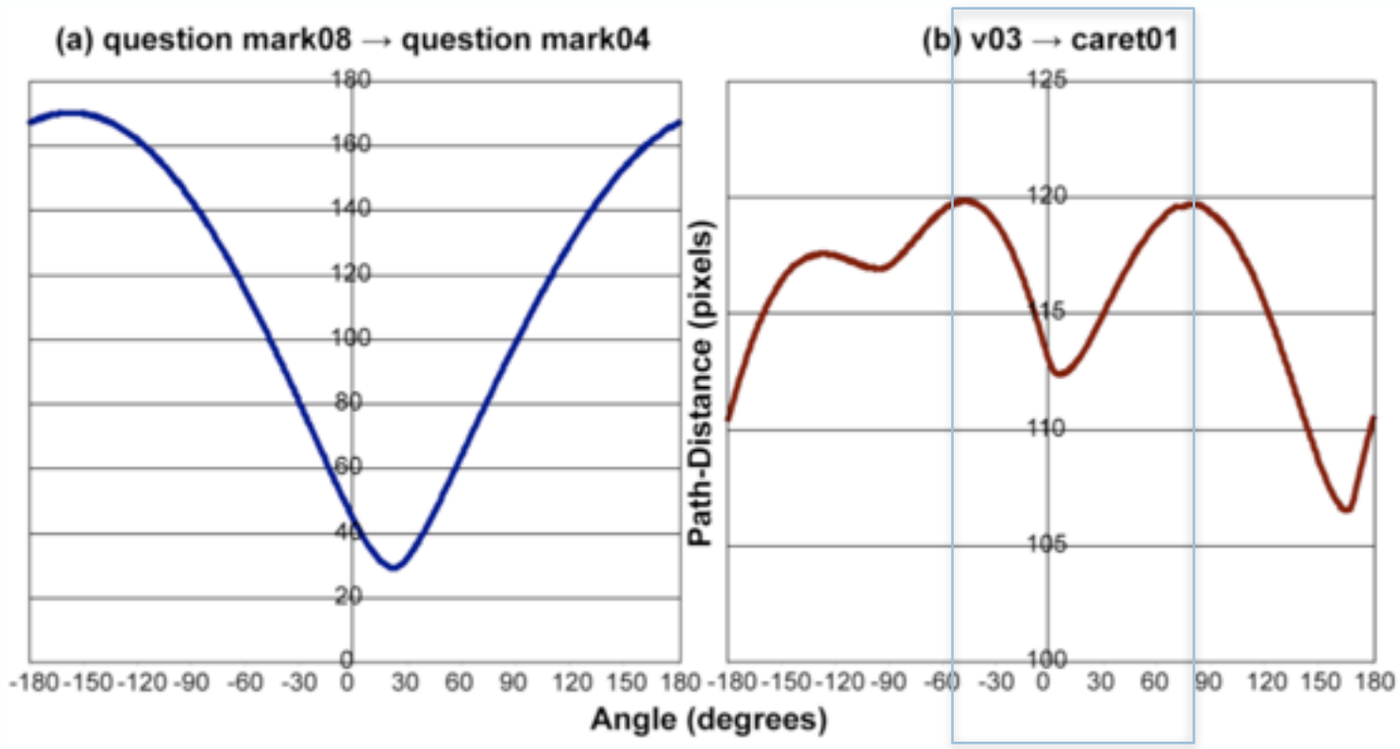
$$d_i = \frac{\sum_{k=1}^{N} \sqrt{\left(C[k]_x - T_i[k]_x\right)^2 + \left(C[k]_y - T_i[k]_y\right)^2}}{N}$$

- Le template avec le $d_i$ plus faible est le résultat
- La distance est transformée en score entre 0 et 1

$$score = 1 - \frac{d_i^*}{\frac{1}{2}\sqrt{size^2 + size^2}}$$

# 4E ÉTAPE : RECONNAISSANCE

- L'angle indicatif ne garantit pas que le geste candidat C sera parfaitement aligné avec un template
- On cherche à ajuster l'angle de rotation de C pour minimiser la distance entre C et Ti

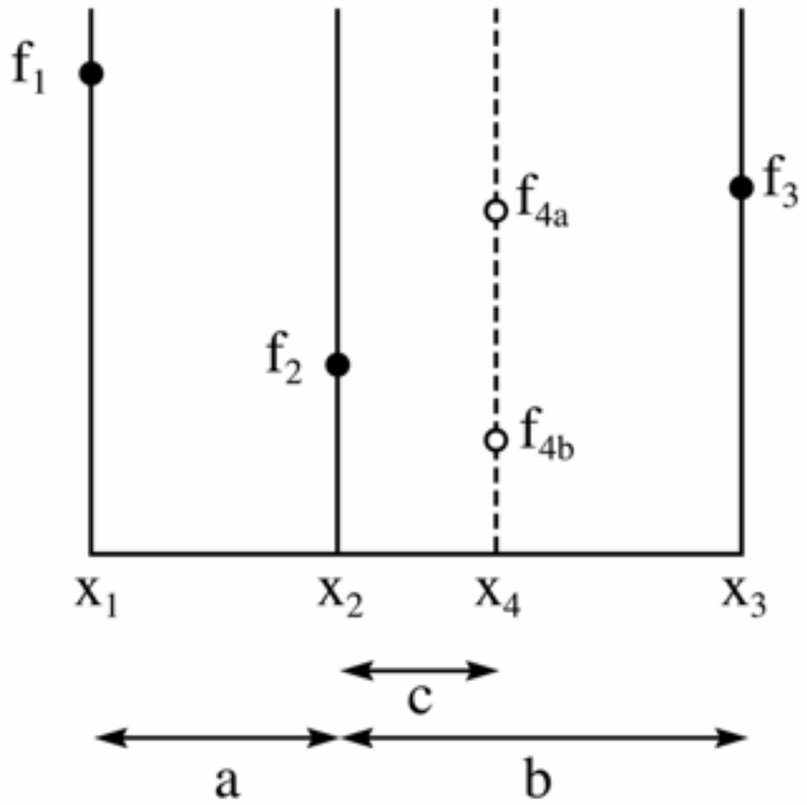


# 4E ÉTAPE : RECONNAISSANCE

- Golden Section Search :
  - Recherche d'une valeur minimale pour une fonction unimodale
- Similaire à la recherche dichotomique

$$a + c = b$$
$$c/a = a/b$$

$$\varphi = b/a$$
$$\varphi^2 - \varphi - 1 = 0$$
$$\varphi_1 = (1 + \sqrt{5})/2$$
$$\varphi_2 = (-1 + \sqrt{5})/2$$




$$x_2 = (1 - \varphi)x_1 + \varphi x_4$$
$$x_3 = \varphi x_1 + (1 - \varphi)x_4$$

# 4E ÉTAPE : RECONNAISSANCE

**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 $\varphi$ 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, θΔ)
4       if d < b then
5           b ← d
6           T' ← T
7   score ← 1 – b / 0.5√(size² + size²)
8   return ⟨T', score⟩
```

# 4E ÉTAPE : RECONNAISSANCE

DISTANCE-AT-BEST-ANGLE(*points*, $T$, $\theta_a$, $\theta_b$, $\theta_\Delta$)

```
1   x1 ← φθa + (1 − φ)θb
2   f1 ← DISTANCE-AT-ANGLE(points, T, x1)
3   x2 ← (1 − φ)θa + φθb
4   f2 ← DISTANCE-AT-ANGLE(points, T, x2)
5   while |θb − θa| > θΔ do
6       if f1 < f2 then
7           θb ← x2
8           x2 ← x1
9           f2 ← f1
10          x1 ← φθa + (1 − φ)θb
11          f1 ← DISTANCE-AT-ANGLE(points, T, x1)
12      else
13          θa ← x1
14          x1 ← x2
15          f1 ← f2
16          x2 ← (1 − φ)θa + φθb
17          f2 ← DISTANCE-AT-ANGLE(points, T, x2)
18  return MIN(f1, f2)
```

# 4E ÉTAPE : RECONNAISSANCE

```
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|
```

# LIMITATIONS

- Pas possible de distinguer un carré d'un rectangle
- Pas possible de distinguer une ellipse d'un cercle
- Pas possible de distinguer une flèche vers le bas/haut
- Pas possible de reconnaître des gestes « 1D »

# CLASSIFIEUR STATISTIQUE

- Rubine
- Reconnaissance de gestes à un seul tracé (unistroke)
- Traitement statistique de caractéristiques

*Dean Rubine*
*Specifying gestures by example*
*SIGGRAPH '91, 329-337*

# RUBINE

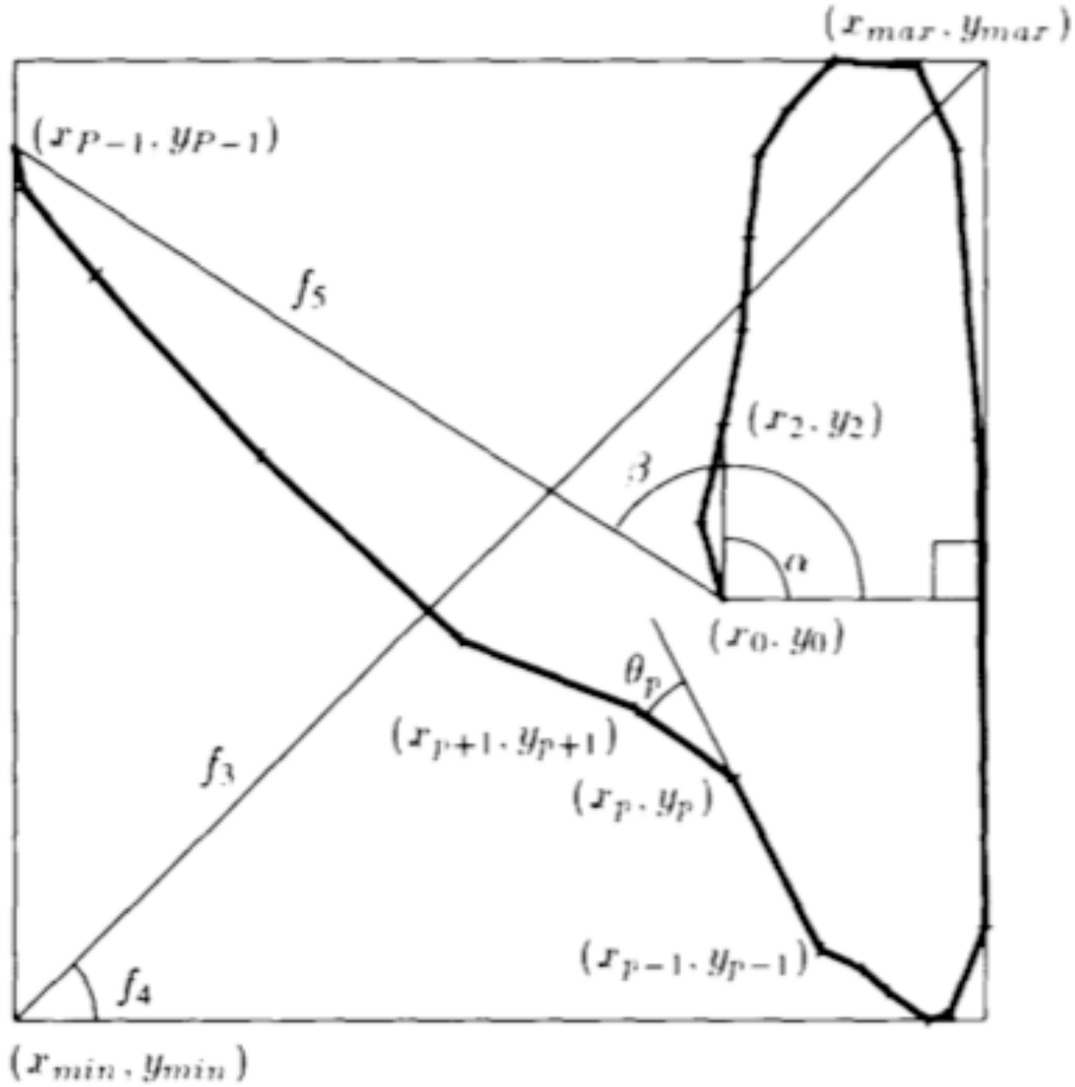
- Liste de points en entrée
  - Élimination des points trop proches.

    d<3 pixels du point précédent
- Calcul d'un vecteur de caractéristiques statistiques.
- Comparaison aux gestes de référence.

  Le geste avec le score le plus grand est renvoyé
- Détection des gestes ambigus

# CARACTÉRISTIQUES

1. cos de l'angle d'origine
2. sin de l'angle d'origine
3. longueur diagonale bounding box
4. angle diagonale bounding box
5. distance entre premier et dernier point
6. cos de l'angle entre le premier et dernier point
7. sin de l'angle entre le premier et dernier point
8. longueur totale du tracé
9. angle total traversé
10. somme des angles absolus à chaque point
11. somme des carrés de ces angles
12. vitesse$^2$ maximum du geste
13. durée du geste



# INITIALISATION

- Calcul des statistiques moyennes pour chaque caractéristique de chaque geste
- Calcul de la matrice de covariance moyenne de tous les gestes
- Objectif : trouver les coefficients pondérateurs des caractéristiques statistiques qui permettent de maximiser le score des gestes de chaque classe

# RECONNAISSANCE

- Calcul des caractéristiques du geste candidat
- Calcul d'une probabilité de correspondance pour chaque geste de référence
- Le geste avec la probabilité la plus importante est choisi

# DYNAMIC TIME WARPING

- DTW
- Déformation temporelle dynamique
- Déterminer pour chaque élément d'une séquence, le meilleur élément correspondant dans l'autre séquence relativement à un certain voisinage et à une certaine métrique
- Complexité polynomiale

# APPLICATIONS

- Vidéo, audio, graphique...
- Toutes données qui peuvent être transformées en représentation linéaire en fonction du temps (séries temporelles)
- Echantillons ordonnés par une étiquette de temps
- Reconnaissance vocale
- Reconnaissance de gestes off-line et on-line
- Alignement de protéines...

# PRINCIPE DE BASE

- Séquence de référence $R = [r_1, r_2, ..., r_n]$
- Séquence de test $T = [t_1, t_2, ..., t_m]$
- Si m = n alors on peut calculer la distance deux à deux:

$$D = \sum_{i=1}^{n} distance(r_i, t_i)$$

- Possibilité de calculer la distance euclidienne ou d'utiliser une autre métrique (fonction qui donne un réel)
- Plus possible d'utiliser cette méthode dès que n ≠ m
- Attention : les échantillons doivent être équidistants en temps

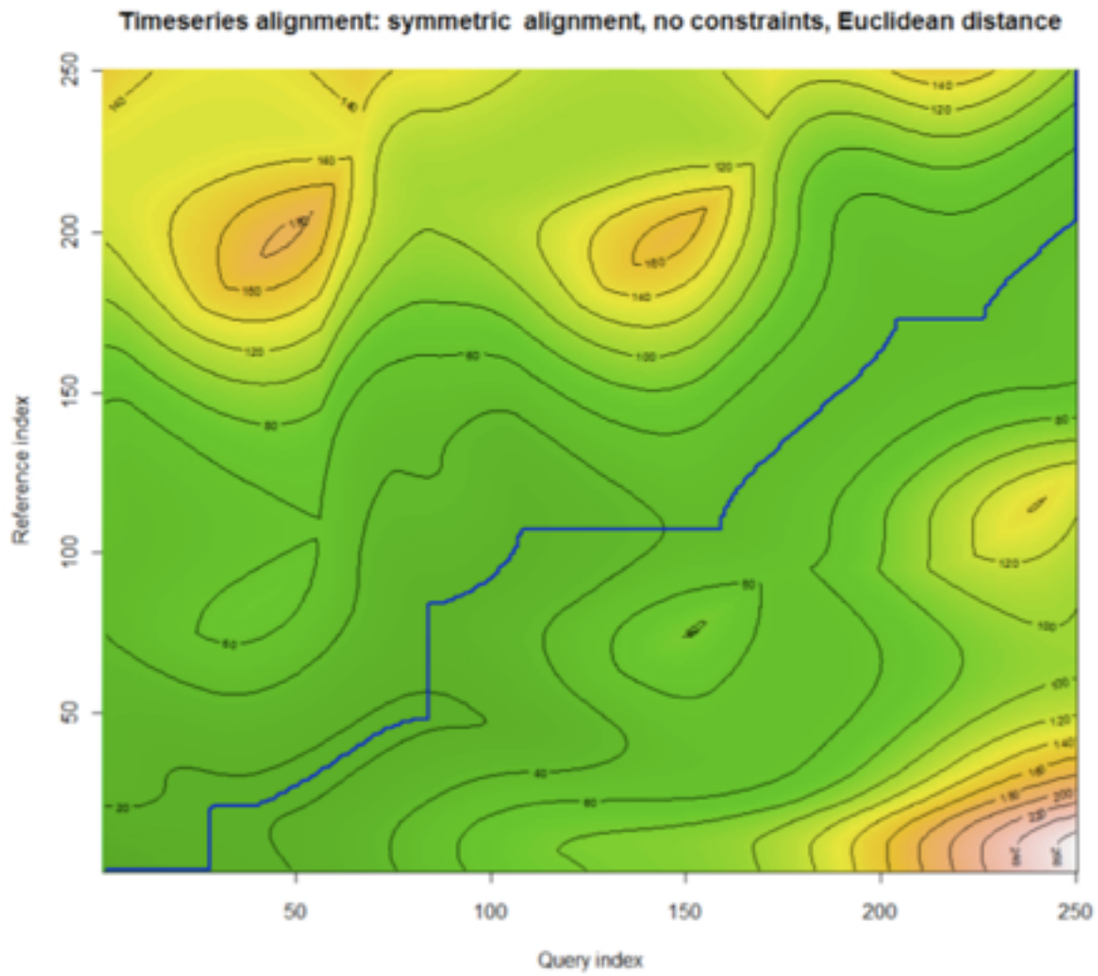
# PRINCIPE DE BASE

- DTW réalise d'abord un alignement non linéaire en recherchant parmi tous les alignements possibles, celui qui minimise une fonction de coût cumulé
- « Time Warping » : Dilation ou compression des séquence pour obtenir le meilleur alignement possible

# PRINCIPE DE BASE

- Calcul du chemin $W = [w_1, w_2, ..., w_k]$ de longueur minimale

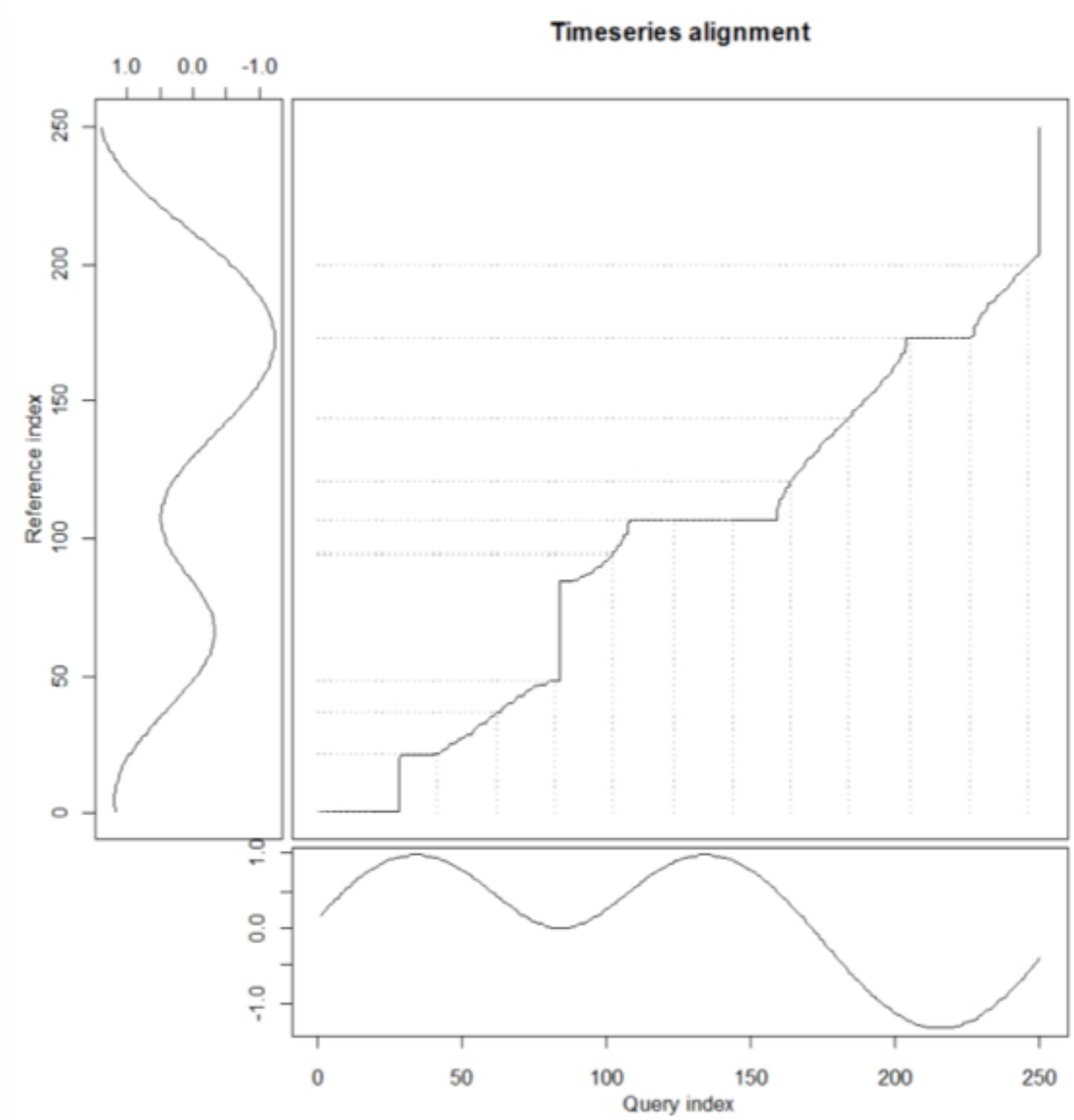
$$\sum_{i=1}^{k} distance(w_i)$$



# PRINCIPE DE BASE

- Conditions aux frontières
  - $w_l = (r_l, t_l)$
  - $w_k = (r_n, t_m)$
- Contraintes locales
  - Monotonicité pour respecter le séquencement des points
  - Eviter les sauts dans le temps
  - Pour tout couple $(r_i, t_j)$, le choix des prédécesseurs est limité à $(r_{i-1},t_j)$, $(r_i,t_{j-1})$, $(r_{i-1},t_{j-1})$
- Exhaustivité
  - Chaque élément de R doit être mis en relation avec au moins un élément de T et vice-versa
  - $\max(m, n) \leq k \leq m + n - 1$

# PRINCIPE DE BASE



# PRINCIPE DE BASE

- Programmation dynamique
- Fonction d'optimisation :
  - Soit D(i, j) la longueur du chemin entre $(r_1, t_1)$ et $(r_i, t_j)$
- Récursion:
  - $D(i, j) = dist(r_i, t_j) + min(D(i-1, j), D(i-1, j-1), D(i, j-1))$
  - Condition initiale : $D(1,1) = dist(r_0, t_0)$
- Distance minimale entre les deux séquences
  - D(n, m)

# MISE EN APPLICATION

- Construction d'une matrice D de dimensions n x m
- Remplissage de D(1,1) avec la condition initiale
- Remplir récursivement ligne par ligne ou colonne par colonne
- Cas particuliers première ligne et première colonne
- Distance minimale : D(n,m)
- Distance minimale peut être normalisée par la longueur du chemin

# ALGORITHME

$\mathrm{n} \leftarrow |X|$

$\mathrm{m} \leftarrow |Y|$

$dtw[] \leftarrow new\ [n \times m]$

$dtw(0,0) \leftarrow 0$

**for** $i = 1;\ i \leq n;\ j++$ **do**

$dtw(i,1) \leftarrow dtw(i-1,1) + c(i,1)$

**end for**

**for** $j = 1;\ j \leq m;\ j++$ **do**

$dtw(1,j) \leftarrow dtw(1,j-1) + c(1,j)$

**end for**

**for** $i = 1; i \leq n; j++$ **do**

**for** $j = 1; j \leq m; j++$ **do**

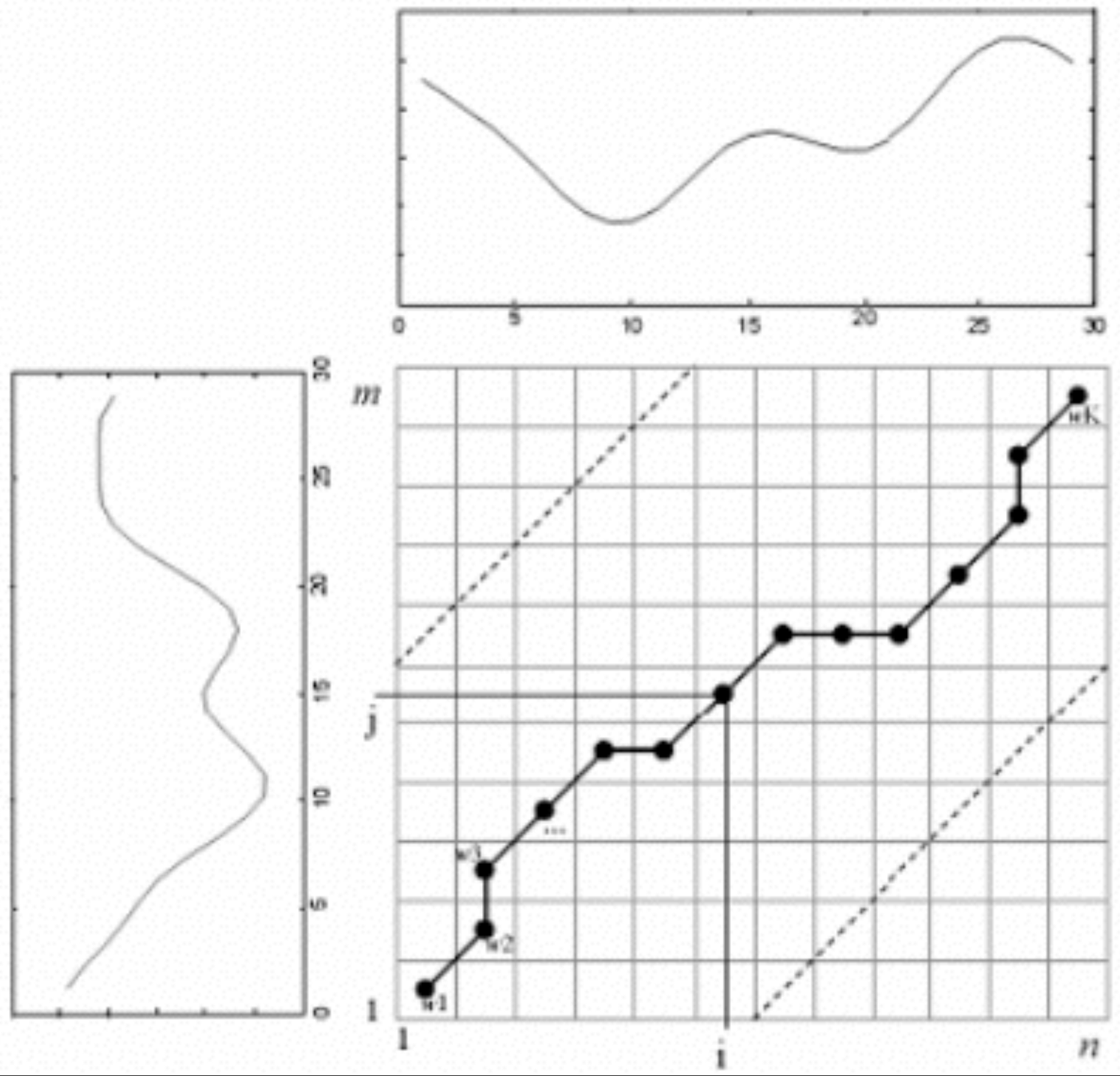
$dtw(i,j) \leftarrow c(i,j) + \min\{dtw(i-1,j);\ dtw(i,j-1);\ dtw(i-1,j-1)\}$

**end for**

**end for**

**return** $dtw$

# CHEMIN DE DÉFORMATION



# CHEMIN DE DÉFORMATION

- A chaque calcul de D(i, j) :

  Sauvegarde du prédécesseur qui minimise la distance

- Parcours des prédécesseurs en partant de D(n, m)

# COMPLEXITÉ

- Complexité : O(m*n)
- Optimisation en limitant la région de recherche

# RECONNAISSANCE DE GESTES

- Le geste fait généralement l'objet de variance dans sa reproduction
- Un seul geste exemple n'est souvent pas suffisant
- Définition d'un modèle statistique tenant compte de ces variations
- Utilisation de plusieurs gestes exemples

  ➡ Calcul de la moyenne et de la variance en fonction du temps

# RECONNAISSANCE DE GESTES

- Chaque exemple est comparé à l'exemple le plus long avec DTW, avant de calculer les caractéristiques statistiques
- Calcul d'un modèle de geste g pour chaque classe d'exemples
  - Calcul de la moyenne et de la variance à chaque pas de temps :

$$\tilde{g}_m[t] \; et \; \sigma^2(g_m[t])$$

# RECONNAISSANCE DE GESTES

- Un geste candidat est aligné avec chaque modèle en utilisant DTW
- A chaque nouvel événement, la séquence de test T est comparée à chaque modèle en utilisant la métrique suivante :

$$D_{i,j} = \sum \frac{1}{\sigma^2(g_m[t])}(\tilde{g}_m[t] - T[j])^2$$

# RECONNAISSANCE DE GESTES

- Alignement de la séquence T avec les modèles

  Retour dans le temps, présent = 0

- Relaxation des contraintes

  Longueur minimale pour la séquence T

- Le score du geste est l'inverse de la distance calculée par DTW
- Le geste avec le score le plus élevé est le geste reconnu