Interaction Gestuelle : Rubine & $1/$N Recognizer

Objectifs

L'objectif du TP est de comparer un moteur deux systèmes de reconnaissance de gestes : l'algorithme de Rubine d'une part et le $1 Recognizer.

Nous détaillerons tout d'abord l'algorithme du « *One Dollar Recognizer* » (cf. plus bas pour le lien).

L'objectif est d'ensuite de coder une extension du $1 Recognizer vers le $N Multistroke Recognizer).

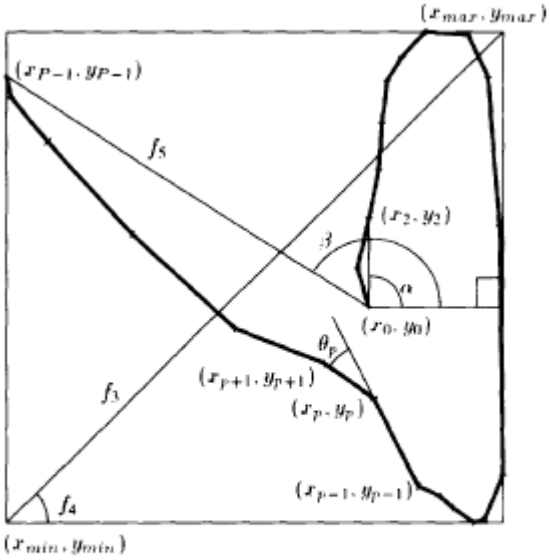
Documents de référence utilisés

- https://dl.acm.org/doi/pdf/10.1145/127719.122753
  Publication : Rubine D. (1991), Specifying Gestures by Example in Computer Graphics, volume 25, Number 4
- http://depts.washington.edu/aimgroup/proj/dollar/
  Explication du projet, exemple de « *One Dollar Recognizer* » à tester, des exemples de code
- http://faculty.washington.edu/wobbrock/pubs/uist-07.01.pdf
  Publication : Wobbrock, J.O., Wilson, A.D. and Li, Y. (2007). Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '07). Newport, Rhode Island (October 7-10, 2007). New York: ACM Press, pp. 159-168.
- http://faculty.washington.edu/wobbrock/pubs/gi-10.02.pdf
  Publication : Anthony L., Wobbrock J.O (2010). A Lightweight Multistroke Recognizer for User Interface Prototypes. Proceedings of Graphics Interface 2010. Ottawa, Ontario (May 31-June 2, 2010), pp. 245-252

L'implémentation d'un moteur de reconnaissance de gestes nécessite normalement des connaissances avancées en apprentissage. Pour un concepteur en IHM, il est donc difficile d'implémenter la reconnaissance de gestes spécifiques. Nous allons présenter 2 systèmes de reconnaissances de gestes simples à mettre en œuvre.

Algorithme de Rubine

L'algorithme développé par Dean Rubine repose sur le calcul de 13 fonctions mathématiques (cf. Figure 1) permettant de déterminer quel geste a été reproduit parmi ceux déjà appris. Il repose donc sur un apprentissage préalable de modèles. Dans cette approche, il y a toujours une solution (la « plus proche ») aussi est-il important de seuiller le résultat afin de déterminer si le résultat est acceptable ou pas.



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

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

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

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

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

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

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

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

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

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

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

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 1 : Fonctions utilisées pour la reconnaissance

Le One Dollar Recognizer

L'objectif du projet « *One Dollar Recognizer* » est de permettre le codage d'un moteur de reconnaissance de gestes dans différents langages, sans que le développeur ait besoin d'avoir des connaissances dans l'apprentissage. Il a été démontré que l'algorithme détecte les gestes avec un taux proche des algorithmes plus complexes. La Figure montre les gestes (non ambigus) reconnus par l'algorithme. Vous pouvez tester le moteur de reconnaissance sur le site proposé par le projet (voir plus haut).

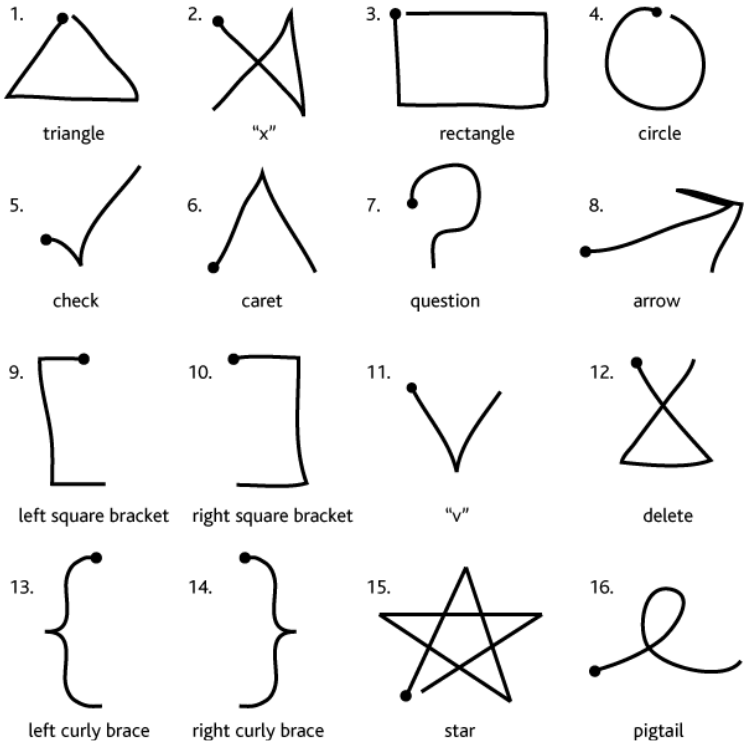


Figure 2 : Gestes reconnus par le One Dollar Recognizer

L'article décrit quant à lui les principes de l'algorithme.
En résumé, le geste de l'utilisateur est une trace de « **Points Candidats** » pour lesquels il faut trouver le modèle (« **Template** ») qui ressemblent le plus à cette trame. Dans la suite de cet énoncé on parlera de modèles (= *template*) et trace (= *candidate*).

L'algorithme est indépendant de la façon comment le geste est effectué (avec la souris, sur une surface tactile, …) mais le hardware et les spécificités de l'utilisateur déterminent la résolution des points, etc. Par conséquent, il faut compenser ces différences. L'algorithme procède en plusieurs étapes (décrit sur pages 3 à 5 de l'article). Les étapes 1 à 3 s'appliquent aux modèles ainsi qu'aux traces :

1. « **Resample** » de la trace : le nombre de points par trace dépend du hardware, logiciel et de la vitesse de l'utilisateur à la saisie. Dans une première étape on fait un ré-échantillonnage pour que les points sont équidistants sur la trace et correspondent au nombre de points dans le modèle. Un nombre de points N entre 32 et 256 aurait donné des bons résultats.

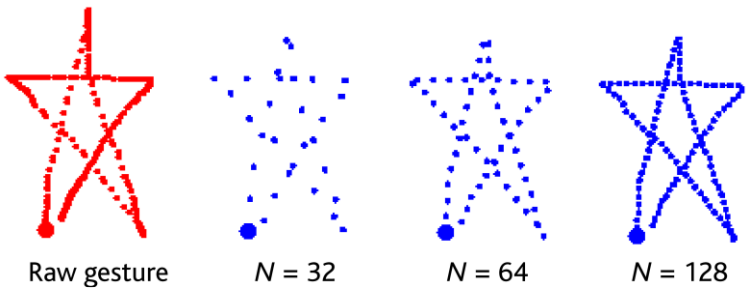


Figure 3 : Exemples de ré-échantillonnage

2. **Rotation** : tourner la trace pour que l'angle entre le centre et le premier point se trouve à 0° (à droite, voir Figure).

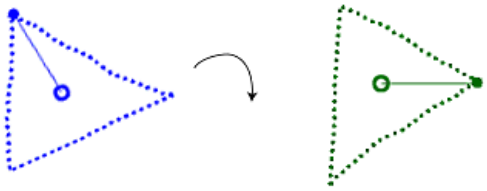


Figure 4 : Rotation de la trace

3. **Redimensionner** et **décaler** : la trace est ensuite redimensionnée à la taille d'un carré de référence (les proportions de la trace originale ne sont donc pas respectées). Ensuite le centre de la trace est placé à la coordonnée (0,0).

4. **Trouver** le meilleur angle pour obtenir le meilleur score : on calcule la distance moyenne entre la trace et les différents modèles avec la formule suivante :

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

Le modèle ayant la plus petite distance est le modèle retenu. Il est de plus possible de convertir cette distance dans un score de reconnaissance (voir article page 4). Il est aussi éventuellement nécessaire de faire une rotation plus précise (voir page 5 de l'article).
L'article fournit enfin en annexe (page 10) le pseudo-code de l'algorithme.

Les grandes étapes à suivre

1. Récupérer le code ICAR (https://github.com/truillet/icar) et OneDollarIvy (https://github.com/truillet/OneDollarIvy), les tester.

2. Comprendre les différences et en déterminer les avantages et les inconvénients de chaque approche

3. Intégrer le ou les systèmes pour manipuler des formes géométriques sur un écran (vous pouvez utiliser le code Processing disponible ici : https://github.com/truillet/upssitech/blob/master/SRI/3A/IHM/TP/Code/Palette.zip

4. A partir de l'exemple fourni par $**1 Recognizer,** programmer l'algorithme **$N Multistroke Recognizer** (pseudo-code ici : http://depts.washington.edu/acelab/proj/dollar/ndollar.pdf) dans un langage de votre choix.

Il n'y a pas de rendu attendu mais vous pouvez transmettre vos notes prises durant le TP.