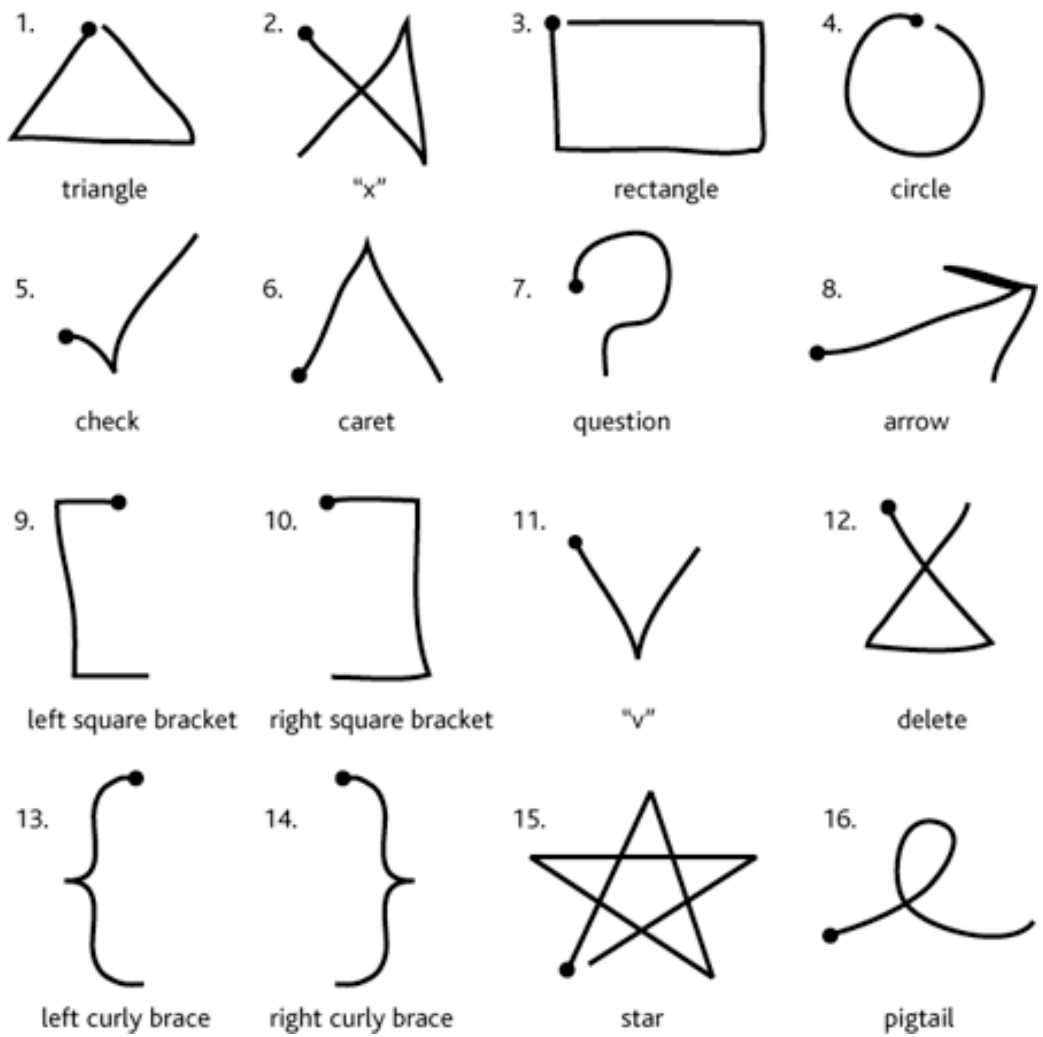
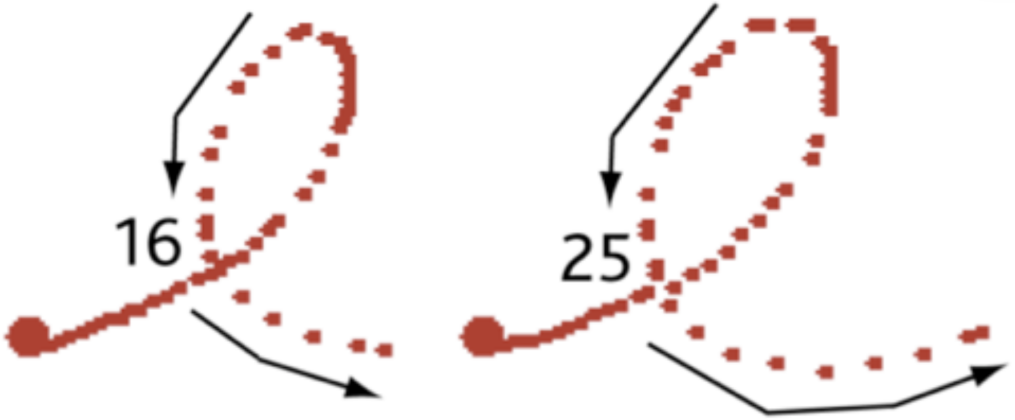
$1 recognizer



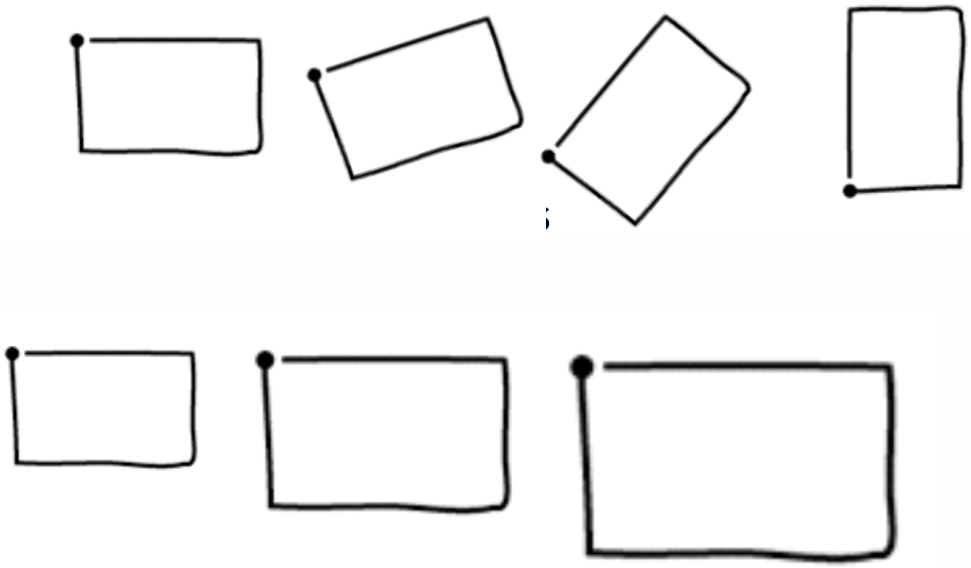
Problèmes posés

Number of points

- Speed & Frequency
- …



Rotation, Translation, Scale



4 étapes

A. Ré-échantillonner le geste
- ➡ Invariant à la fréquence d'acquisition
- ➡ Invariant à la vitesse d'exécution

B. Ré-orientation du geste
- ➡ Invariant à l'orientation

C. Mise à l'échelle et translation
- ➡ Invariant à l'échelle
- ➡ Invariant à la position

D. Reconnaissance du geste

1ère étape : Ré-échantillonnage

Le geste est défini par M points ordonnés.

On veut N = 64 points ordonnés équidistants les uns des autres.

B. Calcul de la distance I entre 2 points :

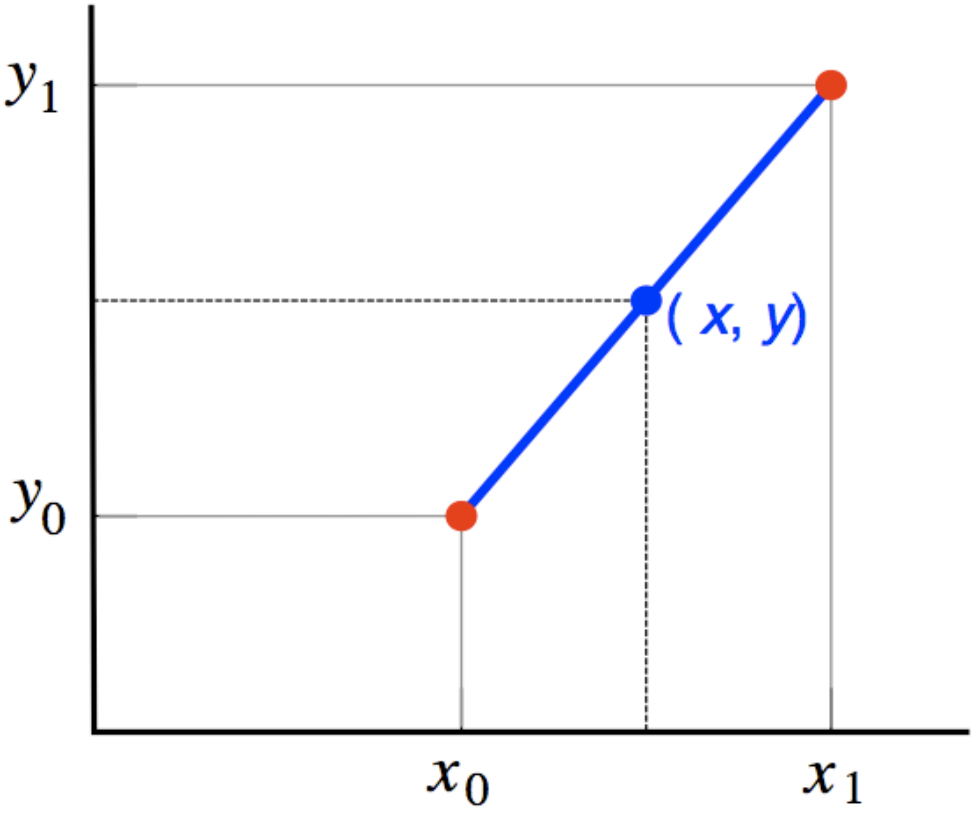
- Calcul de la longueur totale du geste.
- I = longueur / (N-1)

C. 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

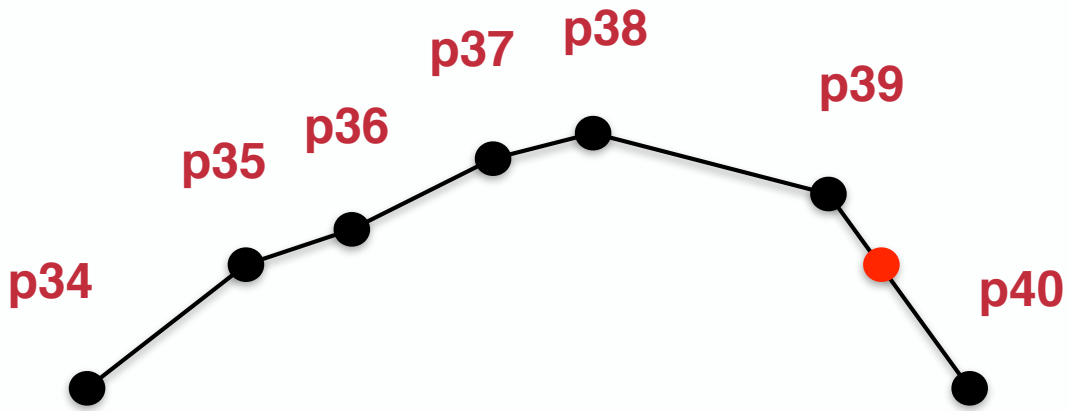
Interpolation linéaire



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

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

1ère étape : Ré-échantillonnage



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

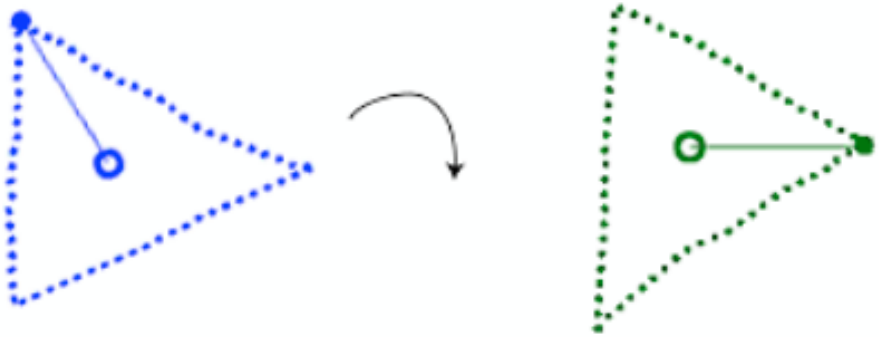
2e étape : Rotation « indicative »

A. Calcul du centre du geste (centroïde)

B. Calcul de l'angle entre :

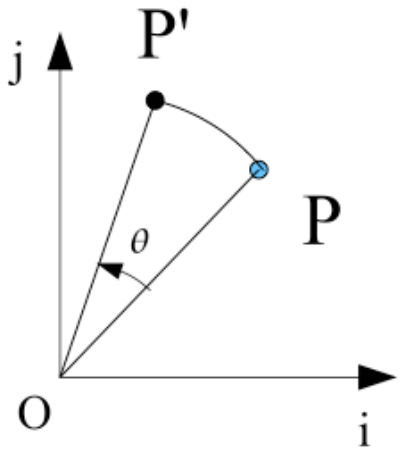
- Le centroïde,
- Le premier point
- L'horizontale

C. 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 α 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 $y = r\sin(\alpha)$

3e étape : mise à l'échelle et translation

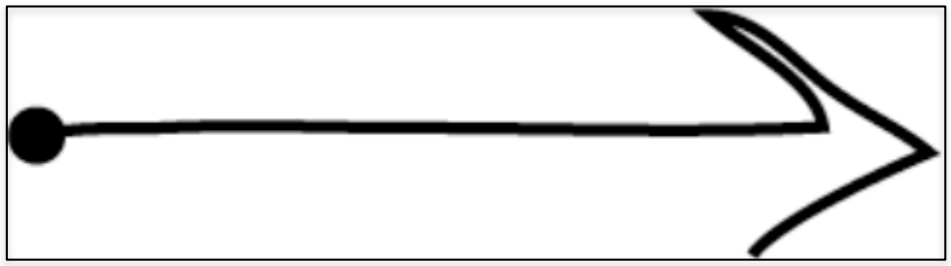
Mise à l'échelle non uniforme: on ramène le geste à un carré de référence

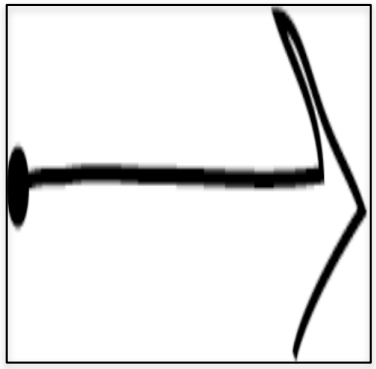
A. Calcul de la bounding box

- Calcul de min_x, max_x, min_y, max_y

B. Mise à l'échelle

C. Translation à l'origine





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 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, 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

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

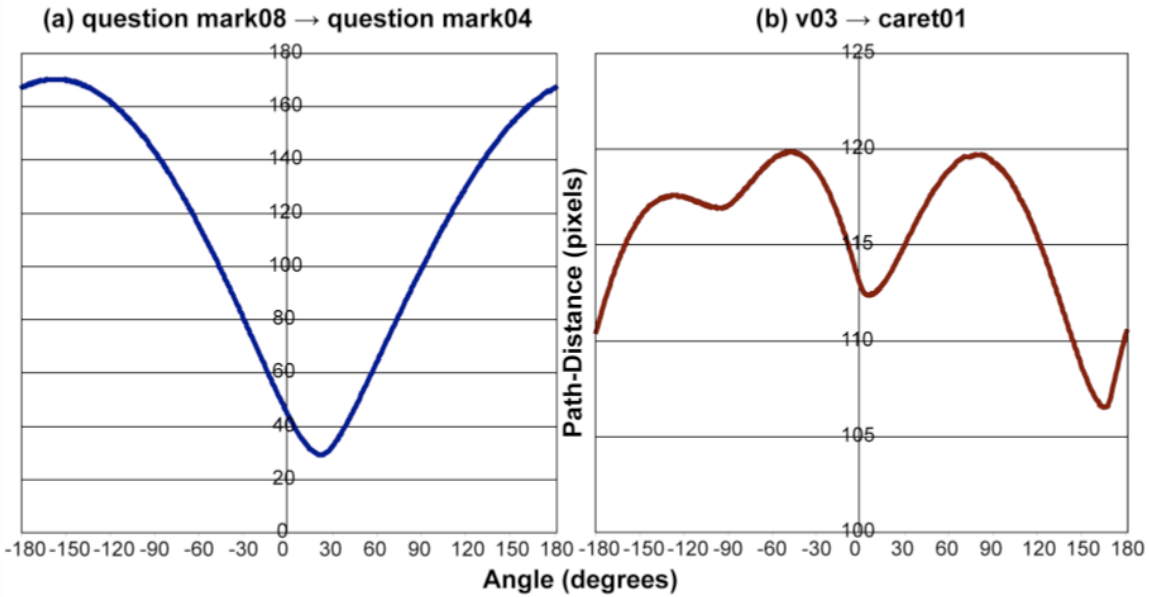
La distance est transformée en score entre 0 et 1

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

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

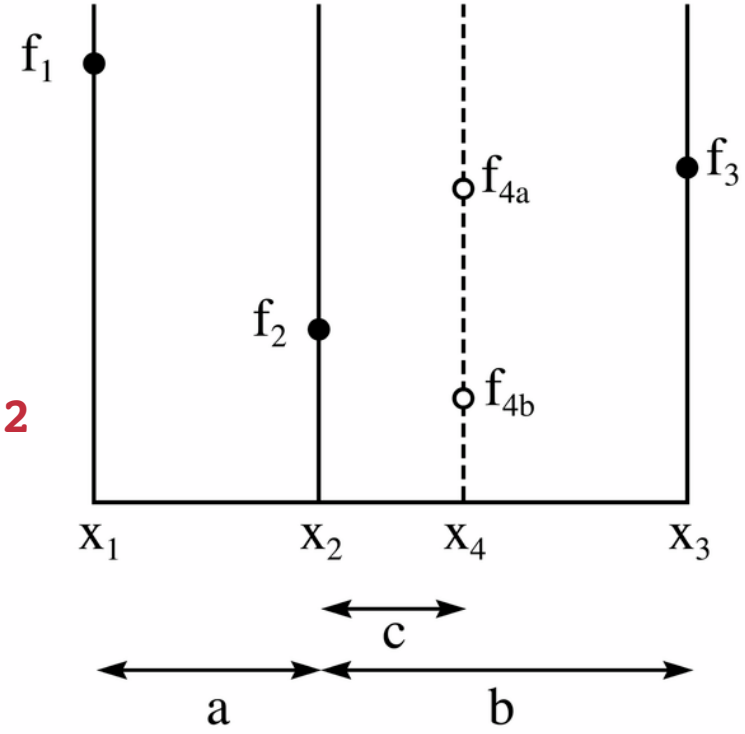
Golden Section Search :

- 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

```
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

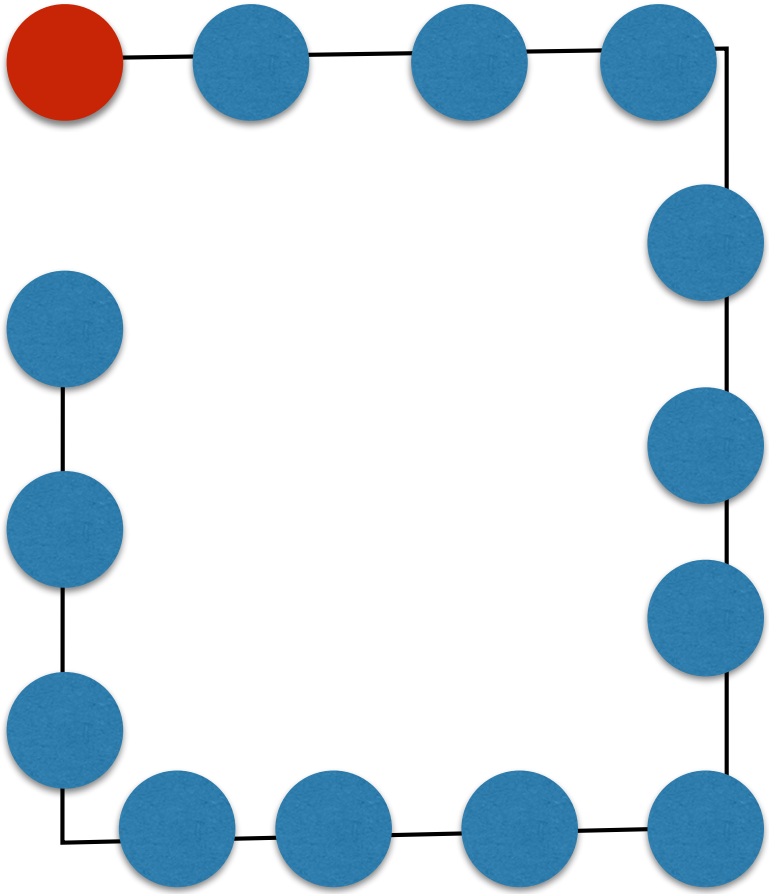
Pas possible de distinguer une flèche vers le bas/haut

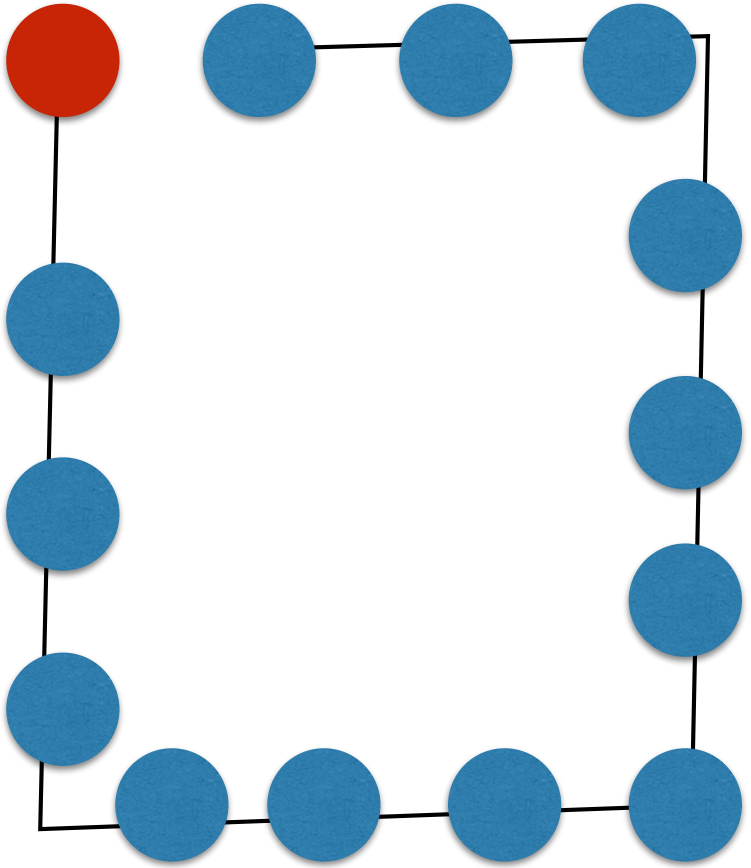
Pas possible de reconnaître des gestes « 1D »

Principles:

1. be **resilient** to variations in sampling
2. rotation, scale, and position **invariance** ⟵ != Rubine
3. require **no** advanced **mathematical** techniques
4. be easily written (**~100 line of codes**)
5. **fast** (interactivity)
6. require only **one example** ⟵ > Rubine
7. independent of the number of input points;
8. **Accurate**

Limitations?



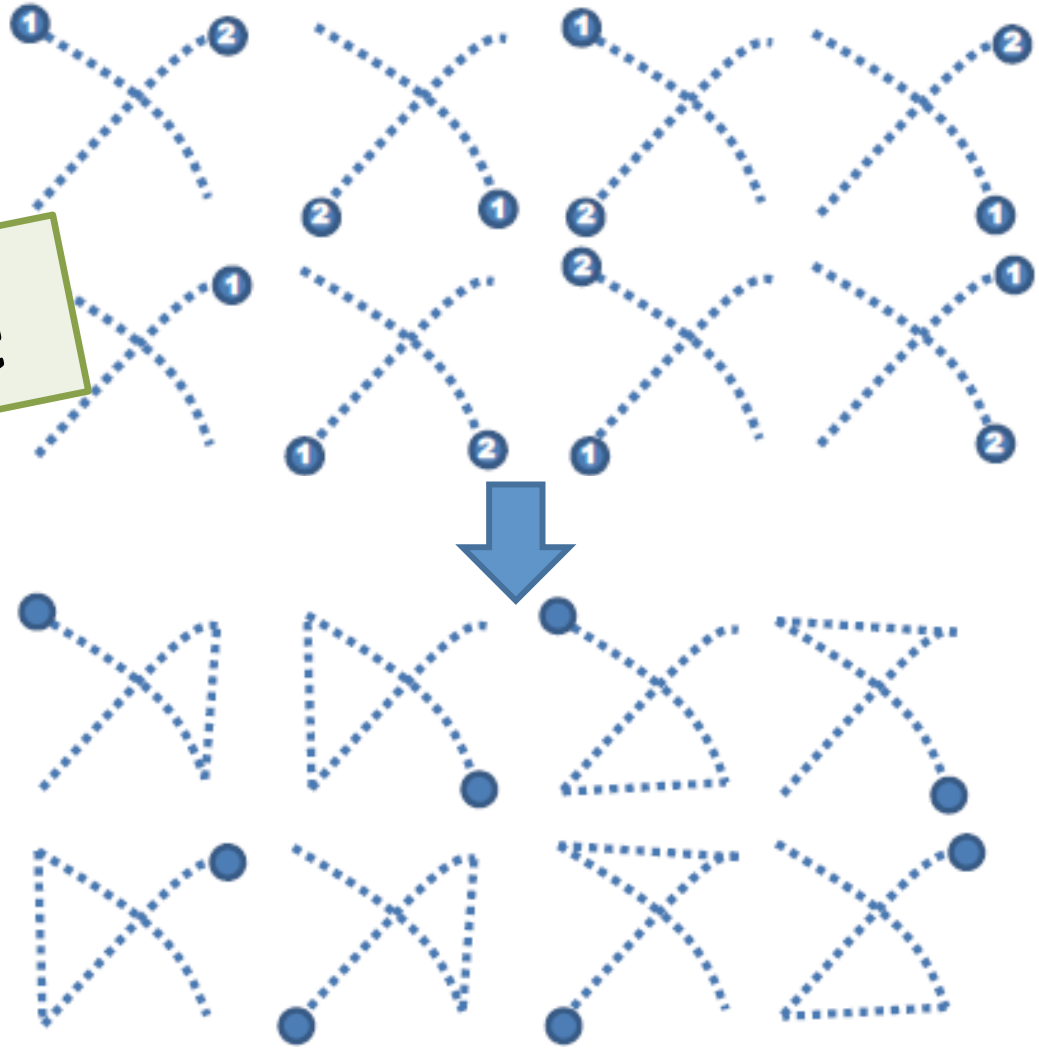


The $N recognizer

Multistrokes

~200 lines of code

1-to-1 point matching
Uses $1 and stroke order permutations.



Anthony, L., Wobbrock, J.O.
A lightweight multistroke recognizer for user interface prototypes
In Proc. of Graphics Interface 2010, 245-252, 2010