

## Projet de Fin d'Etudes

# RECONNAISSANCE DE GESTES GRAPHIQUES



## Sommaire

---

<b>Sommaire .....</b>	<b>3</b>
<b>Introduction .....</b>	<b>4</b>
<b>I. La reconnaissance de l'écrit .....</b>	<b>7</b>
1. La reconnaissance de l'écriture .....	7
2. La reconnaissance de gestes graphiques .....	10
3. Conclusion.....	14
<b>II. Méthode de Rubine .....</b>	<b>16</b>
1. Description de la méthode.....	16
2. Caractéristiques de Rubine .....	16
3. Classification des gestes.....	19
4. Apprentissage.....	19
5. Rejet .....	21
<b>III. Application logicielle .....</b>	<b>23</b>
1. Présentation du logiciel .....	23
2. Système d'apprentissage et de reconnaissance de gestes.....	26
3. Editeur d'organigrammes .....	38
<b>Conclusion .....</b>	<b>58</b>
<b>Table des matières .....</b>	<b>59</b>
<b>BIBLIOGRAPHIE .....</b>	<b>62</b>
<b>ANNEXES .....</b>	<b>64</b>

## Introduction

Depuis le début des années 80, les interfaces des ordinateurs évoluent en même temps que le nombre d'utilisateurs d'outils informatiques. Elles se veulent de plus en plus conviviales, de plus en plus agréables et faciles à utiliser. L'apparition de nouveaux médias et de nouveaux moyens de communication favorise d'autant plus cette évolution.

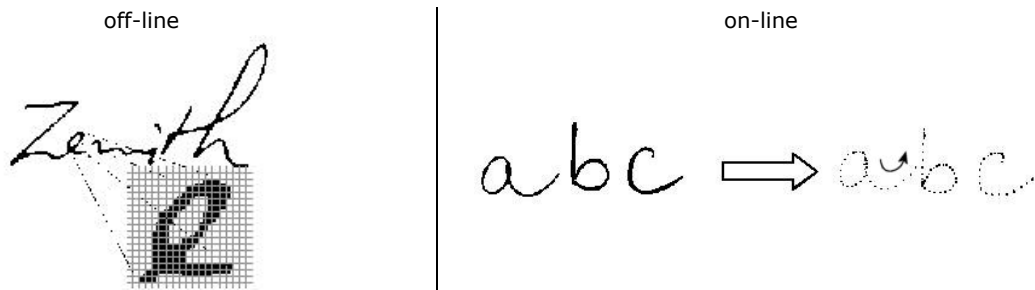
Nous nous intéresserons tout particulièrement aux systèmes digitaux à base de stylo : PDA, ordinateurs de poche, tablettes graphiques...



L'arrivée sur le marché de ces nouveaux dispositifs ouvre de nouvelles perspectives pour les protocoles d'interaction homme-machine. Ils permettent de recréer la situation papier-stylo où l'utilisateur exprime rapidement ses idées, sous forme visuelle. L'interaction directe au stylo évite l'apprentissage de langages ou de logiciels graphiques.

Grâce à ces technologies de pointe, on peut s'intéresser à l'aspect dynamique du document lors de son traitement. La reconnaissance manuscrite en ligne (on-line) sous-entend que le système analyse le document pendant son élaboration : il doit être capable de répondre en temps réel aux actions de l'utilisateur, alors que le off-line (traitement du document après sa numérisation) néglige cette donnée temporelle, la vitesse étant alors un critère de performance et non de qualité.

La reconnaissance on-line utilise en général les représentations vectorielles et non la reconnaissance d'images avec extraction de contours ou autres. Les données correspondent donc à une liste de points ordonnés. D'autres critères sont également pris en compte comme par exemple, la pression et l'inclinaison du stylet, sa vitesse ou son accélération...



Jusqu'à récemment, la reconnaissance de caractères imprimés était beaucoup plus étudiée que la reconnaissance on-line. Mais, avec cette avancée étonnante des systèmes informatiques à base de stylo, la reconnaissance du manuscrit devient un domaine très important, de plus en plus développé.

Les domaines d'application sont très variés :

- reconnaissance de l'écriture manuscrite ([9], [25]) ;
- reconnaissance de signatures ;
- annotation et la correction de documents ([1], [2], [3], [18]) ;
- création de tableaux ([11]) ;
- élaboration de diagrammes entité-relation ([16]) ;
- édition d'équations mathématiques ([6]) ;
- édition de musique ;
- contrôle aérien ([10]) ;
- ...

L'objectif du projet de fin d'études présenté dans ce rapport a été d'étudier les travaux existants sur la reconnaissance de l'écrit (chapitre I), afin de dégager une théorie intéressante pour traiter les gestes en ligne (chapitre II). On appliquera ensuite la méthode choisie pour créer notre propre système d'apprentissage et de reconnaissance de gestes graphiques (chapitre III).

# La reconnaissance de l'écrit

## I. La reconnaissance de l'écrit

---

L'idée de trouver une méthode qui reconnaisse l'écrit à 100% reste illusoire car l'être humain lui-même a parfois des difficultés à reconnaître sa propre écriture. Toutefois, beaucoup de chercheurs dans le monde travaillent sur des approches variées afin de résoudre au mieux ce problème de reconnaissance. Les méthodes stochastiques, qui requièrent beaucoup de données exemples pour l'apprentissage, sont utilisées dans la plupart des systèmes on-line. Ces techniques correspondent aux chaînes de Markov cachées, aux réseaux de neurones, à la logique floue combinées parfois à des méthodes statistiques ou géométriques.

Les paragraphes suivants traitent tout d'abord des études concernant la reconnaissance de l'écriture manuscrite, puis la reconnaissance de gestes. On verra que ces deux applications sont très liées.

### 1. La reconnaissance de l'écriture

#### 1.1. Ecriture cursive

De nombreux travaux sur la reconnaissance de l'écriture manuscrite ont été élaborés mettant généralement en œuvre des méthodes à base de chaînes de Markov cachées et/ou de réseaux de neurones.

##### **CHAINES DE MARKOV**

*Les chaînes de Markov sont tout particulièrement intéressantes pour la reconnaissance de la parole et parallèlement de l'écriture cursive, c'est-à-dire la reconnaissance des mots avec des lettres attachées. Le principe est basé sur un modèle stochastique avec une représentation graphique des caractères et des mots écrits à la main. Le tracé du crayon est converti en un graphe où les nœuds reflètent les changements de direction (utilisant le code de Freeman), incluant les informations temporelles. La chaîne de Markov est construite sur une matrice de probabilité de transition. La matrice markovienne est initialisée pour chaque mot du dictionnaire pour ce qui est de la phase d'apprentissage et, lors de la phase de reconnaissance, l'entrée est comparée à cette matrice. La reconnaissance de mots entiers facilite le problème de séparation, mais il est alors nécessaire d'avoir un large dictionnaire. D'un autre côté, avec une approche de pré-segmentation, des parties de mots peuvent être reconnus et finalement comparées aux entrées du dictionnaire. Cette approche est pratique car les utilisateurs écrivent souvent des mots en plusieurs parties.*

##### **RESEAUX DE NEURONES**

*Dans la reconnaissance de l'écriture, les valeurs choisies pour les entrées d'un réseau de neurones représentent un vecteur de caractéristiques extraites du tracé effectué. D'autre part, les nœuds de sortie correspondent à des caractères. L'activation de chaque sortie équivaut à la probabilité que le caractère associé au nœud concorde avec le geste dessiné. La technique du réseau de neurones demande une longue phase d'apprentissage et le réseau peut être sans cesse mis à jour.*

On peut citer par exemple, les applications françaises REMUS [9] et MyScript [25], qui utilisent le modèle neuro-markovien, mise en relation des modèles de Markov cachés et des réseaux de neurones.

Ces produits permettent non seulement de reconnaître tous styles d'écriture...



Reconnaissance de chaque lettre séparément

*handprint*

Deux possibilités :

- reconnaissance de chaque lettre séparément
- utilisation d'un dictionnaire

*cursive handwriting*

C'est le degré le plus complexe en reconnaissance de l'écriture :

- segmentation en lignes, en mots, puis en caractères
- utilisation d'un lexique

...dans différents langages mais également les dessins, les formes, les diagrammes et les tables.

On trouve également sur le marché les logiciels CalliGrapher [27] et PenReader [28] développés pour les ordinateurs de poche, ainsi que l'application PenOffice [27] utilisée avec n'importe quelle interface à base de stylo, sous Microsoft Windows.

### 1.2. Alphabet prédéfini

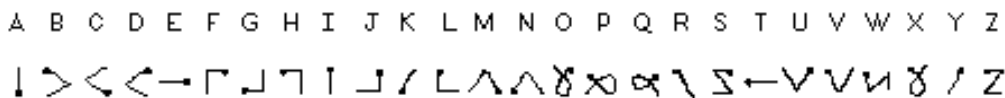
Le domaine de l'écriture est assez délicat puisque chacun possède un style d'écriture personnel. Une solution à ce problème est de forcer l'utilisateur à écrire selon un alphabet prédéfini comme nous l'explique Goldberg et Richardson [24]. L'alphabet exposé dans l'article en question contient des formes représentées en un seul coup de crayon.

Techniquement, on utilise cinq formes basiques...



... qui peuvent être tournées selon quatre orientations et tracées suivant deux directions, ce qui fait en totalité quarante possibilités.

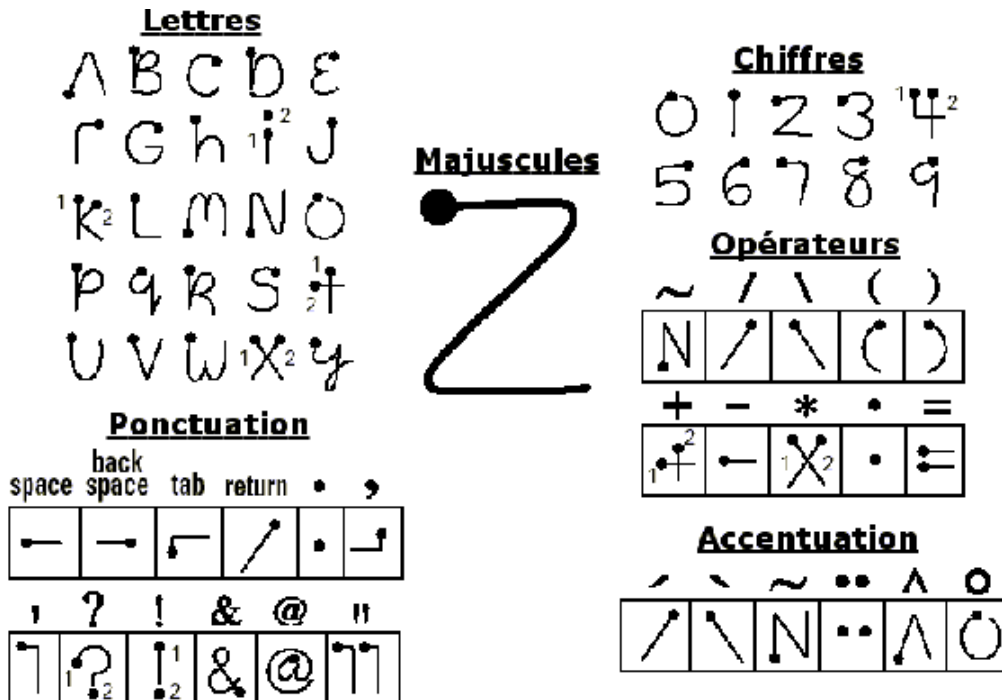
Parmi ces gestes, les chercheurs en ont choisi 26 pour symboliser l'alphabet :



On reconnaît un inconvénient majeur à cette méthode : l'ensemble des caractères possibles est insuffisant, limité aux vingt-six lettres de l'alphabet, sans aucune possibilité de représenter les lettres accentuées, ni les caractères spéciaux, ni les signes de ponctuation,...

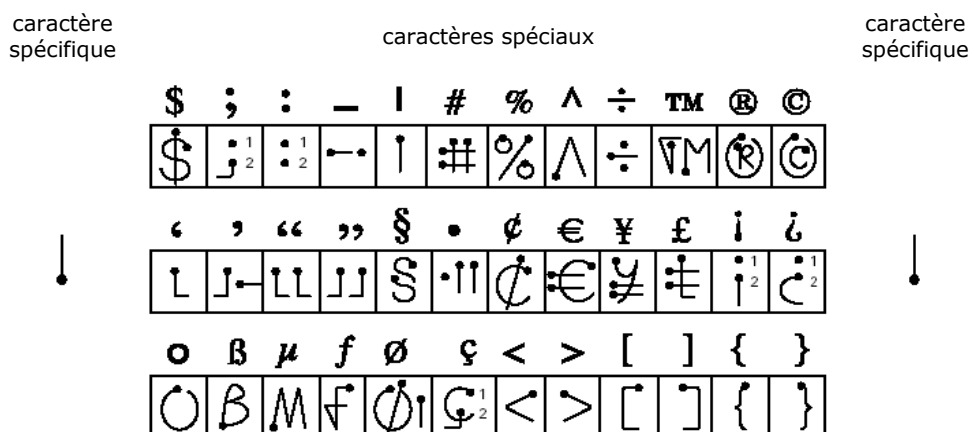
Avec l'avancée des nouvelles technologies (systèmes portables, PDA et autres), il est devenu inévitable de développer des systèmes plus pratiques, avec plus de formes. Même si ce type de méthodes oblige l'utilisateur à apprendre un nouvel alphabet, c'est néanmoins cette solution que les développeurs de Palm Computing ont choisie pour implémenter leur produit appelé Graffiti [26].

La figure ci-dessous représente l'ensemble des caractères utilisés, les points en gras indiquant le début des gestes. Les lettres, les chiffres et les caractères spéciaux sont tracés d'une manière bien précise, identique pour tous.



En plus de l'alphabet prédéfini à apprendre, l'utilisateur est contraint de spécifier le type de caractères qu'il souhaite écrire. Sur la partie gauche de la zone d'écriture, il doit représenter les lettres et la ponctuation ; sur la partie droite, les chiffres et leurs opérateurs éventuels et, sur la partie centrale, les lettres en majuscule. Quant aux caractères accentués, il faut d'abord dessiner la lettre dans la partie de gauche puis l'accent sur la droite.

Pour tout ce qui est des caractères spéciaux, dans un premier temps, on est obligé d'entrer le caractère spécifique représentant une barre verticale, puis le caractère souhaité, puis de nouveau la barre verticale pour revenir en mode normal :



Graffiti est très facile à utiliser et bien qu'il faille apprendre un nouvel alphabet, l'ensemble des gestes choisis par les développeurs est assez rapide à mémoriser, d'autant plus qu'ils ressemblent beaucoup aux lettres et aux chiffres que nous connaissons.

Pour une reconnaissance optimale avec un risque de confusion moindre, les deux logiciels présentés ici (MyScript et Graffiti) utilisent des lignes de référence sur lesquelles l'utilisateur doit positionner ses caractères :



## 2. La reconnaissance de gestes graphiques

Les applications graphiques type dessin, annotation et correction de documents électroniques,... nécessitent des traitements concernant la reconnaissance de gestes.

On retrouve principalement deux modèles d'algorithmes de reconnaissance de gestes. Tout d'abord, il est possible d'utiliser les réseaux de neurones mais cette technique demande beaucoup de gestes exemples. D'autre part, on distingue des méthodes d'ordre statistique ou géométrique. Pour l'article, deux procédés ont été retenus : celui des chercheurs Fonseca et Jorge ([12], [13], [14]) et plus particulièrement l'algorithme basé sur les caractéristiques de Rubine ([22]) qui est souvent cité dans les articles spécialisés. Une dernière technique, assez peu développée pour le on-line, concerne les méthodes structurales.

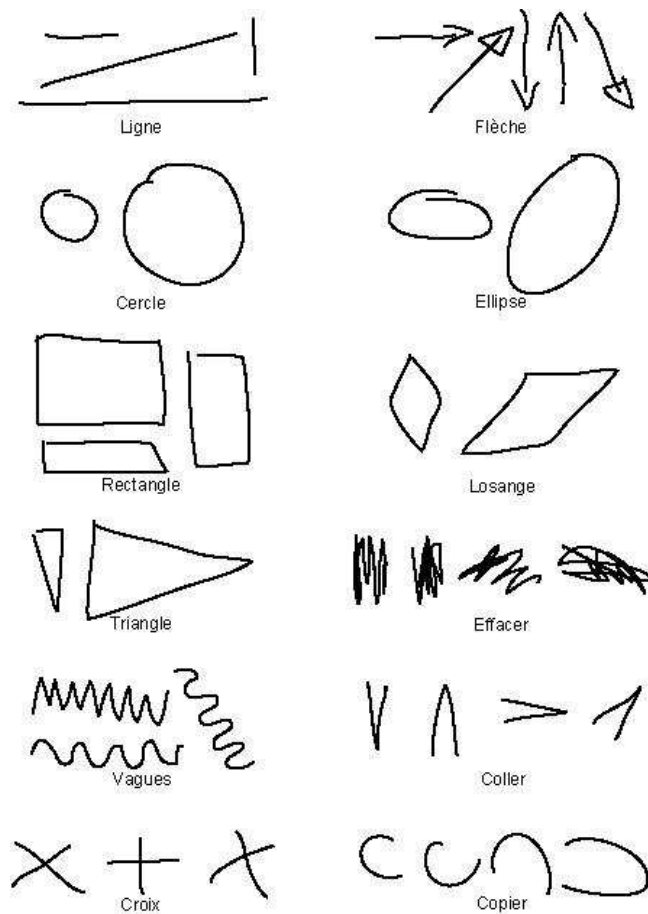
### 2.1. Méthode structurale

Pour l'instant, peu d'études sur les méthodes structurales ont été développées dans le cadre de la reconnaissance de gestes on-line. Par contre, au niveau des traitements hors ligne, il existe déjà quelques travaux. Par exemple, les articles [7] et [8] proposent une méthode d'appariement de deux graphes structuraux quelconques en vue de la reconnaissance de lettres manuscrites isolées.

### 2.2. Méthode statistique et géométrique de Fonseca et Jorge

Dans le cadre de son PFE ([23]), N. Hervouet a développé un logiciel nommé TextChecker mettant en œuvre les travaux de Fonseca et Jorge ([12], [13], [14]). Son éditeur de texte avancé propose à l'utilisateur de corriger un document texte électronique à l'aide de gestes manuscrits. L'utilisation de l'interface stylo dans ce cadre semble plus simple que la manipulation de la souris. En effet, on associe un seul geste à une commande plus ou moins complexe.

Les formes reconnues correspondent aux figures géométriques élémentaires (triangle, rectangle, losange, cercle, ellipse, ligne et flèche) auxquelles on ajoute cinq commandes gestuelles : supprimer, vague, déplacer, copier, coller :



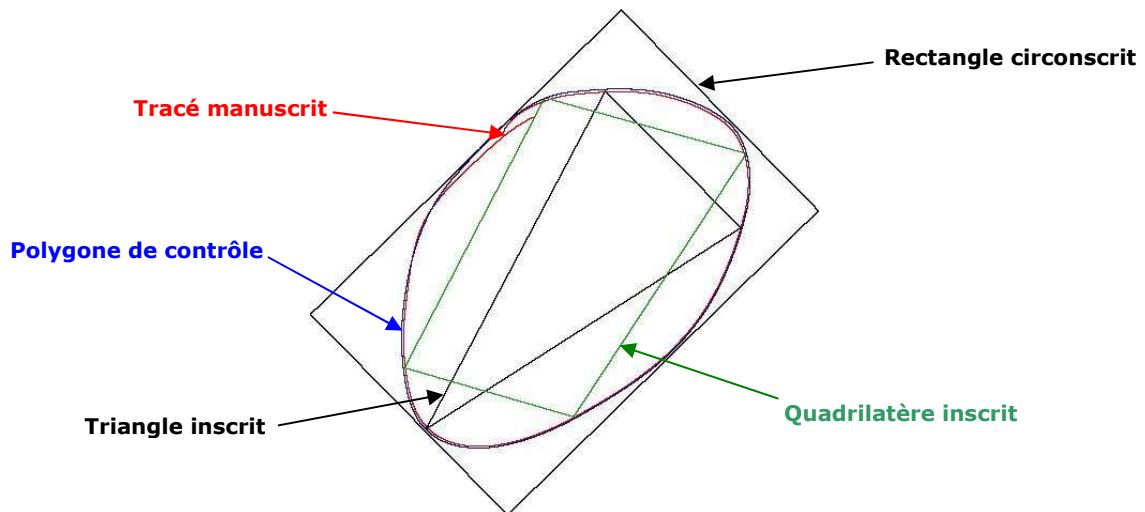
On remarque que les formes peuvent être tracées en un ou plusieurs coups de crayon, le problème des gestes dessinés en plusieurs fois (cas d'une flèche, d'une croix ou d'une forme en pointillés...) étant résolu par l'utilisation d'un timeout.

Les commandes associées à ces formes sont :

- sélectionner du texte avec une ellipse, un cercle, un rectangle ou deux barres verticales ;
- déplacer du texte avec le symbole 'V' (ou le triangle) ;
- insérer du texte avec une flèche (symbole créé avec un trait + un 'V') ;
- supprimer du texte avec un trait horizontal ou avec une rature ;
- couper du texte avec une croix ;
- copier avec le symbole 'C' ;
- coller avec le symbole 'V' (ou un triangle).

La technique de reconnaissance de gestes de Fonseca et Jorge comporte les étapes suivantes :

Dans un premier temps, on applique l'algorithme de segmentation de Graham [21] permettant de mettre en évidence le polygone de contrôle. Puis, on trace plusieurs formes géométriques qui permettront de différencier les gestes par la suite : le triangle et le quadrilatère inscrits de plus grande aire [5] et le rectangle circonscrit de plus petite aire [15] :



De ces formes géométriques, on extrait toutes les données concernant leur hauteur, largeur, périmètre et aire. Enfin, en fonction de ces paramètres, on dégage des coefficients significatifs qui sont associés à la logique floue. Les ensembles flous, contrairement aux ensembles traditionnels, autorisent leurs membres à leurs appartenir partiellement. Cette technique est ici utilisée pour reconnaître les formes dessinées en biais et également pour lever les problèmes d'imprécision ou d'ambiguïté.

Par exemple, les études ont montré qu'un cercle est reconnu par le rapport  $P_{PC}^2 / A_{PC} \approx 4\pi$  (avec  $P_{PC}$  : le périmètre du polygone de contrôle et  $A_{PC}$  son aire), d'un autre côté, une ligne est identifiée par un rapport entre la hauteur et la largeur du rectangle circonscrit proche de 0, etc.

Avec cette technique, on peut également distinguer les lignes en pointillés des lignes normales en observant le nombre de coups de crayon pour une forme. De la même manière, on peut mettre en évidence les lignes en gras qui correspondent simplement à une répétition de lignes normales.

Les résultats expérimentaux utilisant cette technique de reconnaissance sont plutôt concluants. En effet, les formes sont reconnues en moyenne à 92 %. Ces résultats sont encore plus élevés (97%) si l'on regroupe les formes les plus ambiguës (soient les ellipses avec les cercles et les losanges avec les rectangles).

L'inconvénient majeur de cette méthode réside dans le fait qu'elle ne reconnaisse qu'une douzaine de gestes sans aucune possibilité d'apprendre de nouvelles formes.

## 2.3. Méthode statistique et géométrique de Rubine

### 2.3.1. Description de la méthode

L'étude la plus conséquente concernant la reconnaissance de gestes par une méthode de classification statistique est sans doute celle de Rubine [22]. Rubine a créé non seulement un système de reconnaissance de gestes mais également un procédé permettant de créer ses propres gestes en les faisant apprendre au système.

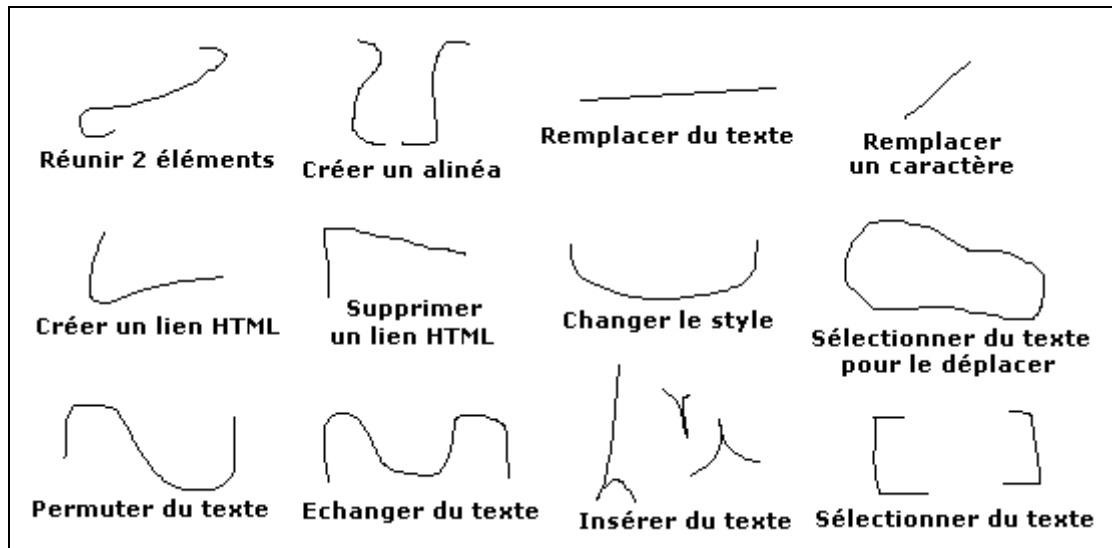
Les gestes considérés sont définis, au départ, par un tableau de points contenant à la fois, leurs coordonnées et l'instant où ils ont été dessinés. De chaque geste, on peut extraire des caractéristiques concernant la taille de la forme, les angles parcourus ou la vitesse de tracé, par exemple. La phase d'apprentissage consiste à construire un ensemble de classes à l'aide de plusieurs gestes exemples. Par la suite, étant donné un geste en entrée, le problème de reconnaissance est de déterminer à quelle classe il appartient.

Les caractéristiques définies par Rubine et sa technique d'apprentissage et de reconnaissance sont définies dans le chapitre II car c'est l'algorithme retenu pour notre système de reconnaissance de gestes.

De nombreux chercheurs ont mis en œuvre les caractéristiques de Rubine. Prenons l'exemple d'une application Web étudiée récemment.

### 2.3.2. Exemple d'application : Interface de correction en ligne de documents électroniques

Les symboles utilisés ici par les chercheurs de l'INSA de Rennes ([1], [2], [3]) sont adaptés aux applications Web :



Pour l'instant, le logiciel créé (appelé AmayaPen) nécessite l'utilisation de l'interface clavier pour les gestes de remplacement et d'insertion de texte. Quand au geste permettant de modifier le style, il engendre l'ouverture d'une fenêtre contenant un menu de styles. Pour le geste concernant la création d'un lien HTML, deux possibilités s'offrent à l'utilisateur : soit l'entrée au clavier du lien, soit le pointage vers le document à lier.

La reconnaissance des gestes graphiques s'effectue en deux étapes : la reconnaissance géométrique et la reconnaissance contextuelle (ou localisation).

Pour la reconnaissance géométrique, on utilise les caractéristiques de [Rubine] auxquelles sont ajoutés le nombre de lignes coupées par le geste et le nombre d'intersections par ligne. La reconnaissance géométrique est effectuée grâce aux réseaux de neurones (sans rejet).

Les points de localisation correspondent aux points d'intersection avec une ligne de texte. Selon la commande à réaliser, les points de localisation doivent être 'SUR' ou 'ENTRE' les caractères. Les zones inter-caractères peuvent être de largeur fixe ou de largeur variable.

Un ordre de correction n'est donné que quand le symbole est reconnu et localisé.

Les résultats sont assez concluants : le taux de reconnaissance géométrique atteint 96,5 % et le taux de reconnaissance des ordres de correction est de 84,9 %. L'avantage de ce programme est que les symboles de correction sont simples et donc faciles à apprendre. L'inconvénient est que l'utilisation du clavier est encore indispensable.

### 3. Conclusion

Pour ce qui est de la reconnaissance de gestes, les logiciels créés pour l'instant sont limités par le nombre de formes reconnues. En général, on remarque que les utilisateurs ont besoin de plus de gestes, et souhaiteraient également créer leurs propres gestes afin de personnaliser leurs applications (cf. étude [17]). Il faudrait donc développer un logiciel permettant d'apprendre de nouvelles formes.

Nous avons retenu l'algorithme de Rubine [22] car il nous donne justement cette opportunité. Dans le chapitre suivant, nous allons détailler cette méthode dans l'optique de créer un logiciel permettant d'apprendre et de reconnaître des gestes en ligne. L'implémentation sera détaillée au chapitre III.

# Méthode de Rubine

## II. Méthode de Rubine

### 1. Description de la méthode

Les formes peuvent être entrées avec la souris ou à l'aide d'un stylet sur une tablette graphique. Les gestes considérés sont définis par un tableau  $g$  de  $P$  points contenant leurs coordonnées et l'instant où ils ont été dessinés, soit :  $g_p = (x_p, y_p, t_p)$  avec  $0 \leq p < P$ . Le début et la fin d'une forme doivent être clairement délimités : par exemple, le début du geste peut être caractérisé par l'appui sur le bouton gauche de la souris et la fin du geste par le relâchement de ce même bouton, ou simplement par le pointage du stylet sur la tablette graphique et la levée du stylet en fin de geste ou la fin d'un timeout.

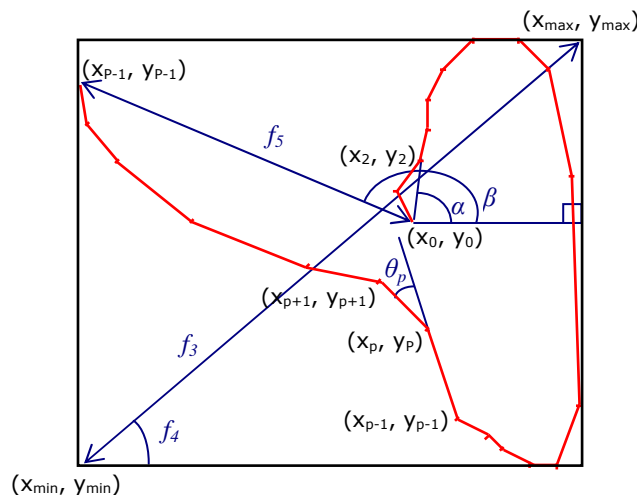
Soit  $C$  le nombre de classes, chacune identifiée par  $E_c$  gestes exemples. Etant donné un geste  $g$  en entrée, le problème de reconnaissance est le suivant : déterminer la classe  $c$  à laquelle  $g$  appartient.

La reconnaissance statistique s'effectue en deux étapes. Dans un premier temps, un vecteur de caractéristiques, noté  $f = [f_1, f_2, \dots, f_F]$ , est extrait du geste entré. Ensuite, le vecteur caractéristique est classé parmi les  $C$  classes possibles via un classificateur linéaire.

### 2. Caractéristiques de Rubine

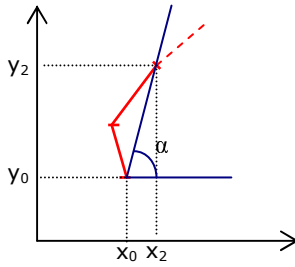
La figure ci-dessous montre les caractéristiques géométriques et algébriques utilisées par Rubine, soient :

- le cosinus ( $f_1$ ) et le sinus ( $f_2$ ) de l'angle initial du geste ;
- la longueur ( $f_3$ ) de la diagonale du rectangle circonscrit ;
- l'angle ( $f_4$ ) effectué par cette diagonale ;
- la distance ( $f_5$ ) entre le premier et le dernier point du geste ;
- le cosinus ( $f_6$ ) et le sinus ( $f_7$ ) de l'angle effectué entre le premier et le dernier point du geste ;
- la longueur totale ( $f_8$ ) du geste ;
- la somme des angles traversés ( $f_9$ ) par le geste ;
- la somme des valeurs absolues de tous les angles ( $f_{10}$ ) ;
- la somme des carrés de ces angles ( $f_{11}$ ) ;
- la vitesse maximum du geste au carré ( $f_{12}$ ) ;
- la durée de représentation du geste ( $f_{13}$ ).



On préfère utiliser le cosinus et le sinus plutôt que la valeur des angles eux-mêmes pour éviter une discontinuité quand l'angle passe de  $2\pi$  à 0.

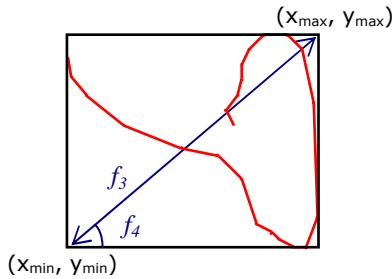
Le cosinus et le sinus de l'angle initial,  $f_1$  et  $f_2$ , sont calculés d'après le premier et le troisième point de la forme, car le résultat est plus intéressant et moins bruité que lorsque l'on utilise seulement les deux premiers points.



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

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

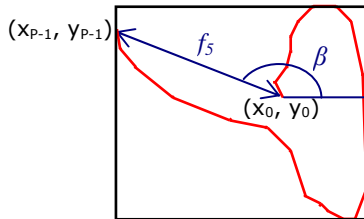
Les caractéristiques  $f_3$  et  $f_4$  sont définies à l'aide du rectangle englobant le geste.  $f_3$  est la distance euclidienne entre les points  $(x_{\min}, y_{\min})$  et  $(x_{\max}, y_{\max})$ , c'est-à-dire la longueur de la diagonale du rectangle. Quant à  $f_4$ , c'est l'angle effectué par cette diagonale.



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

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

$f_5$  est la distance entre le premier  $(x_0, y_0)$  et le dernier point  $(x_{p-1}, y_{p-1})$ .  $f_6$  et  $f_7$  sont le cosinus et le sinus de l'angle  $\beta$  créé par ces deux points, soient :



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

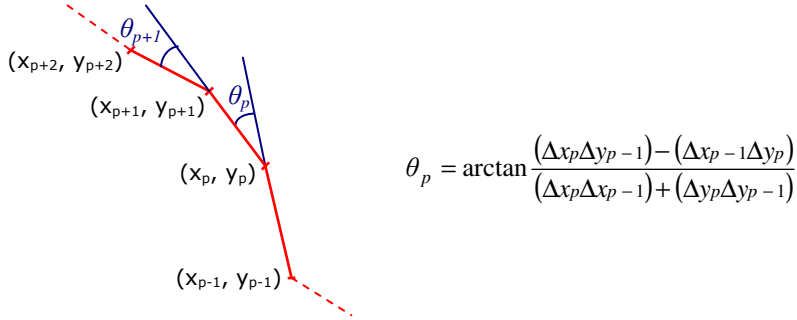
$$f_6 = \cos \beta = \frac{(x_{p-1} - x_0)}{f_5}$$

$$f_7 = \sin \beta = \frac{(y_{p-1} - y_0)}{f_5}$$

On note  $\Delta x_p$  et  $\Delta y_p$  les différences d'abscisses et d'ordonnées entre les points  $p$  et  $p+1$ , soient  $\Delta x_p = x_{p+1} - x_p$  et  $\Delta y_p = y_{p+1} - y_p$ . La distance entre les points  $p$  et  $p+1$  peut donc s'exprimer par  $\sqrt{\Delta x_p^2 + \Delta y_p^2}$ .  $f_8$ , la longueur totale du geste, correspond à la somme des distances entre chaque point, c'est à dire :

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

On note  $\theta_p$  l'angle formé au point  $p$ , c'est-à-dire l'angle effectué par les segments  $(p-1, p)$  et  $(p, p+1)$  :



On peut ainsi calculer la somme totale de tous les angles traversés par le geste ( $f_9$ ). On remarque que  $f_9$  peut être négatif car il peut y avoir des  $\theta_p$  négatifs. Il est également intéressant de rechercher la somme des valeurs absolues de ces angles ( $f_{10}$ ).

$$f_9 = \sum_{p=1}^{P-2} \theta_p \quad f_{10} = \sum_{p=1}^{P-2} |\theta_p|$$

La caractéristique  $f_{11}$  est nécessaire pour distinguer les gestes arrondis des gestes anguleux, comme 'U' et 'V' par exemple. Elle correspond à la somme des valeurs au carré des angles traversés par le geste :

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

On note  $\Delta t_p$  le temps écoulé entre les points  $p$  et  $p+1$ , soit  $\Delta t_p = t_{p+1} - t_p$ .

La caractéristique  $f_{12}$  correspond à la vitesse maximum du geste, c'est-à-dire le temps minimum entre deux points successifs. On utilise le carré des valeurs pour éviter les problèmes de signe.

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

Quant à  $f_{13}$ , elle représente la durée totale de construction du geste, soit la différence entre le moment de départ et l'instant final du tracé.

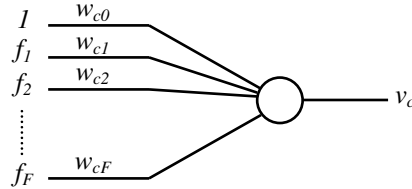
$$f_{13} = t_{P-1} - t_0$$

Ces deux dernières propriétés ajoutent donc l'aspect dynamique au traitement. Grâce à ces valeurs, les gestes ne représentent pas uniquement des images (statiques). On pourrait supprimer ces caractéristiques pour réaliser des traitements off line.

### 3. Classification des gestes

Etant donné le vecteur caractéristique  $f$  calculé pour le geste  $g$ , l'algorithme de classification est simple et efficace. Une fonction d'évaluation est associée à chaque classe de gestes. On donne un poids  $w_{cj}$  à chaque classe (avec  $0 \leq j \leq F$ ,  $F$  étant le nombre total de caractéristiques), le poids de la caractéristique 0 étant considérée comme le biais.

Pour chaque classe  $c$ ,



La fonction d'évaluation  $v_c$  est alors calculée pour chaque classe  $0 \leq c < C$  par l'expression :

$$v_c = w_{c0} + \sum_{i=1}^F w_{ci} f_i .$$

On classe le geste  $g$  dans la classe  $c$  pour laquelle la fonction d'évaluation  $v_c$  est maximum.

La notion de rejet sera développée plus loin.

### 4. Apprentissage

L'apprentissage consiste à déterminer les poids  $w_{ci}$  associés aux caractéristiques  $i$  de la classe  $c$ . La formule utilisée est connue pour donner de bons classificateurs.

Tout d'abord, notons  $f_{ce}$ , le vecteur caractéristique du  $e^{ième}$  exemple de la classe  $c$  et par conséquent,  $f_{cei}$ , la caractéristique  $i$  associée à cet exemple, soit :

$$f_{ce} = \begin{pmatrix} f_{ce1} \\ f_{ce2} \\ \dots \\ f_{cei} \\ \dots \\ f_{ceF} \end{pmatrix} .$$

A une classe  $c$  va donc correspondre  $E_c$  vecteurs de propriétés :

$$f_{c0}, f_{c1}, \dots, f_{c(E_c-1)} = \begin{pmatrix} f_{c01} \\ f_{c02} \\ \dots \\ f_{c0i} \\ \dots \\ f_{c0F} \end{pmatrix}, \begin{pmatrix} f_{c11} \\ f_{c12} \\ \dots \\ f_{c1i} \\ \dots \\ f_{c1F} \end{pmatrix}, \dots, \begin{pmatrix} f_{c(E_c-1)1} \\ f_{c(E_c-1)2} \\ \dots \\ f_{c(E_c-1)i} \\ \dots \\ f_{c(E_c-1)F} \end{pmatrix} .$$

On calcule ensuite  $\overline{f_c}$  qui représente le vecteur moyen de la classe  $c$ , c'est-à-dire la moyenne des caractéristiques associées à ses exemples :

$$\overline{f_c} = \begin{pmatrix} \overline{f_{c1}} \\ \overline{f_{c2}} \\ \dots \\ \overline{f_{ci}} \\ \dots \\ \overline{f_{cF}} \end{pmatrix} \text{ avec } \overline{f_{ci}} = \frac{1}{E_c} \sum_{e=0}^{E_c-1} f_{cei} = \frac{1}{E_c} (f_{c0i} + f_{c1i} + \dots + f_{c(E_c-1)i}), \forall i \ 1 \leq i < F,$$

$$\text{soit : } \overline{f_c} = \frac{1}{E_c} \left( \begin{pmatrix} f_{c01} \\ f_{c02} \\ \vdots \\ f_{c0F} \end{pmatrix} + \begin{pmatrix} f_{c11} \\ f_{c12} \\ \vdots \\ f_{c1F} \end{pmatrix} + \dots + \begin{pmatrix} f_{c(E_c-1)1} \\ f_{c(E_c-1)2} \\ \vdots \\ f_{c(E_c-1)F} \end{pmatrix} \right).$$

Puis, on cherche la matrice de variances-covariances des caractéristiques d'une classe  $c$  :

$$\Sigma_c = \frac{1}{E_c} \begin{pmatrix} \Sigma_{c11} & \Sigma_{c12} & \dots & \Sigma_{c1j} & \dots & \Sigma_{c1F} \\ \Sigma_{c21} & \Sigma_{c22} & \dots & \Sigma_{c2j} & \dots & \Sigma_{c2F} \\ \vdots & & \ddots & \vdots & & \\ \Sigma_{ci1} & \Sigma_{ci2} & \dots & \Sigma_{cij} & \dots & \Sigma_{ciF} \\ \vdots & & & \vdots & \ddots & \\ \Sigma_{cF1} & \Sigma_{cF2} & \dots & \Sigma_{cFj} & \dots & \Sigma_{cFF} \end{pmatrix} \text{ avec } \Sigma_{cij} = \sum_{e=0}^{E_c-1} (f_{cei} - \overline{f_{ci}}) (f_{cej} - \overline{f_{cj}})$$

On peut alors déduire la matrice moyenne de variances-covariances (sur toutes les classes) :

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \dots & \Sigma_{1j} & \dots & \Sigma_{1F} \\ \Sigma_{21} & \Sigma_{22} & \dots & \Sigma_{2j} & \dots & \Sigma_{2F} \\ \vdots & & \ddots & \vdots & & \\ \Sigma_{i1} & \Sigma_{i2} & \dots & \Sigma_{ij} & \dots & \Sigma_{iF} \\ \vdots & & & \vdots & \ddots & \\ \Sigma_{F1} & \Sigma_{F2} & \dots & \Sigma_{Fj} & \dots & \Sigma_{FF} \end{pmatrix} \text{ avec } \Sigma_{ij} = \frac{\sum_{c=0}^{C-1} \Sigma_{cij}}{\sum_{c=0}^{C-1} E_c - C}$$

Les poids sont alors estimés à l'aide de l'inverse de cette matrice, notée  $\Sigma^{-1}$  :

$$w_{cj} = \sum_{i=1}^F (\Sigma^{-1})_{ij} \cdot \overline{f_{ci}}, \forall j \ 1 \leq j \leq F.$$

Pour terminer, le biais  $w_{c0}$  est déterminé à l'aide des poids  $w_{cj}$ ,  $1 \leq j \leq F$ , de la manière suivante :

$$w_{c0} = -\frac{1}{2} \sum_{i=1}^F w_{ci} \cdot \overline{f_{ci}}.$$

## 5. Rejet

Un classificateur linéaire permet de classer n'importe quel geste. On présente ci-dessous une méthode pour rejeter les gestes ambigus.

### 5.1. Calcul des probabilités

Intuitivement, on peut calculer la probabilité que le geste  $g$  appartienne réellement à la classe  $i$  :

$$p(i/g) = \frac{1}{\sum_{j=0}^{C-1} e^{(v_j - v_i)}},$$

$v_i$  et  $v_j$  étant les valeurs des fonctions d'évaluation des classes  $i$  et  $j$ ,  $v_i > v_j \quad \forall j \neq i$ .

Rejeter le geste si  $p(i/g) < 0.95$  marche plutôt bien dans la pratique.

### 5.2. Distance de Mahalanobis

Une autre possibilité est d'utiliser la distance de Mahalanobis (entre le geste et le centre de gravité de la classe à laquelle on l'a affecté) :

$$\delta^2 = \sum_{j=1}^F \sum_{k=1}^F (\Sigma^{-1})_{jk} (f_j - \overline{f_{ij}})(f_k - \overline{f_{ik}}),$$

$\overline{f_{ij}}$  étant la moyenne de la caractéristique  $j$  pour la classe  $i$ .

On peut rejeter le geste si  $\delta^2 > \frac{1}{2} F^2$ .

# Application logicielle

## III. Application logicielle

---

### 1. Présentation du logiciel

Afin de comparer facilement les résultats des méthodes de reconnaissance de gestes, l'implémentation de l'algorithme de Rubine sera intégrée à un logiciel existant appliquant la méthode de Jorge et Fonseca.

#### 1.1. Le logiciel existant

L'application logicielle existante a été réalisée par Nicolas Hervouet (PFE 2001-2002 - E3i [18]) en C++. Le logiciel créé est un éditeur de textes combiné avec un système de correction de documents à l'aide de gestes simples, tracés à la souris ou avec un stylet sur une tablette graphique.

##### 1.1.1. Présentation

Habituellement, pour corriger un document électronique, la technique la plus utilisée est assez fastidieuse. Elle consiste à imprimer tout d'abord le document texte, puis effectuer une relecture du document papier afin d'ajouter des commentaires, des annotations, des ratures,... Enfin, l'utilisateur peut reprendre le document électronique en associant les annotations du document papier.

On peut également corriger le fichier directement avec l'éditeur utilisé lors de la création, mais la plupart des personnes préfèrent la version papier pour sa commodité et la possibilité de faire des annotations manuscrites.

D'où l'idée de proposer un outil permettant de corriger son document directement dans un éditeur, à l'aide de gestes simples. L'utilisateur aura, avec le stylet, des sensations proches de l'écriture manuscrite.

Le logiciel comprend donc trois axes principaux :

- un éditeur de textes ;
- un système de reconnaissance de gestes ;
- un système de correction de documents.

##### 1.1.2. L'éditeur de textes

L'éditeur de textes est comparable à WordPad de Microsoft. Il possède les fonctionnalités incontournables du type :

- création de nouveaux documents, ouverture de documents existants, enregistrement ;
- impression du document en cours ;
- fonctions d'édition indispensable : copier, couper, coller ;
- alignement du texte ;
- changement de police de caractères, formatage de texte : gras, italique, souligné...

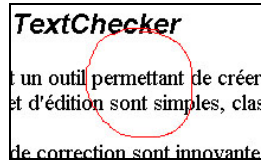
##### 1.1.3. Le système de reconnaissance de gestes

La reconnaissance de gestes utilise la technique de Jorge et Fonseca ([12], [13], [14]) développée dans le premier chapitre (paragraphe 2.2.). Cette méthode consiste à reconnaître une douzaine de gestes : d'un côté, des formes géométriques simples (cercle, ellipse, rectangle, losange, triangle, ligne) et de l'autre, des gestes un peu plus complexes (symboles 'V' et 'C', flèche, rature, vague, croix).

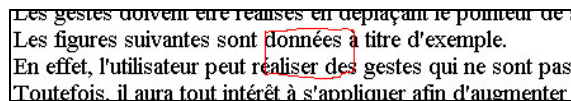
#### 1.1.4. Le système de correction de documents

Le but de cette étape a été d'associer les gestes présentés précédemment à des actions utilisateur :

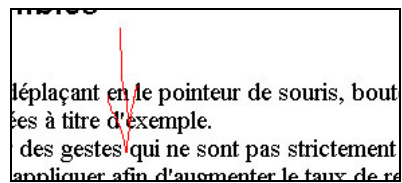
- sélection d'un mot avec un cercle ou une ellipse :



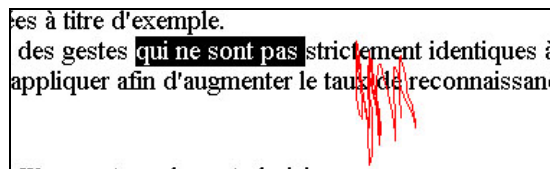
- sélection d'une zone de texte avec un rectangle ou deux barres verticales délimitant le début et la fin du texte à sélectionner :



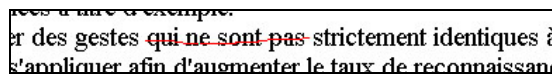
- insertion de texte avec une flèche :



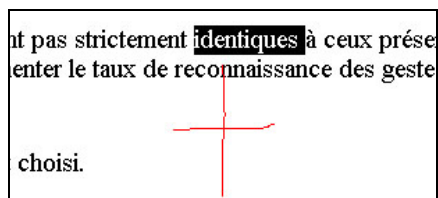
- suppression de la sélection avec une rature :



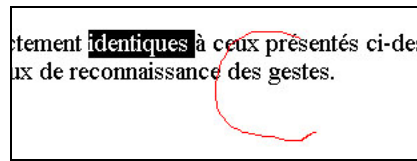
- suppression de caractères avec une ligne horizontale :



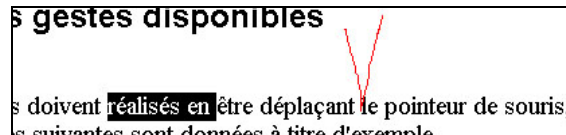
- couper la sélection avec une croix :



- copier la sélection avec le symbole 'C' :



- et enfin, coller le texte copié ou déplacer la sélection avec le symbole 'V' :



### 1.2. Besoins

La technique de reconnaissance de gestes utilisée dans l'éditeur de textes a le principal inconvénient de ne reconnaître que douze formes simples définies préalablement par le programmeur. Pour la plupart des applications informatiques à base de stylet, un ensemble comprenant plus de gestes serait nécessaire et bien plus pratique. On va donc mettre en œuvre la méthode de Rubine développée au chapitre II afin de créer un outil permettant d'apprendre des gestes dessinés par l'utilisateur lui-même.

Après avoir implémenté l'algorithme de Rubine, il serait intéressant d'associer le résultat à un domaine précis. Plusieurs possibilités s'offrent alors à nous. On pourrait améliorer le correcteur de documents électroniques de N. Hervouet en ajoutant beaucoup plus de commandes d'annotation et de correction (changer le style de la police, sa taille, surligner du texte, permuter de mots, insérer une en-tête ou un pied de page...) ; on pourrait également créer un logiciel permettant de redessiner le tracé manuscrit de tableaux ; etc.

L'application retenue consiste à associer le principe de reconnaissance de gestes à un éditeur d'organigrammes. On a choisi plusieurs formes géométriques habituellement utilisées lors de la construction d'un organigramme (losanges, rectangles, lignes,...). Ces formes sont automatiquement redessinées lorsque le geste associé est reconnu. On propose également une liste de commandes utilisateur permettant de modifier les composants graphiques.

Dans les paragraphes suivants, on présente les analyses conceptuelles de ces deux applications, suivies de leur fonctionnement logiciel.

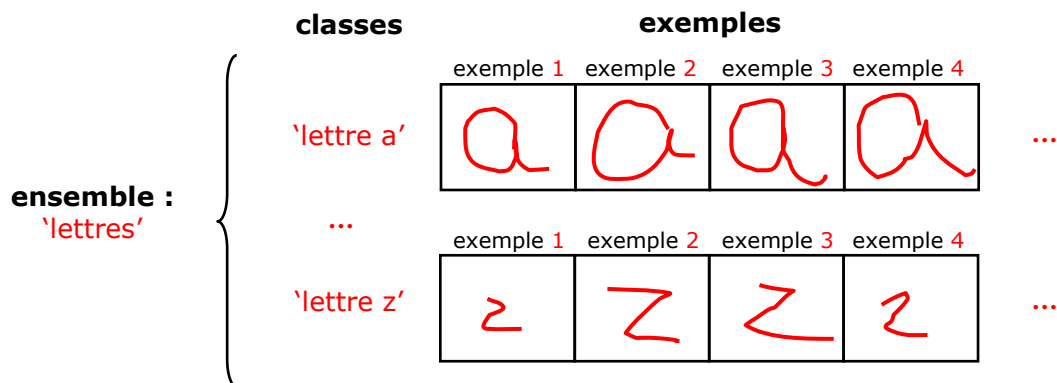
## 2. Système d'apprentissage et de reconnaissance de gestes

### 2.1. Présentation

#### 2.1.1. Vocabulaire utilisé

On appelle *ensemble*, la base de gestes utilisée. Les gestes sont ordonnés sous forme de classes, chaque *classe* contenant plusieurs *exemples*.

Illustrons avec un ensemble nommé '*lettres*'. Les classes correspondent ici aux vingt-six lettres de l'alphabet et à chaque lettre sont associés plusieurs gestes exemples :



#### 2.1.2. Fonctionnalités principales

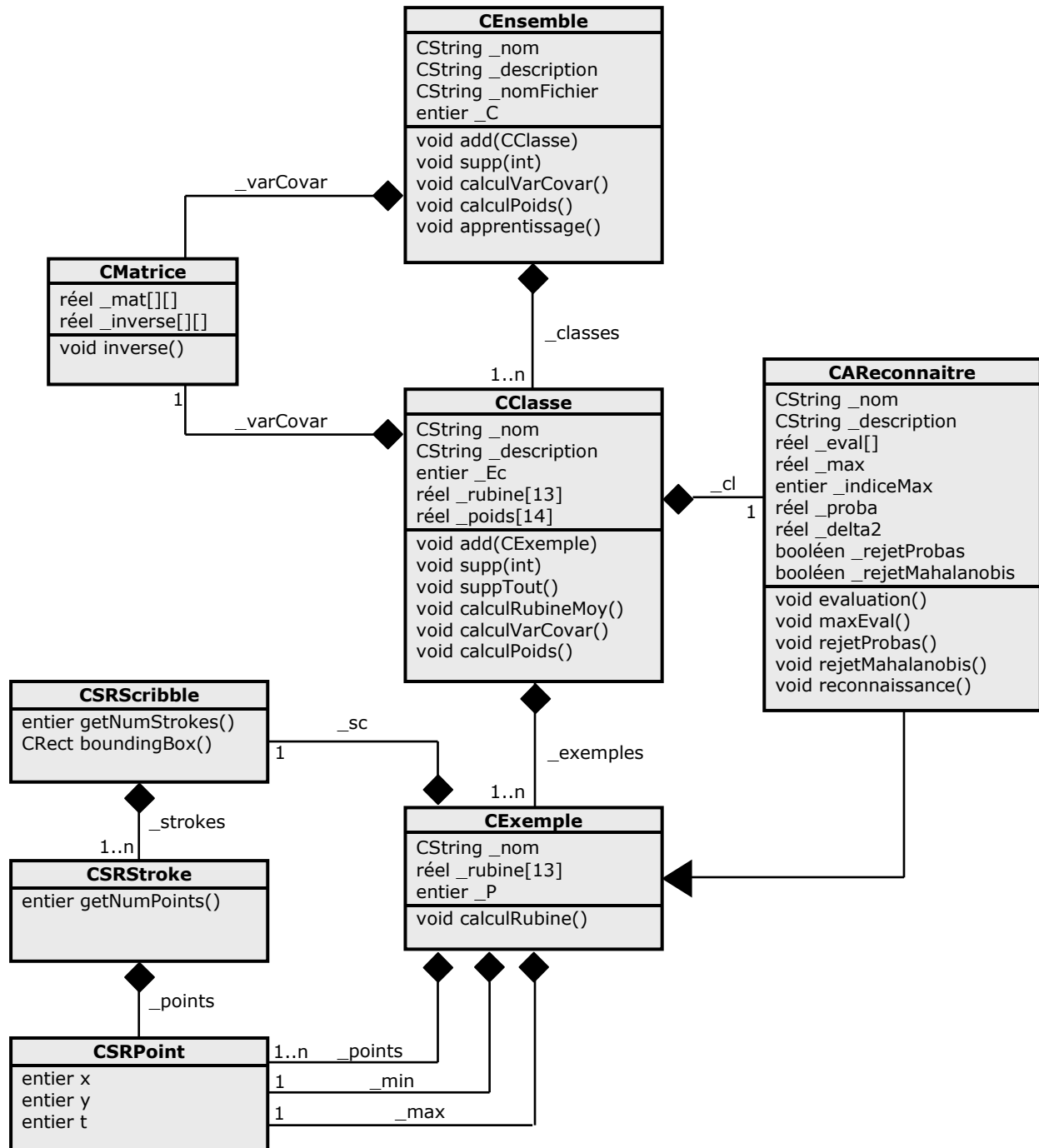
L'application à implémenter devra permettre à l'utilisateur d'apprendre un nouvel ensemble de gestes au système. Pour cela, le logiciel devra être capable d'apprendre de nouvelles classes et donc de nouveaux gestes exemples.

On devra également trouver une solution afin d'autoriser l'utilisateur d'effectuer des modifications sur un ensemble, à tout moment (ajout, suppression d'exemples ou de classes).

## 2.2. Programmation

Le logiciel étant développé en C++, il est indispensable d'effectuer une modélisation orientée objet. Dans une première partie, on exposera un diagramme de classes contenant les éléments principaux du système. Ensuite, on décrira les classes une par une, en dégagant les points essentiels des algorithmes d'apprentissage et de reconnaissance.

### 2.2.1. Diagramme de classes



légende : les losanges représentent la relation 'est composé de' et le triangle exprime la notion d'héritage.

### 2.2.2. Apprentissage

Les trois classes principales concernant l'apprentissage de gestes sont *CEnsemble*, *CClasse* et *CExemple*. Le diagramme de classes met en évidence le fait qu'un ensemble de gestes est composé de classes, elles-mêmes constituées de gestes exemples.

#### 2.2.2.1. Description des classes

##### 1. CExemple

Un geste, défini par la classe *CExemple*, correspond principalement à une forme (*CSRScribble*) composée d'un ou plusieurs coups de crayon (*CSRStroke*). Un trait (ou coup de crayon) est caractérisé par une liste de points. On utilise ici notre propre classe *CSRPoint*, au lieu de la classe *CPoint* existant dans Visual Studio, car en plus des coordonnées du point ( $x, y$ ), on souhaite mémoriser l'instant pendant lequel il a été tracé ( $t$ ).

L'attribut appelé *\_points* regroupe la liste de points ordonnés constituant le geste. Ainsi, pour gérer les gestes dessinés en plusieurs coups de crayon, on a décidé de concaténer les listes de points représentant chaque tracé. Cette caractéristique donne la possibilité au système de reconnaître les formes en pointillés de la même manière que les formes en trait plein. Par conséquent, si l'utilisateur relâche légèrement le stylet de la tablette, involontairement, les résultats ne seront pas faussés. De cette façon on pourra très facilement reconnaître tous les gestes nécessitant plusieurs traits, comme, par exemple, les croix, les flèches, les lettres majuscules A, E, F,...

L'entier *\_P* contient le nombre total de points représentant le geste.

Les points *\_min* et *\_max* correspondent respectivement aux points en haut à gauche et en bas à droite du rectangle englobant du geste, ce dernier étant trouvé grâce à la fonction *boundingBox()* de la classe *CSRScribble*.

Enfin, l'attribut *\_rubine* est le tableau regroupant les treize caractéristiques de Rubine pour le geste en question.

##### 2. CClasse

Cette classe permet de regrouper les gestes exemples. L'entier *\_Ec* représente le nombre d'exemples appartenant à la classe. Chaque classe contient donc une liste de *CExemple*. Les fonctions *add()* et *supp()* permettent d'ajouter ou de supprimer un exemple à la liste. On peut même utiliser la méthode *suppTout()* pour supprimer directement tous les exemples d'une classe.

A chaque fois que l'on va modifier la base (ajout ou suppression d'exemple), il sera nécessaire de mettre à jour différents points. Tout d'abord, il faut calculer le vecteur moyen des caractéristiques de Rubine (*\_rubine*). Ce vecteur correspond à la moyenne des vecteurs caractéristiques des exemples de la classe, il est calculé à l'aide de la fonction *calculRubineMoy()*. Ensuite, on a besoin de réinitialiser la matrice de variance-covariance (*\_varCovar*) avec la méthode *calculVarCovar()*. Plus tard, on devra mettre à jour les poids (*\_poids*) associés aux caractéristiques de la classe (utilisation de la méthode *calculPoids()*).

En plus, on utilise deux chaînes de caractères définissant le nom et la description du geste. Par exemple, pour l'ensemble des lettres de l'alphabet, chaque classe portera le nom de la lettre dessinée ('a', 'b', 'c',...) et une éventuelle description (expliquant par exemple la trajectoire à reproduire pour que le geste soit reconnu).

### 3. CEnsemble

*CEnsemble* permet de regrouper les différentes classes de gestes. Par conséquent, à un ensemble correspond une liste de *\_C* classes. De la même manière que pour la gestion des exemples au sein d'une classe, on possède ici les fonctions *add()* et *supp()* pour ajouter ou supprimer une classe à un ensemble.

A chaque modification d'une classe de l'ensemble (ajout ou suppression d'un exemple ou d'une classe), il faudra remettre à jour les poids associés aux caractéristiques de chaque classe en utilisant la méthode *calculPoids()*. Cette fonction calcule tout d'abord la matrice de variance-covariance moyenne (*\_varCovar*) avec la méthode *calculVarCovar()*, puis sa matrice inverse, pour enfin déduire les poids de chaque classe. La fonction *apprentissage()* reprend la fonction *calculPoids()* après avoir pris en compte les modifications nécessaires apportées à chaque classe.

Enfin, un ensemble est identifié par un nom et d'éventuels commentaires (*\_description*). On sauvegarde également le nom du fichier où sera enregistré l'ensemble, dans l'attribut *\_nomFichier*.

#### 2.2.2.2. Algorithme d'apprentissage

On reprend ici l'algorithme d'apprentissage de façon plus claire. Comme nous l'avons vu dans la description de chaque classe, différents calculs sont à effectuer à chaque mise à jour de l'ensemble, c'est-à-dire quand l'utilisateur va ajouter ou supprimer un exemple à une classe ou une classe à l'ensemble :

1. calcul du vecteur caractéristique de chaque exemple pour chaque classe
2. calcul du vecteur caractéristique moyen pour chaque classe
3. calcul de la matrice de variance-covariance pour chaque classe
4. calcul de la matrice moyenne de variance-covariance
5. calcul de la matrice inverse de variance-covariance
6. mise à jour des poids associés aux caractéristiques de chaque classe

```

fonction apprentissage()
  début
    pour c allant de 1 à _C faire

      pour e allant de 0 à _Ec faire
        _classes[c]._exemples[e].calculRubine() (1)
      fin pour
      _classes[c].calculRubineMoy() (2)
      _classes[c].calculVarCovar() (3)

    fin pour

    calculVarCovarMoy() (4)
    _varCovar.inverse() (5)

    pour c allant de 1 à _C
      _classes[c].calculPoids() (6)
    fin pour
  fin

```

Le fait de recalculer toutes les caractéristiques à chaque mise à jour de l'ensemble permettra de reconnaître dynamiquement les gestes au cours de l'apprentissage. En effet, le système va reconnaître en temps réel les formes apprises à partir du moment où il existe au moins une classe dans l'ensemble. Ainsi, l'utilisateur peut optimiser l'apprentissage en interagissant sur le système, c'est-à-dire qu'il peut affiner l'apprentissage en ajoutant des gestes exemples qui sont encore ambigus.

### 2.2.2.3. Enregistrement

#### 1. Enregistrement auxiliaire

La classe *CEnsemble* possède une fonction permettant de sauvegarder la valeur de tous les calculs effectués : vecteurs caractéristiques de Rubine pour chaque exemple, vecteur caractéristique moyen et matrice de variance-covariance pour chaque classe, matrice de variance-covariance moyenne et matrice inverse, ainsi qu'un vecteur de poids pour chaque classe.

La fonction *enregistrer()* conserve ces informations dans un fichier texte avec l'extension '.dat'. On lui donne le nom 'app\_' suivi du nom de l'ensemble de gestes utilisé (par exemples, 'app\_lettres.dat') et ce fichier est classé dans le répertoire 'Statistiques/Apprentissage/'. La structure d'un fichier de ce type est développée en *Annexe I*, accompagnée d'un exemple de fichier.

Le programmeur peut consulter ces données et ainsi comprendre plus facilement quand la reconnaissance échoue. C'est grâce à ce fichier que l'on a compris que le calcul de la matrice inverse n'était pas toujours possible et par conséquent faisait planter la reconnaissance (cf. paragraphe 2.4. Problèmes rencontrés).

#### 2. Enregistrement de l'application globale

L'ensemble des gestes est enregistré définitivement dans un unique fichier texte portant l'extension '.gst', situé par défaut dans le répertoire 'Gestes/'. On sauvegarde un à un les éléments indispensables des classes *CEnsemble*, *CClasse* et *CExemple* :

```

<nom de l'ensemble> ↵
<description de l'ensemble> ↵
<nombre de classe> ↵

pour chaque classe c de l'ensemble
  <nom de la classe c> ↵
  <description de la classe c> ↵
  <nombre d'exemples> ↵

  pour chaque exemple e de la classe c
    <nom de l'exemple e> ↵
    <nombre de coups de crayon> ↵

    pour chaque trait représentant l'exemple e
      <nombre de points du premier tracé> ↵
      liste des points du tracé (t = instant du point) :
        <x, y, t> ↵
    fin
  fin
fin

```

Remarque : les classes vides, seront éliminées de l'ensemble et ne seront donc pas enregistrées.

Pour éviter une surcharge inutile, les informations concernant les caractéristiques de Rubine, les matrices, les poids,... ne sont pas mémorisées. Lors de l'ouverture d'un fichier, on met à jour tous les champs de l'ensemble ouvert, puis, on applique l'algorithme d'apprentissage (méthode *apprentissage()* de la classe *CEnsemble*).

### 2.2.3. Reconnaissance

#### 2.2.3.1. *CAReconnaître*

Un geste à reconnaître est défini de la même manière qu'un geste exemple, avec néanmoins quelques informations supplémentaires concernant la reconnaissance. C'est pour cette raison que l'on utilise la classe *CAReconnaître* qui hérite de la classe *CExemple*.

Quand on crée une classe de ce type, on lui associe toujours l'ensemble de gestes à utiliser pour la reconnaissance (*\_ensemble*). La méthode *evaluation()* va calculer les fonctions d'évaluation de toutes les classes, le résultat étant placé dans le tableau dynamique *\_eval*. On peut ainsi déduire la fonction d'évaluation maximale et par conséquent, on peut déterminer à quelle classe appartient le geste dessiné. Dans ce sens, la méthode *maxEval()* met à jour les attributs *indice\_max* (l'indice de la classe possédant la fonction d'évaluation maximale), *\_max* (la valeur de cette fonction) et *\_cl* (la classe reconnue).

Avec cette méthode, tous les gestes sont reconnus. On introduit donc la notion de rejet pour éliminer les gestes ambigus. Comme nous l'avons vu dans la partie théorique (cf. chapitre II, paragraphe 5), deux méthodes peuvent être combinées. On peut tout d'abord calculer la probabilité que le geste appartienne effectivement à la classe reconnue (*\_proba*) et rejeter tous les gestes dont la probabilité est inférieure à un seuil. Cette technique est implémentée dans la fonction *rejetProbas()* qui met le booléen *\_rejetProbas* à 'vrai' si le geste est rejeté. Parallèlement, on calcule la distance de Mahalanobis (*\_delta2*) dans la fonction *\_rejetMahalanobis()* qui rejette également le geste quand *\_delta2* est inférieur à un seuil, tout en mettant à jour le booléen *\_rejetMahalanobis*.

Enfin, la fonction *reconnaissance()* regroupe tous ces calculs : après avoir trouvé la classe qui possède la plus grande fonction d'évaluation, elle cherche si le geste est ambigu.

#### 2.2.3.2. *Algorithme de reconnaissance*

L'algorithme de reconnaissance se déduit donc assez simplement :

1. calcul des fonctions d'évaluation de toutes les classes
2. recherche de la fonction d'évaluation maximale
3. rejet des gestes ambigus avec le calcul de probabilité
4. rejet des gestes ambigus avec la distance de Mahalanobis

```

fonction reconnaissance()
  début
    evaluation() (1)
    maxEval() (2)
    rejetProbas() (3)
    rejetMahalanobis() (4)
  fin

```

#### 2.2.3.3. *Enregistrement*

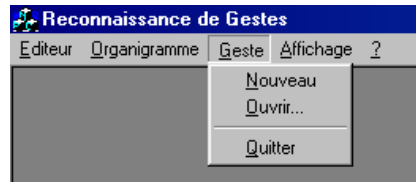
La classe *CAReconnaître* possède également une fonction d'enregistrement, appelée *enregistrer()*, permettant au programmeur de consulter facilement ces données et par la suite de faire d'éventuelles statistiques. On sauvegarde le nom de l'ensemble de gestes testé, le vecteur caractéristique du geste à reconnaître, la valeur de toutes les fonctions d'évaluation et enfin le nom du geste reconnu avec la valeur de la fonction d'évaluation maximum. Quand l'utilisateur souhaite faire des statistiques, on pourrait sauvegarder, en plus, si l'utilisateur est satisfait du résultat.

La fonction *enregistrer()* conserve ces informations dans un fichier texte avec l'extension '.dat'. On le nomme 'rec\_' suivi du nom de l'ensemble de gestes utilisé (par exemples, 'rec\_lettres.dat') et ce fichier est classé dans le répertoire 'Statistiques/Reconnaissance/'. La structure d'un fichier de ce type est développée en *Annexe II*, accompagnée d'un fichier exemple.

## 2.3. Application finale

### 2.3.1. Lancement de l'application

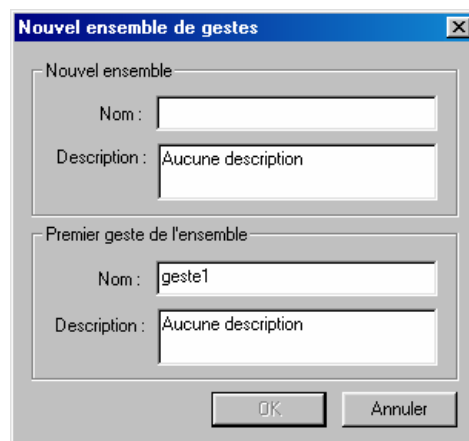
Pour utiliser le système d'apprentissage et de reconnaissance de gestes, sélectionner l'item **Geste** dans le menu principal du logiciel :



On peut choisir ensuite d'ouvrir un fichier existant ou d'en créer un nouveau.

#### 2.3.1.1. Nouveau

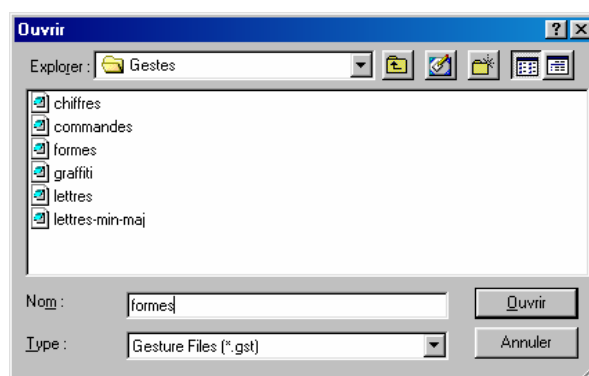
Si l'utilisateur choisit l'option **Nouveau**, une fenêtre s'ouvre lui demandant le nom et la description du nouvel ensemble, ainsi que le nom et la description de la première classe à créer :



Initialement, le champ contenant le nom de l'ensemble est vide, alors que les trois autres champs possèdent des valeurs par défaut. On peut valider ces informations en cliquant sur **OK** seulement quand les deux champs **Nom** sont remplis. Le système crée alors un nouvel ensemble avec une classe vide.

#### 2.3.1.2. Ouvrir

Par défaut, la fenêtre d'ouverture présente les fichiers '.gst' contenus dans le répertoire 'Gestes/'.

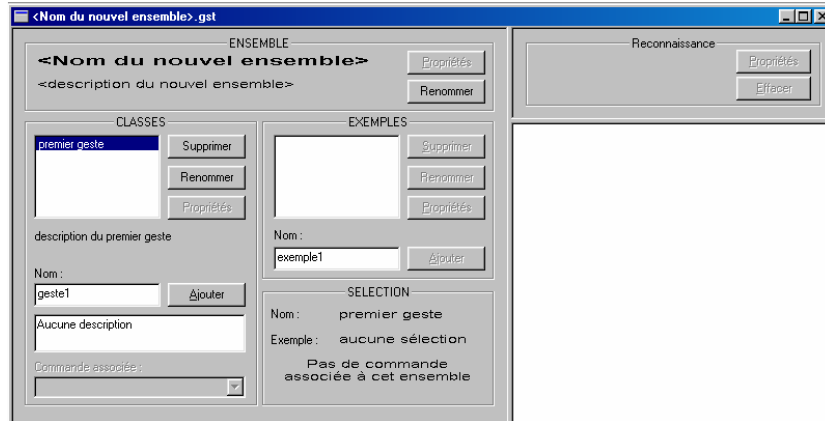


Sur clic du bouton **Ouvrir**, le système met à jour les attributs de l'ensemble ouvert en extrayant les informations du fichier texte. Puis, il applique l'algorithme d'apprentissage afin de compléter les variables concernant la reconnaissance (caractéristiques de Rubine, matrices...).

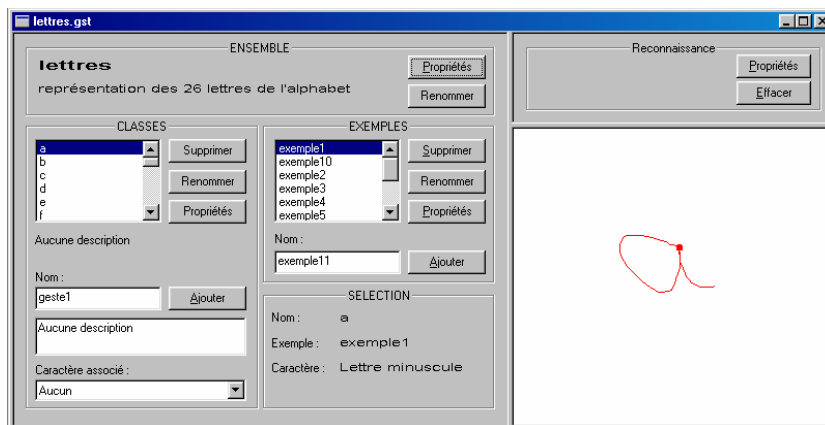
### 2.3.2. Fenêtre principale

Selon le choix de l'utilisateur (**N**ouveau ou **O**uvrir), une fenêtre principale s'affiche, soit avec un ensemble vide (1), soit avec un ensemble existant (2).

1



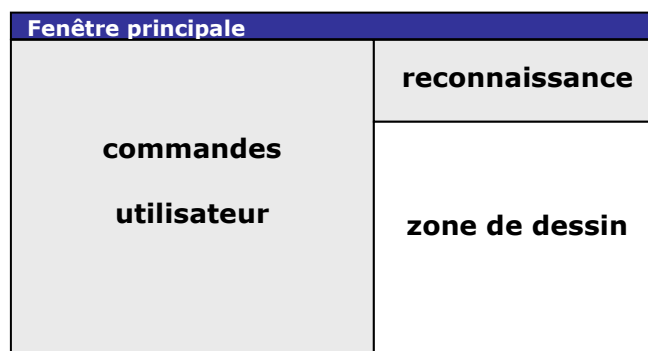
2



#### 2.3.2.1. Structure générale

Beaucoup d'informations sont à faire apparaître dans la fenêtre principale pour que l'application soit complète et la plus simple possible, sans passer par de multiples boîtes de dialogue. Aussi, la reconnaissance des gestes se fera au même endroit que leur apprentissage, à partir du moment où il existe au moins une classe dans la base.

La fenêtre principale est donc divisée en trois grandes parties :



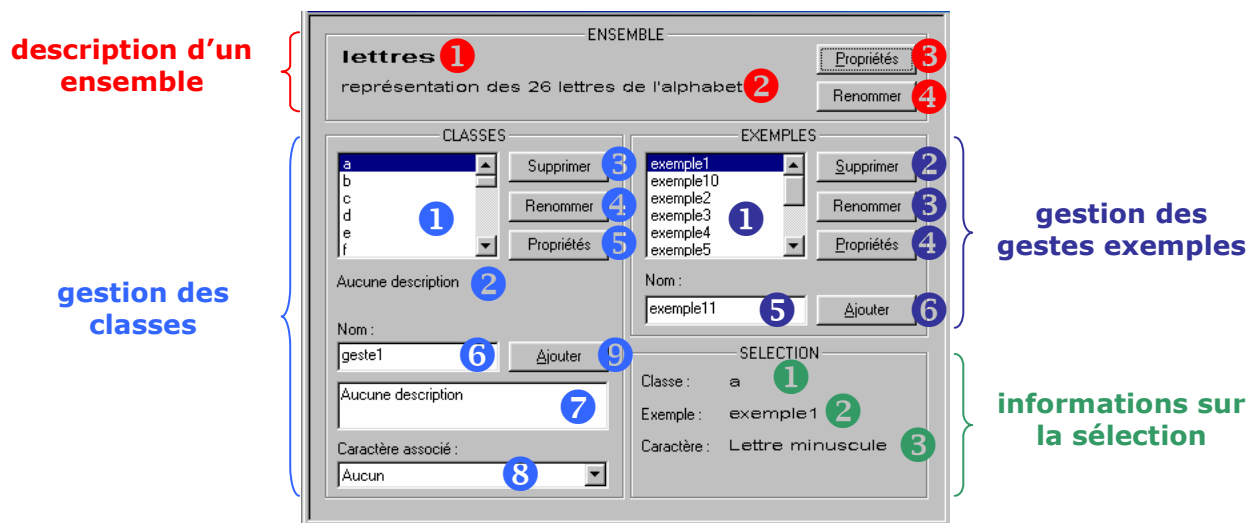
La partie blanche, sur la droite, est la zone de dessin. C'est ici que tous les gestes seront représentés graphiquement.

La zone de gauche est la fenêtre de commandes, là où l'utilisateur va pouvoir effectuer les traitements concernant l'apprentissage de gestes.

Enfin, au-dessus de la zone de dessin, se trouve les informations concernant la reconnaissance d'un geste.

### 2.3.2.2. Commandes utilisateur

Toutes les commandes nécessaires à l'apprentissage sont rassemblées dans cet emplacement :



Dans la partie supérieure, on retrouve la **description de l'ensemble**. La partie de gauche concerne les **classes** appartenant à l'ensemble et celle de droite, les **gestes exemples** associés. Enfin, dans le cadre inférieur droit, on dispose des informations liées à la **sélection**.

#### 1. Description d'un ensemble

Les données relatives à l'ensemble courant regroupent tout d'abord, son nom **1**, ainsi que son éventuelle description **2**. Le bouton **Propriétés** **3** permet d'afficher les propriétés générales de l'ensemble, c'est-à-dire la matrice de variance-covariance moyenne, sa matrice inverse et les poids associés aux caractéristiques de chaque classe. Le bouton **Renommer** **4** permet de modifier le nom et/ou la description de l'ensemble.

#### 2. Gestion des classes

Sur sélection d'une classe existante **1**, sa description, si elle existe, s'affiche en dessous de la liste **2**. Sur la droite, la liste des gestes exemples **1** se met à jour, affichant dans la zone de dessin le premier exemple de la classe.

#### ✓ Supprimer

Pour supprimer une classe existante, il faut sélectionner la classe en question dans la liste des classes **1** et cliquer tout simplement sur le bouton **Supprimer** **3**.

✓ **Renommer**

De la même façon, pour renommer une classe ou changer sa description, il faut également sélectionner la classe voulue ❶ et cliquer sur **Renommer** ❷. Après avoir indiqué le nouveau nom et la nouvelle description dans la boîte de dialogue réservée à cet effet, les modifications sont immédiatement prises en compte.

✓ **Propriétés**

Le bouton **Propriétés** ❸ permet d'afficher le vecteur caractéristique moyen de Rubine, la matrice de variance-covariance et le vecteur poids associés à la classe sélectionnée.

✓ **Ajouter**

Pour ajouter une classe, il suffit de remplir les champs concernant le nom ❹ et la description ❺ du nouveau geste, puis cliquer sur le bouton **Ajouter** ❻. La liste des classes ❶ est automatiquement mise à jour et la liste des exemples ❶ est réinitialisée. Il faudra ensuite ajouter des exemples à cette nouvelle classe.

### 3. Gestion des exemples

Sur sélection d'un geste existant ❶, le tracé correspondant est affiché dans la zone de dessin.

✓ **Supprimer**

De manière identique que pour la suppression d'une classe, sélectionner un exemple ❶ et cliquer sur le bouton **Supprimer** ❷.

✓ **Renommer**

Sélectionner un exemple ❶ et cliquer simplement sur le bouton **Renommer** ❸.

✓ **Propriétés**

Cliquer sur le bouton **Propriétés** ❹ pour consulter le vecteur des caractéristiques de Rubine associé au geste exemple sélectionné ❶. On a également des informations complémentaires concernant sa description graphique (nombre de points représentant la forme, coordonnées de quelques points, rectangle englobant).

✓ **Ajouter**

Tout d'abord, sélectionner la classe dans laquelle on souhaite ajouter l'exemple ❶. Ensuite, remplir le champ **Nom** ❺. Par défaut, sa valeur correspond à la chaîne 'exemple' concaténée à un indice. Enfin, dessiner le nouveau geste et cliquer sur le bouton **Ajouter** ❻.

### 4. Informations sur la sélection

Cette zone rappelle simplement le nom de la classe ❶ et celui de l'exemple ❷ sélectionnés (et la commande ❸ associée à l'édition d'organigrammes, cf. 3<sup>ème</sup> partie).

### 2.3.2.3. Zone de reconnaissance



Lorsque l'on dessine un geste, il est immédiatement analysé. On peut ainsi connaître la valeur du vecteur caractéristique de Rubine associé en cliquant sur le bouton **Propriétés**.

En cours d'apprentissage, les gestes sont reconnus de façon dynamique, à partir du moment où l'ensemble de classes est valide, c'est-à-dire qu'il existe au moins une classe dans l'ensemble. Le résultat est affiché dans la zone de reconnaissance. Si le geste est reconnu, le message indique 'Geste reconnu' en précisant le nom de la classe associée, sinon c'est 'Geste rejeté' qui apparaît.

Un dernier bouton, la commande **Effacer**, permet de réinitialiser la zone de dessin.

## 2.4. Problèmes rencontrés

Lors de l'implémentation de la méthode de Rubine, quelques points se sont avérés délicats et notamment le calcul de l'inverse de la matrice de variance-covariance. En effet, si une des caractéristiques d'un geste est identique pour tous les exemples, alors cela entraîne une matrice de variance-covariance avec des termes nuls. Par conséquent, il est impossible d'en déduire la matrice inverse utilisée pour trouver les poids de chaque classe. C'est le cas par exemple pour le cosinus et le sinus de l'angle initial des lignes horizontales ou verticales.

On a également le cas particulier où l'utilisateur ajouterait toujours le même exemple à une classe. Cette fois, la matrice de variance-covariance est totalement nulle (on fait la différence terme à terme de deux matrices identiques) et donc la matrice inverse ne peut pas être calculée.

Pour que la reconnaissance soit optimale, plusieurs points seront donc à respecter. Il faudra éviter d'ajouter deux fois le même geste exemple à l'ensemble et une classe devra contenir plus d'un exemple. Un ensemble comprenant une vingtaine d'exemples par classe marche plutôt bien dans la pratique.

## 2.5. Résultats

Dans son article, Rubine a testé son algorithme sur dix ensembles de gestes de taille différente. Pour les ensembles inférieurs à 15 classes avec une quinzaine d'exemples chacune, plus de 98 % des formes sont reconnues correctement. Les ensembles avec 30 classes comprenant chacune 40 exemples ont un taux de reconnaissance de 97 %. Ce taux descend seulement à 96 % quand on diminue le nombre d'exemples à 15 pour ce même ensemble de 30 classes.

Ces résultats, très satisfaisants, ont été confirmés par l'utilisation de notre système de reconnaissance et d'apprentissage.

En comparaison avec l'algorithme de Jorge et Fonseca, la méthode de Rubine permet de reconnaître beaucoup plus de gestes. En effet, l'ensemble de gestes utilisé dans la première méthode est fixe : ni l'utilisateur, ni même le programmeur ne peuvent le modifier. La technique de Rubine est beaucoup plus souple, donnant l'opportunité d'apprendre plusieurs ensembles de gestes.

Toutefois, on peut émettre quelques réserves. Dans la méthode de Jorge et Fonseca, la reconnaissance avait lieu grâce à des rapports de distances, ce qui permettait à l'utilisateur de dessiner des formes sans se soucier de leur taille ni de leur orientation. A ce niveau, la méthode de Rubine est légèrement plus stricte dans le sens où elle prend en compte des distances et des angles (et non des rapports). Ces propriétés impliquent que les gestes soient tracés selon l'orientation dans laquelle ils ont été appris et en fonction de leur taille initiale. Ces caractéristiques peuvent être contournées en apprenant au système des exemples de taille différente.

### 2.6. Améliorations

Pour l'instant, les résultats concernant la reconnaissance sont calculés manuellement. Il serait donc intéressant d'enregistrer les données statistiques dans un fichier. Après la création d'un ensemble de gestes, il faudrait évaluer le taux de bonne reconnaissance : dessiner plusieurs gestes à la suite en indiquant à chaque fois, au système, si la forme est reconnue correctement.

### 3. Editeur d'organigrammes

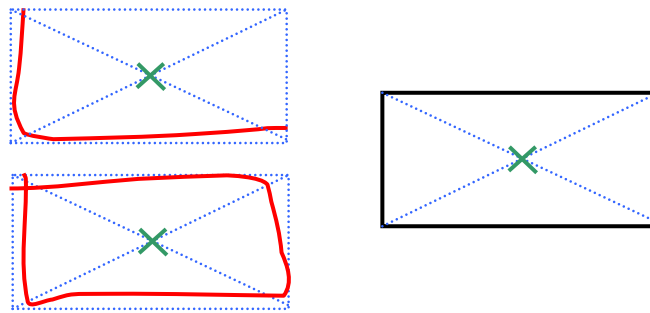
Cette application est totalement liée au système d'apprentissage et de reconnaissance de gestes. Ici, un geste reconnu est automatiquement transformé en une forme géométrique. Les formes utilisées sont les objets nécessaires à la construction d'organigrammes : rectangles, carrés, cercles, ellipses, losanges, triangles, lignes et flèches sont disponibles pour l'instant. De plus, on associe un ensemble de commandes gestuelles permettant de modifier les composants de l'organigramme.

#### 3.1. Présentation

##### 3.1.1. Association geste-commande

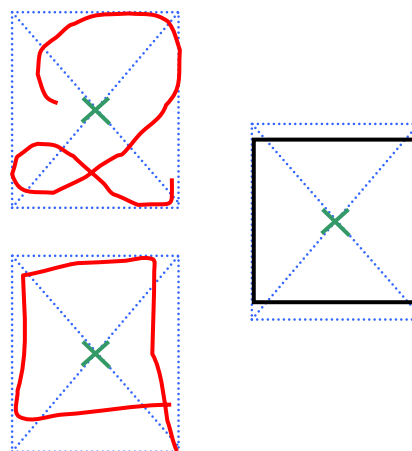
Dans un premier temps, l'utilisateur doit attacher une action aux gestes dessinés lors de l'apprentissage.

Pour la création de formes géométriques, on souligne le fait que les gestes appris peuvent ressembler sensiblement aux tracés finaux, mais peuvent également être totalement différents. Par exemple, on peut choisir de dessiner la lettre 'L' pour représenter un rectangle, ce qui semble légèrement plus rapide que le tracé complet du rectangle :



*légende : en rouge : les gestes initiaux associés aux dessins d'un rectangle (lettre 'L' et rectangle)  
 en bleu : les rectangles englobants des gestes initiaux  
 en vert : le centre des rectangles englobants  
 en noir : le rectangle final*

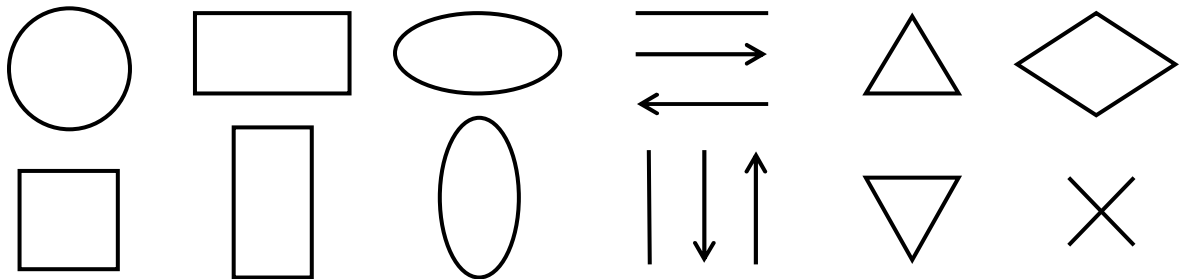
Cette solution offre une grande souplesse à l'utilisateur : il choisit vraiment les gestes qu'il veut, ceux qui sont pour lui les plus faciles à mémoriser ou les plus pratiques. De plus, grâce à cette méthode, on peut écarter le problème des gestes ambigus. Par exemple, les tracés de deux gestes tels un rectangle et un carré sont difficiles à différencier. On peut alors choisir d'utiliser un autre symbole (ici le chiffre '2') pour représenter le carré (par exemple), ce qui lève définitivement l'ambiguïté :



*légende : en rouge : les gestes initiaux associés aux dessins d'un carré (chiffre '2' et carré)  
 en bleu : les rectangles englobants des gestes initiaux  
 en vert : le centre des rectangles englobants  
 en noir : le carré final*

### 3.1.2. Construction des formes

Les formes choisies ne possèdent qu'une seule orientation, c'est-à-dire qu'elles sont alignées horizontalement et verticalement, de la façon suivante :



On voit bien ici qu'aucune forme n'est dessinée en biais.

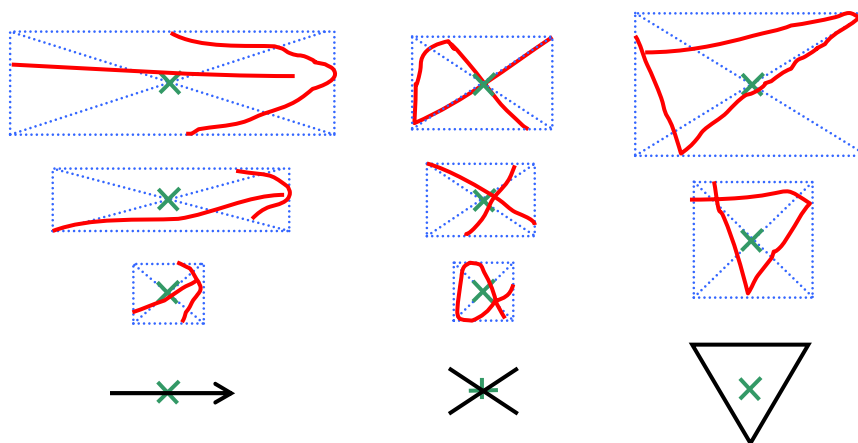
Deux options sont proposées au concepteur d'organigrammes concernant la taille des éléments :

- OPTION 1 : les formes sont de taille fixe ;
- OPTION 2 : les formes sont incluses dans le rectangle englobant du geste initial.

#### 3.1.2.1. Taille fixe

Au départ, les dimensions de tous éléments sont définies par défaut, mais l'utilisateur peut les changer à tout moment.

Dans ce cas, lors de la reconstruction, chaque forme est centrée par rapport au centre du rectangle englobant. Prenons, par exemple, le cas d'une flèche, d'une croix et d'un triangle avec la pointe vers le bas. Quelque soit la taille du geste initial, la forme possède une taille prédéfinie et donc est toujours dessinée de la même façon, toujours centrée par rapport au rectangle englobant initial :

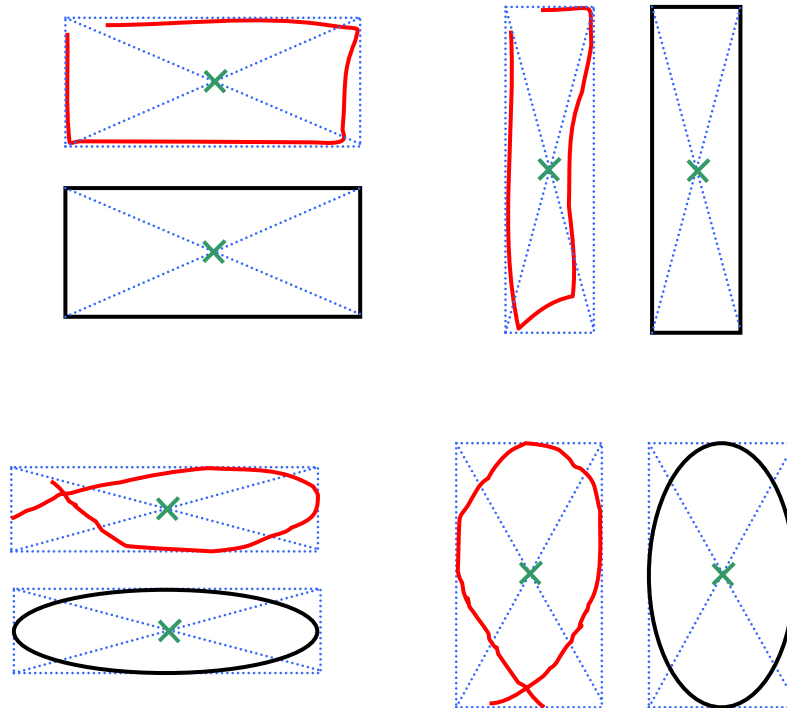


légende : en rouge : les gestes initiaux associés aux dessins d'une flèche, d'une croix et d'un triangle  
 en bleu : les rectangles englobants des gestes initiaux  
 en vert : le centre des rectangles englobants (et donc des formes finales)  
 en noir : les formes redessinées (flèche, croix et triangle) avec une taille fixe prédéfinie

### 3.1.2.2. Taille variable

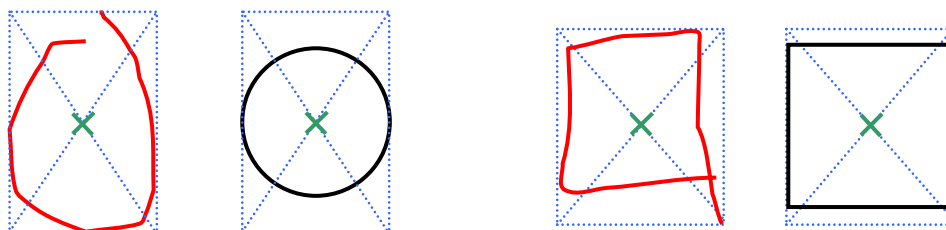
Cette option semble la plus intéressante. Les formes redessinées sont incluses dans le rectangle englobant du geste initial.

Dans l'exemple suivant, on expose le cas de formes simples, comme les rectangles et les ellipses, où aucun calcul n'est nécessaire pour la construction (les coordonnées du rectangle englobant suffisent) :



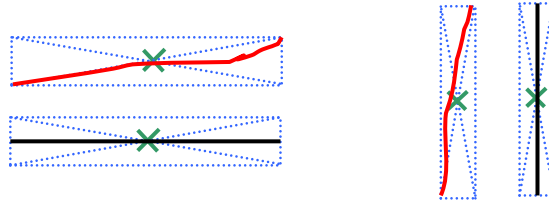
légende : en rouge : le geste initial  
en bleu : le rectangle englobant du geste initial  
en vert : le centre du rectangle englobant  
en noir : la forme finale

Pour que les formes soient toujours comprises dans le rectangle englobant du geste initial, il est nécessaire de recentrer les cercles et les carrés. Par conséquent, le rayon des cercles et la largeur des carrés correspondent à la dimension la plus petite entre la largeur et la hauteur du rectangle englobant, comme ci-dessous :

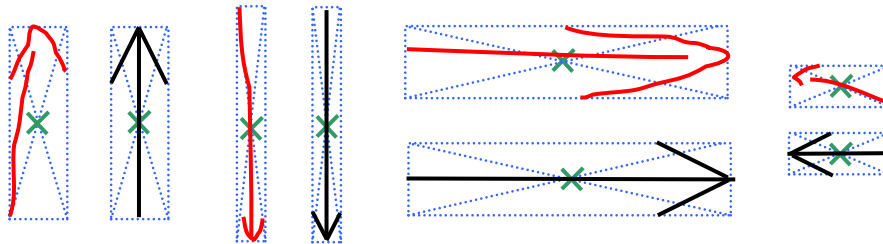


légende : en rouge : les gestes initiaux associés aux dessins d'un cercle et d'un carré  
en bleu : les rectangles englobants des gestes initiaux  
en vert : le centre des rectangles englobants (équivalent au centre des formes redessinées)  
en noir : le cercle et le carré redessinés

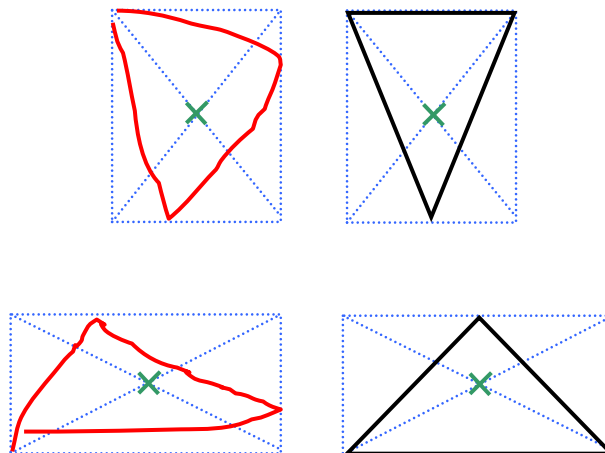
Les types de lignes disponibles pour la construction d'organigrammes sont les lignes verticales ou horizontales. Aussi, à chaque fois qu'un geste de ce type est reconnu, on le remplace par une ligne située au cœur de son rectangle englobant :



Pour les flèches, on applique le même principe. La partie principale de la flèche est tracée au centre du rectangle englobant et la pointe de la flèche est construite en fonction de la hauteur du rectangle englobant (ou de sa largeur si la flèche est orientée verticalement). Avec les deux orientations et les deux directions principales, on fournit quatre types de flèches :



On propose deux styles de triangles : ceux avec la pointe vers le bas et ceux avec la pointe vers le haut, représentés comme ceci :



### 3.1.3. Commandes utilisateur

Afin d'éviter au maximum l'utilisation de l'interface clavier ou de menus déroulants, l'éditeur d'organigrammes propose également d'exécuter des commandes sur les composants de l'organigramme, toujours grâce à la reconnaissance de gestes graphiques.

Pour l'instant, les commandes qui nous semblent indispensables, ont été développées :

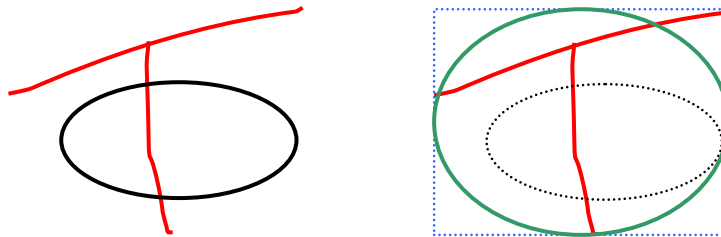
- changement de couleur ;
- suppression ;
- modification de taille ;
- déplacement.

Cette liste pourrait être étoffée dans de futurs travaux.

Les notions de changement de couleur ou de suppression sont assez simples. Par conséquent, on détaillera simplement comment les formes sont agrandies, rétrécie ou déplacée.

Pour la taille, on propose deux solutions :

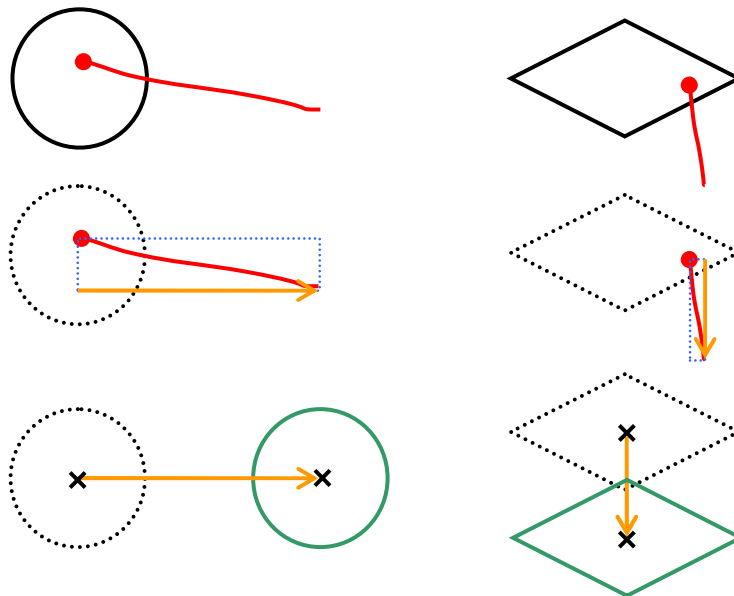
- agrandissement ou réduction selon un nombre de pixels fixe ;
- modification de la taille en fonction de la taille du rectangle englobant du nouveau geste tracé.  
Dans l'exemple suivant, on cherche à modifier la taille d'une ellipse :



légende : en noir : la forme initiale  
 en rouge : la commande de modification de taille  
 en bleu : le rectangle englobant de la commande gestuelle  
 en vert : la forme avec sa nouvelle taille

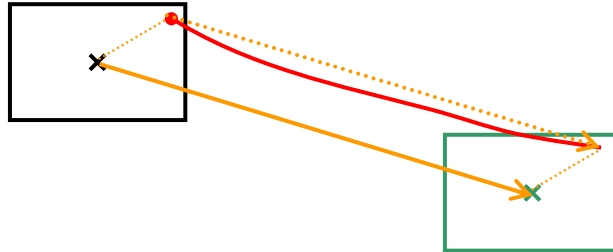
Deux types de déplacement ont également été choisis :

- les déplacements dits « stricts ». Dans ce cas, on translate la forme selon un vecteur horizontal ou vertical de la taille du rectangle englobant de la commande gestuelle tracée.  
Dans l'exemple suivant, les gestes associés aux commandes de déplacement sont des lignes tracées selon l'orientation du vecteur de translation (de gauche à droite pour une translation horizontale vers la droite, de haut en bas pour une translation verticale vers le bas, etc.). On déplace ici deux formes représentant un cercle et un losange :



légende : en noir : la forme en position initiale  
 en rouge : la commande gestuelle de déplacement (• : point de départ)  
 en bleu : le rectangle englobant du geste  
 en orange : le vecteur de translation horizontal ou vertical  
 en vert : la forme traduite

- les déplacements « variables ». Cette option est plus souple et permet d'effectuer un déplacement dans n'importe quelle direction. Cette fois, le vecteur de translation est représenté par la distance entre le premier et le dernier point du geste.



*légende : en noir : la forme en position initiale  
en rouge : la commande gestuelle de déplacement (• : point de départ)  
en orange : le vecteur de translation  
en noir : la forme translatée*

### 3.2. Programmation

Nous allons voir dans cette partie, les points importants liés à l'implémentation de l'éditeur d'organigramme et en particulier, comment les formes sont-elles représentées.

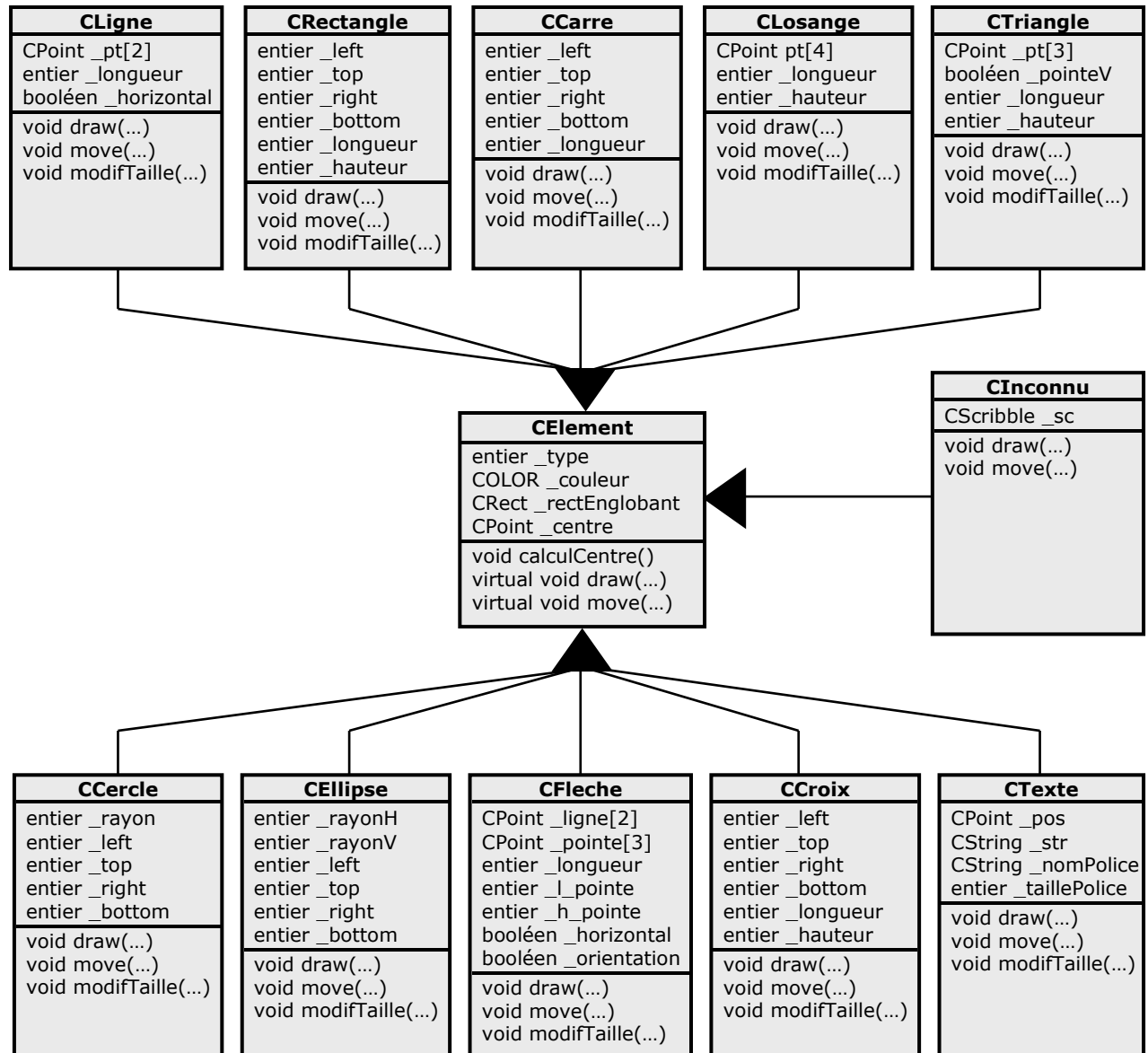
Au préalable, il a été nécessaire de modifier légèrement la modélisation de la première application (système d'apprentissage et de reconnaissance de gestes). Pour associer une commande à un geste, on ajoute un attribut `_commande` à la classe `CClasse`.

Pour l'instant, deux modèles d'ensembles de gestes sont disponibles dans l'éditeur d'organigramme : le tracé des formes et les commandes de traitement. Plus tard, on pourrait également utiliser un ensemble de gestes permettant d'écrire du texte, ce qui est pour l'instant possible uniquement avec l'interface clavier.

Dans cet optique, on a trouvé judicieux d'ajouter l'attribut `_type` à la classe `CEnsemble`. Cet entier prendra les valeurs :

- 0 ↔ aucune commande associée (apprentissage de gestes simples) ;
- 1 ↔ tracé des formes, des composants d'un organigramme ;
- 2 ↔ commandes de traitement des formes ;
- 3 ↔ insertion de texte, de caractères.

### 3.2.1. Diagramme de classes



légende : les triangles expriment la notion d'héritage.

### 3.2.2. Description des classes

#### 3.2.2.1. CElement

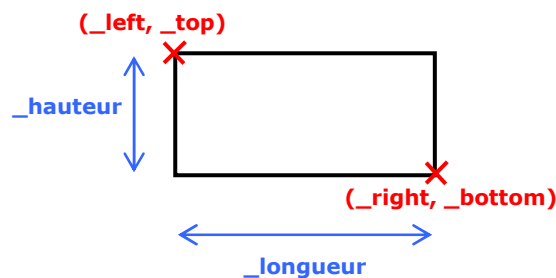
On regroupe les attributs communs à tous les objets (le type, la couleur, le rectangle englobant ainsi que son centre calculé avec la fonction *calculCentre()*) dans la classe appelée *CElement*. Chaque forme possède sa propre classe héritant de *CElement*.

### 3.2.2.2. CLigne

Pour l'instant, une ligne ne peut être qu'horizontale ou verticale : on utilise le booléen *\_horizontal* pour différencier ces deux types. Cette contrainte peut être facilement modifiée par le programmeur qui rajoutera les attributs nécessaires pour des lignes obliques (utilisation d'un entier au lieu d'un booléen...). Une ligne est alors définie par ses deux extrémités et un attribut permettant de sauvegarder sa longueur.

### 3.2.2.3. CRectangle

De la même façon que pour une ligne, un rectangle ne peut être dessiné que horizontalement ou verticalement, le critère de rotation n'étant pas nécessaire pour la construction d'organigrammes. Un rectangle est donc caractérisé par quatre entiers représentant les coordonnées des sommets : le point de coordonnées (*\_left*, *\_top*) représentera le sommet en haut à gauche et celui de coordonnées (*\_right*, *\_bottom*) sera le sommet en bas à droite. On sauvegarde, en plus, la longueur et la hauteur du rectangle.

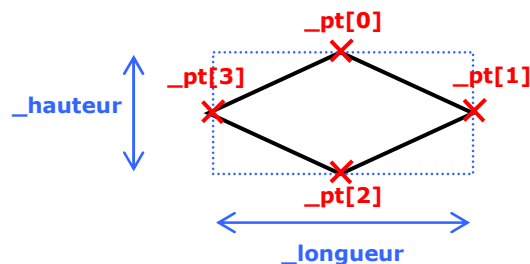


### 3.2.2.4. CCarre

Un carré se différencie d'un rectangle simplement par le fait qu'il ne possède pas d'attribut *\_hauteur*, puisque sa hauteur est identique à sa longueur. Pour simplifier cette modélisation, on pourrait regrouper les carrés et les rectangles.

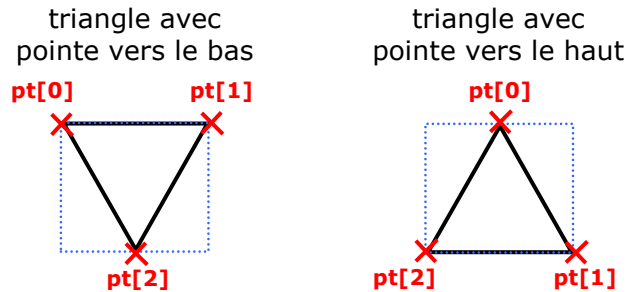
### 3.2.2.5. CLosange

On définit un losange à l'aide de ses quatre sommets (*\_pt[4]*), ces points étant toujours placés au milieu des côtés du rectangle englobant. De plus, comme pour les autres formes, on retient sa hauteur et sa longueur dans les attributs réservés à cet effet.



3.2.2.6. *CTriangle*

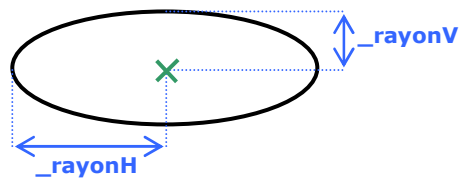
Il y a deux façons de représenter un triangle : soit avec la pointe vers le bas, soit vers le haut. Pour cette raison, on utilise le booléen `_pointeV` qui prend la valeur 'vrai' quand le triangle a sa pointe vers le bas (pointe en 'V') et 'faux' dans l'autre cas. Un triangle est évidemment défini par trois points. Ces derniers possèdent toujours la même disposition selon l'orientation du triangle :

3.2.2.7. *CCercle*

Un cercle est défini de la même manière qu'un carré, avec les quatre extrémités du rectangle englobant. On remarque que l'on sauvegarde ici le rayon du cercle et non son diamètre.

3.2.2.8. *CEllipse*

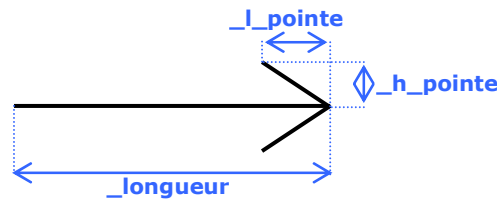
Une ellipse est également définie à l'aide des coordonnées du rectangle englobant. Les attributs `_rayonH` et `_rayonV` correspondent respectivement aux rayons horizontal et vertical :

3.2.2.9. *CFleche*

Une flèche possède deux booléens permettant de distinguer l'orientation (horizontale ou verticale) et la direction de la flèche (droite-gauche / haut-bas).



L'attribut *\_longueur* correspond à la taille de la partie centrale de la flèche, *\_h\_pointe* et *\_l\_pointe* sont la hauteur et la longueur de la pointe :



Les points *\_ligne[0]* et *\_ligne[1]* coïncident avec les extrémités de la ligne principale et les trois autres (*\_pt[0]*, *\_pt[1]* et *\_pt[2]*) décrivent la pointe.

### 3.2.2.10. CCroix

Une croix est définie avec les mêmes attributs qu'un rectangle. La différence se situe simplement dans le tracé de la forme : une croix correspond aux diagonales du rectangle.

### 3.2.2.11. CTexte

Cet objet va permettre d'ajouter du texte pendant l'édition d'un organigramme. La chaîne de caractères entrée par l'utilisateur est mémorisée dans la variable *\_str*. Elle sera insérée à la position indiquée par l'attribut *\_pos*, qui prend les coordonnées du point supérieur gauche du rectangle englobant. Pour pouvoir mettre en forme le texte, on utilise, en plus, deux attributs concernant sa taille et sa police.

### 3.2.2.12. CInconnu

Cette classe permet d'enregistrer les gestes qui ne sont pas reconnus par le système de reconnaissance. On utilise pour cela un attribut de type *CScribble* qui contient la liste de tous les points du geste. L'utilisateur peut choisir de garder le tracé de la forme ou alors de l'effacer et donc de recommencer un autre geste.

## 3.2.3. Les méthodes

Les méthodes *draw(...)* et *move(...)* ont la même en-tête pour tous les éléments. *draw(...)* permet de dessiner les formes et *move(CSize translation)* déplace les éléments selon le vecteur de translation passé en paramètre.

Tous les éléments, à l'exception de la classe *CInconnu*, possèdent, en plus, une méthode permettant de modifier leur taille (*modifTaille()*) : agrandissement ou réduction des formes géométriques ou de la taille des caractères pour l'élément textuel.

## 3.2.4. Algorithme

Le principal problème étant de discriminer les formes de l'organigramme avec les commandes utilisateur, on a décidé, pour l'instant, de dessiner les formes avec le bouton gauche de la souris et les commandes avec le bouton droit.

Le déroulement de l'algorithme est simple. A chaque fois que l'on dessine un geste avec le bouton gauche, on crée un nouvel élément que l'on ajoute à la liste d'éléments du document courant. Après chaque modification, tous les objets de la liste sont redessinés grâce aux fonctions *draw()*.

Ensuite, quand une commande est reconnue (geste tracé avec le bouton droit de la souris), on exécute le traitement associé à la forme sélectionnée ou à l'ensemble du document.

### 3.2.5. Enregistrement

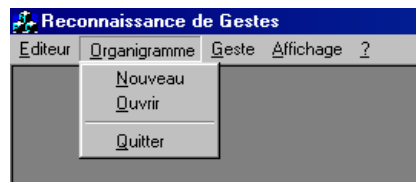
Un organigramme peut être sauvegardé par l'utilisateur, dans un fichier possédant l'extension '.org', dans le répertoire 'Organigrammes/' par défaut.

Pour l'enregistrement d'un fichier de ce type, on a utilisé la technique de sérialisation de Microsoft Visual Studio. Cette méthode est assez simple quand les classes sont parfaitement définies, ce qui est le cas ici. Chaque objet possède une méthode appelée *Serialize()* qui permet de sauvegarder les données utiles. On met de côté les dimensions pour ne garder que les coordonnées des points nécessaires à la reconstruction des formes.

## 3.3. Application finale

### 3.3.1. Lancement de l'application

Dans le menu principal du logiciel, sélectionner l'item **Organigramme** :



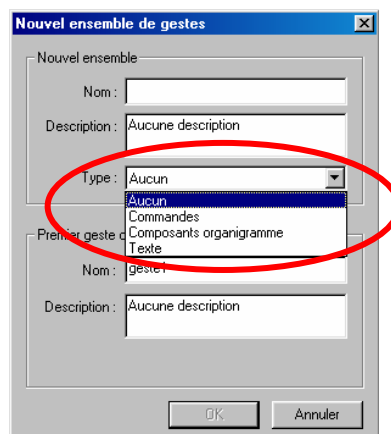
On peut choisir ensuite de créer un nouvel organigramme ou d'en ouvrir un existant. Les deux possibilités mènent à la même fenêtre principale : une zone de dessin vierge où l'utilisateur pourra éditer des organigrammes, ou, une zone de dessin comprenant les composants de l'application ouverte.

### 3.3.2. Association geste-commande

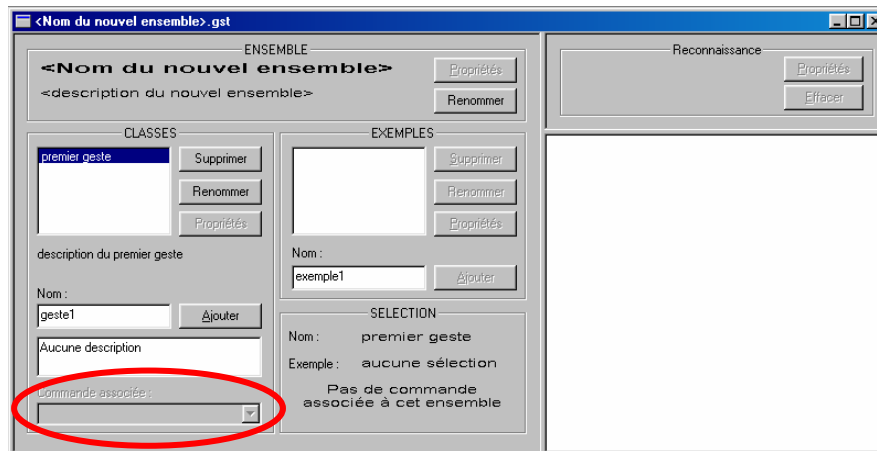
#### 3.3.2.1. Modifications du système d'apprentissage et de reconnaissance de gestes

Les ensembles de gestes nommés 'formes.gst' et 'commandes.gst' existent dans le répertoire 'Gestes'. Par défaut, ce sont ces ensembles qui se chargeront lors de la création d'un nouvel organigramme. Mais l'utilisateur a tout intérêt à créer ses propres gestes pour avoir une reconnaissance optimale. Dans ce cas, se reporter au paragraphe sur le système d'apprentissage et de reconnaissance de gestes. Le seul détail supplémentaire à prendre en compte est l'association d'actions utilisateur aux gestes appris.

Lors de la création d'un nouvel ensemble de geste, choisir tout d'abord le type d'actions à associer ('Aucun', 'Commandes', 'Composants organigramme' ou 'Texte') :

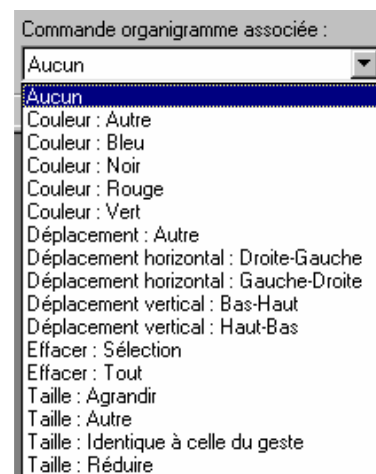
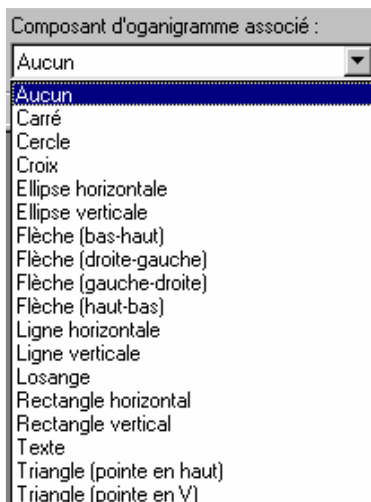


Ensuite, dans la fenêtre principale, l'utilisateur va disposer d'une liste déroulante contenant la liste des actions disponibles dans l'éditeur d'organigramme :



Dans l'exemple ci-dessus, le type d'ensemble choisi est 'Aucun', par conséquent, on ne peut pas associer d'actions aux gestes qui seront appris et donc la liste est grisée.

Dans les autres cas, il existe un menu déroulant pour chaque type d'ensemble. On ne parle ici que des composants de l'organigramme (liste de gauche) et des commandes (liste de droite), l'insertion de texte n'étant pas développée pour l'instant :



### Remarques sur la liste des composants :

L'option 'Aucune' indique que le geste n'est pour l'instant associé à aucune forme. Les autres items correspondent aux composants d'un organigramme. On souligne que les formes horizontales et verticales sont distinctes, offrant ainsi une plus grande souplesse de création.

Le cas particulier de l'élément 'Texte' va permettre d'insérer du texte dans l'organigramme (cf. paragraphe 3.3.7.).

### Remarques sur la liste des commandes :

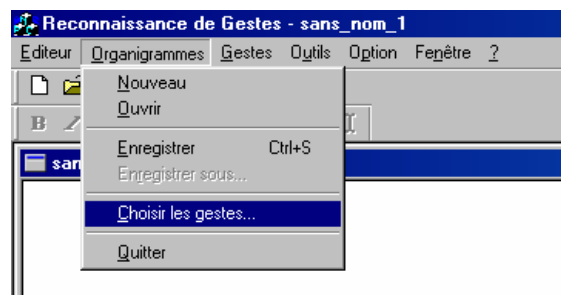
L'option 'Aucune', comme dans la liste précédente, permet de dessiner des gestes sans pour autant leur associer une commande. On remarque quatre types de commandes bien distinctes :

- modifier la couleur d'une forme : 'Couleur : xx' ;
- déplacer une forme : 'Déplacement : xx' ;
- modification de la taille d'une forme : 'Taille : xx' ;
- suppression d'une forme ('Effacer : Sélection') ou de toutes les formes ('Effacer : Tout').

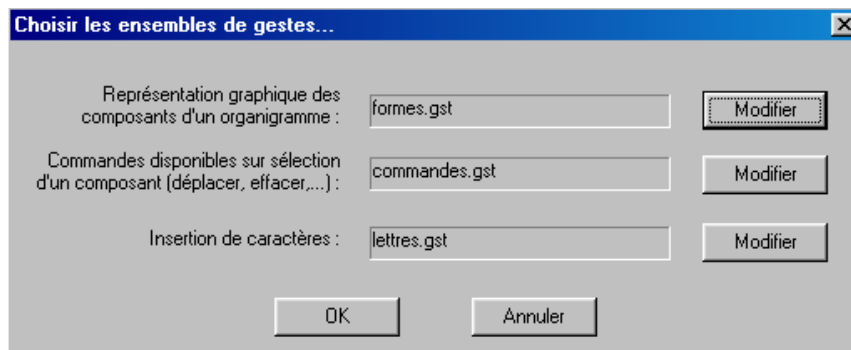
Ces listes ne sont pas exhaustives, un programmeur pourrait aisément les étoffer.

### 3.3.2.2. Choix des ensembles associés à l'organigramme

Pour modifier le nom des fichiers associés aux ensembles de gestes, sélectionner l'item **Organigrammes / Choisir les gestes...** dans le menu principal :



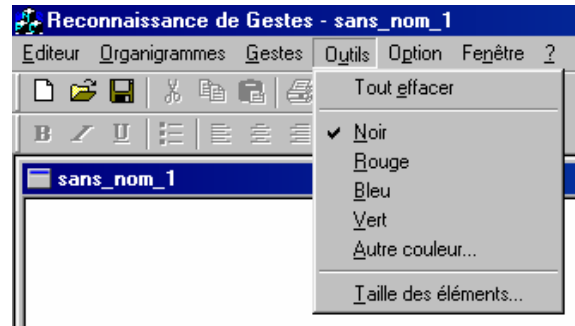
La fenêtre attachée propose à l'utilisateur le nom, par défaut, des fichiers '.gst' associés aux trois types d'ensembles disponibles (le troisième ensemble n'étant pas utilisé pour l'instant) :



Avec les commandes **Modifier**, on peut changer le nom des fichiers : le système présente alors la liste de tous les ensembles contenus dans le répertoire 'Gestes/'.

### 3.3.3. Fonctionnalités du menu

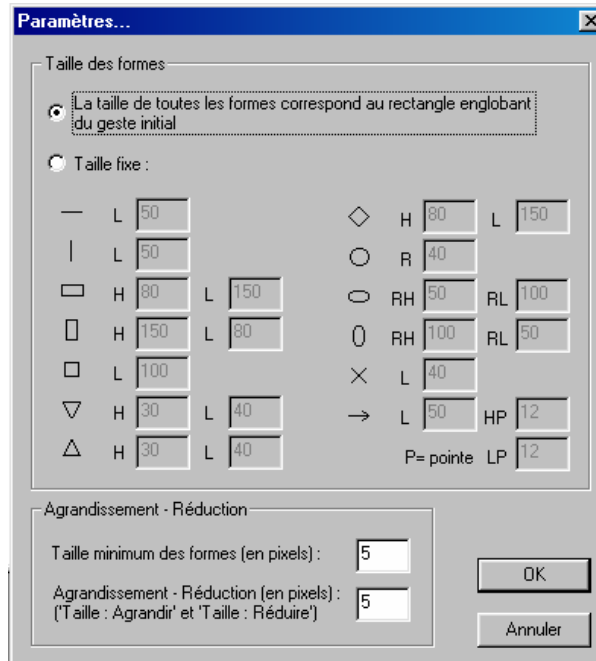
Plusieurs fonctions sont proposées à l'utilisateur par l'intermédiaire du menu **Outils** :



L'option **Tout effacer** initialise la zone de dessin en effaçant toutes les formes présentes.

La deuxième partie de ce menu permet de modifier la couleur courante du document (**Noir** par défaut). L'item **Autre couleur...** affiche une boîte de dialogue avec toutes les couleurs disponibles.

La rubrique **Taille des éléments...** permet de régler quelques paramètres associés à la taille générale des éléments :



Dans le premier cadre, deux options concernant la taille des éléments sont proposées à l'utilisateur :

- soit la taille des objets est définie par le rectangle englobant du geste initial (première option, choisie par défaut) ;
- soit la taille des composants est fixe (deuxième option). Dans ce cas, l'utilisateur dispose de la liste de tous les éléments accompagnés de leur taille.

On peut noter la signification des initiales utilisées :

§ L	↔	longueur
§ H	↔	hauteur
§ R	↔	rayon d'un cercle
§ RH et RV	↔	rayons horizontal et vertical d'une ellipse
§ HP et LP	↔	hauteur et longueur de la pointe d'une flèche

Dans l'exemple précédent, on peut voir les valeurs par défaut définies par le programmeur : 50 pour la longueur d'une ligne horizontale, 50 pour la hauteur d'une ligne verticale,...

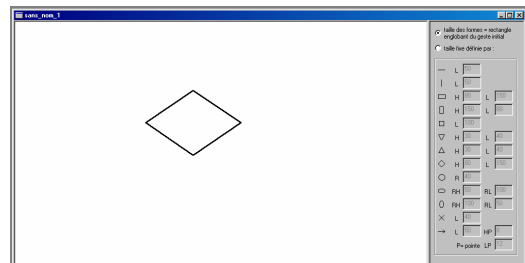
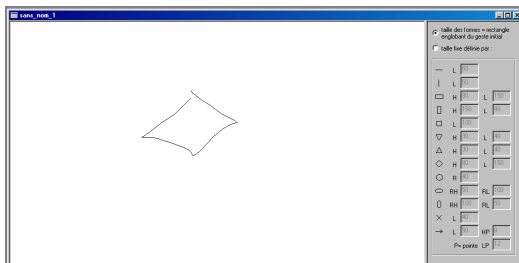
Le deuxième cadre permet de modifier les paramètres nécessaires pour l'agrandissement ou la réduction des formes (cf. 3.3.6.3.1).

### 3.3.4. Création d'organigrammes

L'application est prête à recevoir des gestes utilisateur.

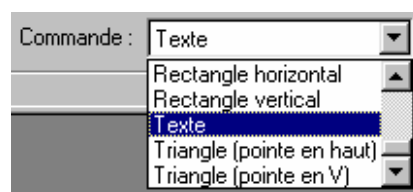
#### 3.3.4.1. Composants graphiques

Pour représenter une forme, il suffit de tenir le bouton gauche de la souris enfoncé lors du tracé ou utiliser simplement le stylet de la tablette graphique. Dans l'exemple suivant, à gauche, se trouve le geste associé au dessin d'un losange et à droite la forme géométrique finale :

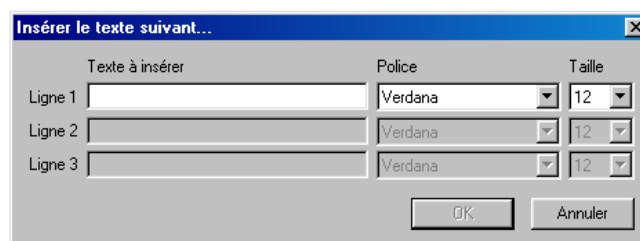


#### 3.3.4.2. Insertion de texte

Un objet particulier indispensable à l'élaboration d'organigrammes est l'insertion de chaînes de caractères. Tout d'abord, dans l'interface concernant l'apprentissage de gestes, il faut associer le geste voulu à la commande intitulée 'Texte' :

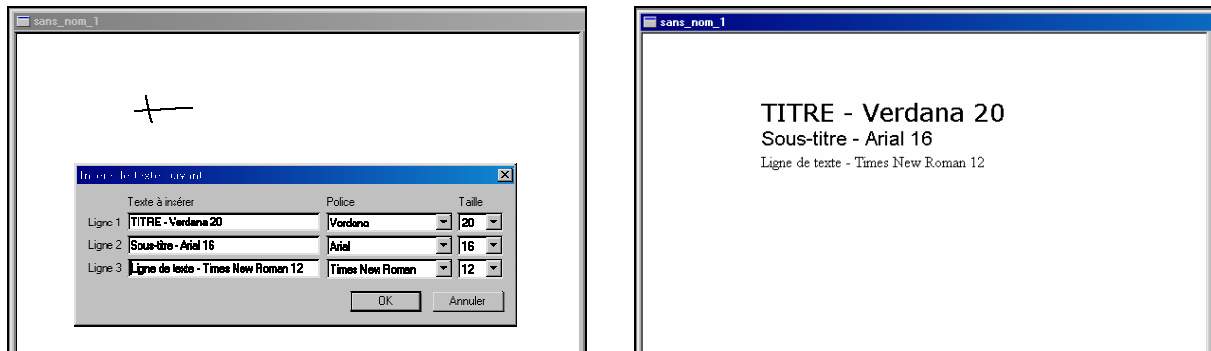


Ensuite, dans l'éditeur d'organigrammes, l'utilisateur trace le geste à l'endroit où il souhaite insérer du texte. Après avoir correctement reconnu le geste, le système affiche une fenêtre d'insertion du type :



L'utilisateur a la possibilité d'écrire jusqu'à trois lignes de texte, en choisissant pour chacune une police et une taille de caractères dans les deux listes déroulantes. La police par défaut est Verdana en 12 points.

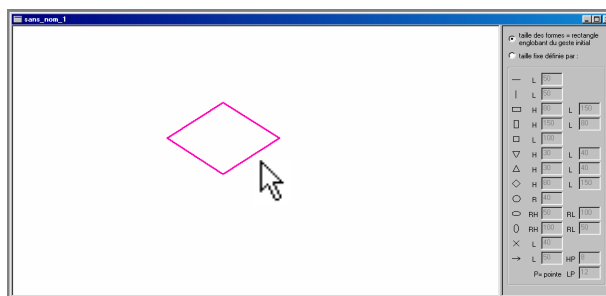
Dans l'exemple suivant, le geste associé à l'insertion de texte est le symbole '+' :



Le texte inséré est positionné par rapport au point supérieur gauche du rectangle englobant du geste initial.

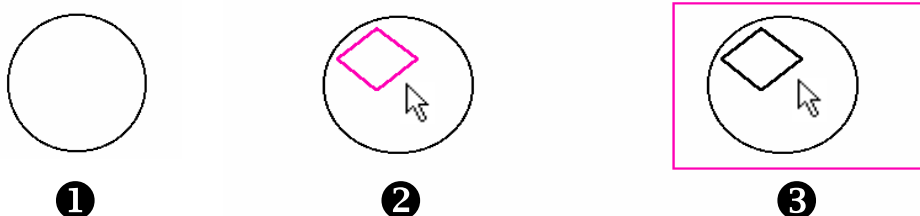
## 3.3.5. Sélection

On peut sélectionner un élément en déplaçant la souris (ou le stylet) dans le rectangle englobant de la forme. La sélection est signifiée par un changement de couleur :



Le principe de sélection utilisé semble le plus approprié mais pose tout de même quelques contraintes. Quand la souris est en mouvement, on analyse sa position : on cherche si les coordonnées du pointeur appartiennent au rectangle englobant d'une forme. Pour cela, on parcourt la liste des formes de la plus récente à la plus ancienne. L'inconvénient est que, quand une forme est incluse dans une autre, on ne peut la sélectionner que si elle a été dessinée après la forme qui l'entoure. Il faudrait approfondir ce sujet pour avoir une sélection encore plus pointue, mais cela semble assez difficile avec la structure des formes que nous utilisons. En effet, on ne possède pas tous les points relatifs à une forme...

Ce problème est illustré ci-dessous :



Dans un premier temps, l'utilisateur a dessiné un cercle (1). Ensuite, à l'intérieur de ce premier composant, il insère un losange (2). Dans ce cas, il n'y a pas de difficultés pour sélectionner le losange. Par contre, si l'on entoure les deux premières formes par une troisième (3), il devient impossible de sélectionner les éléments internes.

Après l'insertion de texte, les lignes de texte sont indépendantes les unes des autres, permettant ainsi de les sélectionner une par une :



### 3.3.6. Commandes gestuelles

Alors que les formes sont tracées avec le bouton gauche de la souris, on représente les commandes avec le bouton droit.

Quatre grands types d'actions sont possibles :

- changement de couleur ;
- suppression ;
- modification de taille ;
- déplacement.

Les fonctionnalités concernant la modification de taille et le déplacement ne sont évidemment disponibles que sur sélection d'un composant de l'organigramme.

#### 3.3.6.1. Modification de la couleur

##### 1. Modifier la couleur d'une forme

Sélectionner une forme et dessiner un des gestes associés aux commandes 'Couleur : Noir', 'Couleur : Rouge', 'Couleur : Vert', 'Couleur : Bleu' pour modifier la couleur de la forme en noir, rouge, vert ou bleu.

L'option 'Couleur : Autre' ouvre une boîte de dialogue proposant un ensemble plus vaste de couleurs :



##### 2. Modifier de la couleur courante

Pour modifier la couleur courante du document, c'est-à-dire la couleur initiale des formes, procéder de la même manière que dans le paragraphe précédent, mais en ne sélectionnant aucune forme.

On peut noter que cette option est également disponible dans le menu principal (rubrique **Outils**).

### 3.3.6.2. Suppression

#### 1. Supprimer une seule forme

Pour supprimer une forme, on la sélectionne et on trace le geste associé à la commande 'Effacer : Sélection'.

#### 2. Supprimer toutes les formes

Pour effacer toute la zone de dessin, on trace le geste associé à la commande 'Effacer : Tout', sur sélection ou non d'une forme.

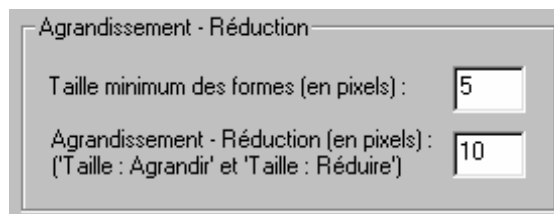
On fait remarquer que cette fonction est également disponible en utilisant l'item **Outils** du menu principal.

### 3.3.6.3. Modification de la taille

Cette fonctionnalité n'est disponible que sur sélection d'un composant de l'organigramme.

#### 1. Agrandissement ou réduction fixe

Avec les commandes 'Taille : Agrandir' et 'Taille : Réduire', on augmente ou diminue la taille de la forme sélectionnée d'un nombre de pixels constant. Ce paramètre est fixé à 10 pixels par défaut. On peut le modifier grâce à l'option **Outils/Taille des éléments...** du menu principal (cf. 3.3.3) qui propose les options suivantes :



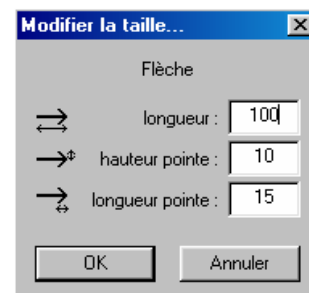
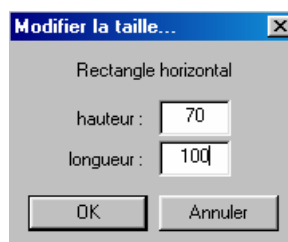
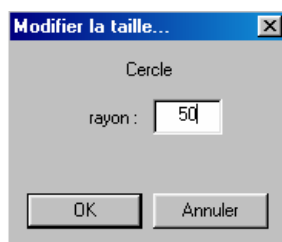
Pour modifier le nombre de pixels à utiliser lors de réduction ou agrandissement, remplir le deuxième champ. Le premier champ, quant à lui, indique la taille minimum des formes à ne pas dépasser.

#### 2. Taille variable

On se sert ici du geste 'Taille : Identique à celle du geste'. La taille de la forme est agrandie ou rétrécie selon la taille du geste dessiné (plus exactement, en fonction de la taille de son rectangle englobant). La forme sélectionnée n'est jamais déplacée, même si le geste n'est pas centré sur la forme.

#### 3. Autre taille

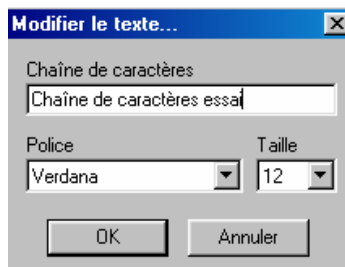
Cette fonctionnalité est exécutée sur la commande 'Taille : Autre'. On donne ici la possibilité d'entrer manuellement la nouvelle taille de la forme dans une boîte de dialogue réservée à cet effet. Cette fenêtre est propre à chaque type de composant, comme nous le montre les quelques exemples suivants :



Cette option semble la moins intéressante pour notre application, puisqu'elle nécessite l'utilisation du clavier. Pour limiter ce problème, on pourrait ajouter, à côté de chaque champ, des flèches permettant d'augmenter ou de diminuer la taille sur simples clics de souris...

### 4. Cas particulier du texte

Sur sélection d'un élément textuel, les commandes 'Taille : Autre' et 'Taille : Identique à celle du geste' mènent au même résultat : une boîte de dialogue proposant à l'utilisateur de modifier le texte ainsi que la police et la taille des caractères :



Pour l'instant, cette fonctionnalité nécessite l'utilisation de l'interface-clavier lorsque la chaîne de caractères est modifiée.

D'un autre côté, on peut utiliser les commandes 'Taille : Agrandir' et 'Taille : Réduire' qui augmentent ou diminuent la taille des caractères d'un unique pixel, sans possibilité ici de modifier le contenu ni la police.

### 3.3.6.4. Déplacement de la forme

#### 1. Déplacer selon l'axe horizontal ou l'axe vertical

Ces déplacements correspondent aux commandes 'Déplacement horizontal : Droite-Gauche / Gauche-Droite' et 'Déplacement vertical : Bas-Haut / Haut-Bas'. On translate la forme sélectionnée selon un vecteur horizontal ou vertical de la taille du rectangle englobant du geste tracé.

#### 2. Déplacements variables

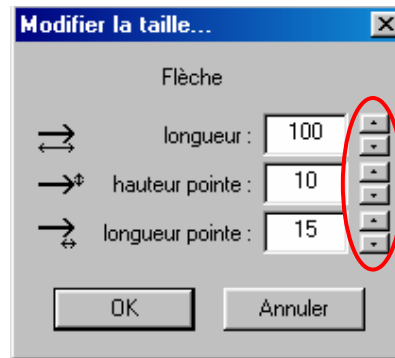
Cette option est plus souple et permet d'effectuer un déplacement dans n'importe quelle direction. Cette fois, le vecteur de translation est représenté par la distance entre le premier et le dernier point du geste. Pour cela, on associe ce style de déplacement à une seule commande appelée 'Déplacement : Autre'. Les lignes sont très bien appropriées à ce style de gestes, car on distingue clairement le premier et le dernier point du tracé.

## 3.4. Améliorations

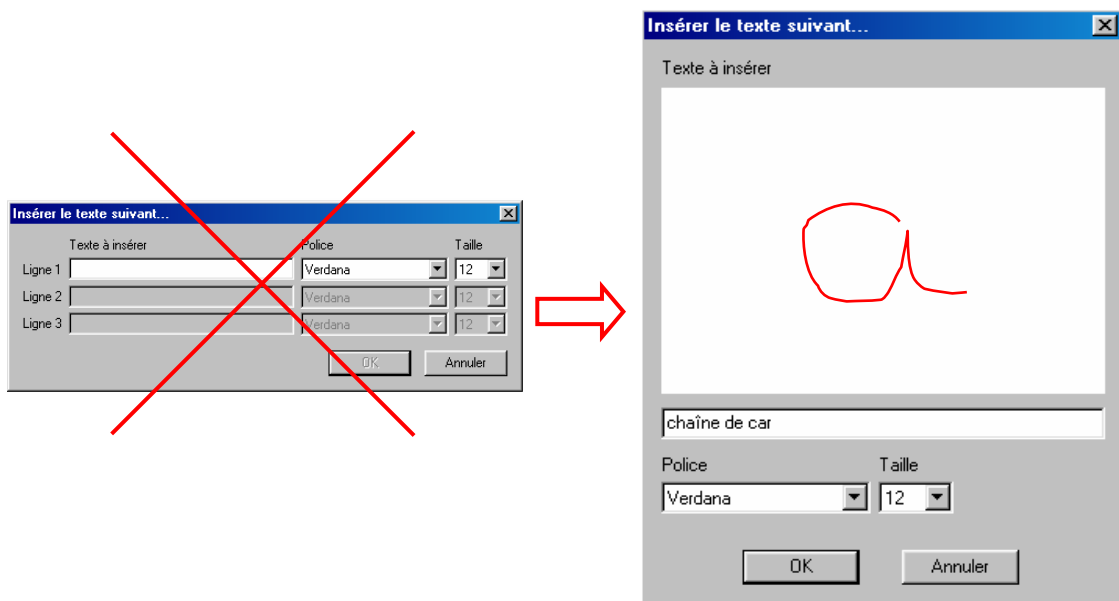
Beaucoup de fonctionnalités supplémentaires pourraient être implémentées dans de futurs travaux.

Comme nous l'avons vu au cours de ce chapitre, on pourrait notamment améliorer la sélection d'une forme qui peut paraître ambiguë dans certains cas (cf. 3.3.5). Pour ce qui est de la sélection, on pourrait également permettre à l'utilisateur de sélectionner plusieurs formes en même temps.

Afin d'utiliser au minimum l'interface clavier, il faudrait insérer des flèches permettant d'augmenter ou de diminuer tous les champs numériques :



Dans le même esprit, pour limiter l'utilisation du clavier, l'insertion de texte à l'aide de gestes graphiques semble indispensable. Dans cette optique, on a déjà associé le type 'Texte' à certains ensembles de gestes, représentant les lettres de l'alphabet, l'espace et le caractère d'effacement. Quand le geste associé à une insertion de texte est reconnu, il suffirait d'afficher une zone de dessin pour reconnaître les caractères (au lieu de la boîte de dialogue définie dans le paragraphe 3.3.4.2) :



D'autre part, les commandes d'édition habituelles, telles l'annulation ou la commande copier-coller, ne sont pas encore développées.

Une dernière fonctionnalité, qui semble essentielle pour la création d'organigrammes structurés, serait d'« aimanter » les objets entre eux. Cette option implique une analyse plus approfondie par rapport aux autres améliorations présentées précédemment. En effet, à quel moment peut-on dire que deux formes sont assez proches pour être « aimantées » ? On pourrait, par exemple, coller les formes dans l'ordre où elles sont tracées, ou alors, proposer à l'utilisateur de sélectionner les formes qu'il souhaite rapprocher, ou encore utiliser la notion de « grille » pour l'alignement des composants... Beaucoup de cas particuliers se présentent, une étude pointue sera donc nécessaire.

## Conclusion

---

Le projet de fin d'étude présenté dans ce rapport avait pour objectif le développement d'un système d'apprentissage et de reconnaissance de gestes graphiques. La première phase du projet a consisté en l'étude des travaux existants sur la reconnaissance de l'écrit (chapitre I - La reconnaissance de l'écrit). De ces recherches, on a pu extraire une méthode qui nous semblait intéressante pour la reconnaissance de gestes. Beaucoup de techniques utilisent les réseaux de neurones, les chaînes de Markov cachées ou la logique floue, l'algorithme choisi ici, élaboré par le chercheur D. Rubine, met en évidence des calculs statistiques et géométriques (chapitre II - Méthode de Rubine).

Pour tester la méthode, on a développé un logiciel en C++ avec Microsoft Visual Studio (chapitre III - Application logicielle, partie 2 - Système d'apprentissage et de reconnaissance de gestes). Cette application propose aux utilisateurs une interface simple leur permettant de créer leurs propres gestes graphiques. Ainsi, le système va apprendre les nouveaux gestes et les reconnaître dynamiquement.

Le bilan de la reconnaissance étant plutôt concluant, on a ensuite décidé d'implémenter une seconde application concernant la création d'organigrammes à l'aide de gestes appris avec notre premier système (chapitre III - Application logicielle, partie 3 - Editeur d'organigrammes).

Le logiciel final regroupe ces deux applications et possède également un éditeur de textes combiné avec une correction de documents à l'aide de gestes simples, développé dans le PFE de N. Hervouet en 2001-2002.

Le système de reconnaissance donne de très bons résultats. En effet, le taux de gestes correctement reconnus atteint facilement 95%. Ces statistiques ont été évaluées manuellement, il serait donc intéressant d'ajouter un procédé permettant de tester ces résultats dynamiquement. On pourrait demander à plusieurs utilisateurs de créer des ensembles de gestes, puis de les tester en dessinant plusieurs formes à la suite, en précisant à chaque fois si la forme est reconnue correctement.

L'éditeur d'organigrammes a été créé dans l'optique d'utiliser au minimum l'interface clavier. Les résultats sont très satisfaisants, cependant de nombreuses fonctionnalités pourraient être apportées (cf. chapitre III - paragraphe 3.4. Améliorations).

Dans notre application, on utilise la souris à la fois pour dessiner des formes et pour effectuer des commandes gestuelles. Après quelques modifications, on pourra même insérer du texte (pour l'instant, l'insertion de texte est effectuée à l'aide de l'interface clavier). Ce procédé fait apparaître la contrainte d'ambiguïté des tracés graphiques qui est propre aux interactions stylo. Cette ambiguïté est gérée pour l'instant au niveau de l'interface. En effet, l'utilisateur informe le système du statut des tracés graphiques qui vont être produits. On utilise le bouton gauche de la souris pour tracer les composants de l'organigramme et le bouton droit pour les commandes gestuelles. Quant à l'insertion de texte, elle est préalablement indiquée par le tracé d'un geste particulier. Cette méthode est évidemment la plus fiable mais également la plus contraignante. Il serait donc intéressant de chercher comment discriminer automatiquement les commandes, les données et le texte. La difficulté réside ici dans le fait que le tracé ne contient pas tout le temps l'information discriminante (par exemple : la lettre O, le chiffre 0, un cercle ou un encerclement peuvent paraître identiques). Pour cette discrimination, il serait souhaitable de recourir en plus au contexte du tracé.

Ce projet, très complet, a fait appel à différentes notions : d'une part, l'esprit de synthèse pour la recherche documentaire sur la reconnaissance de l'écrit, d'autre part, l'aspect « analyse de données » pour la compréhension de la méthode de Rubine, les connaissances en programmation objet pour l'implémentation de la méthode,... De plus, j'ai pu non seulement travailler sur un logiciel existant, mais également créer un nouvel outil, jusqu'à son terme. J'ai pu apprécier la programmation Windows qui mène à des interfaces graphiques vraiment agréables et conviviales pour les utilisateurs. Toutefois, j'ai rencontré quelques difficultés au niveau de la programmation et tout particulièrement lors de l'utilisation des MFC sous Visual Studio.

La reconnaissance des formes est sans aucun doute mon domaine favori. Je suis donc ravie d'avoir élaboré ce logiciel. Pour finir, je souhaiterais remercier Jean-Yves Ramel, mon encadrant, pour son aide précieuse.

# Table des matières

---

<b>Sommaire .....</b>	<b>3</b>
<b>Introduction .....</b>	<b>4</b>
<b>I. La reconnaissance de l'écrit .....</b>	<b>7</b>
1. La reconnaissance de l'écriture .....	7
1.1. <i>Ecriture cursive</i> .....	7
1.2. <i>Alphabet prédéfini</i> .....	8
2. La reconnaissance de gestes graphiques .....	10
2.1. <i>Méthode structurelle</i> .....	10
2.2. <i>Méthode statistique et géométrique de Fonseca et Jorge</i> .....	10
2.3. <i>Méthode statistique et géométrique de Rubine</i> .....	13
2.3.1. Description de la méthode.....	13
2.3.2. Exemple d'application : Interface de correction en ligne de documents électroniques.....	13
3. Conclusion.....	14
<b>II. Méthode de Rubine .....</b>	<b>16</b>
1. Description de la méthode.....	16
2. Caractéristiques de Rubine .....	16
3. Classification des gestes.....	19
4. Apprentissage.....	19
5. Rejet .....	21
5.1. <i>Calcul des probabilités</i> .....	21
5.2. <i>Distance de Mahalanobis</i> .....	21
<b>III. Application logicielle .....</b>	<b>23</b>
1. Présentation du logiciel .....	23
1.1. <i>Le logiciel existant</i> .....	23
1.1.1. Présentation.....	23
1.1.2. L'éditeur de textes .....	23
1.1.3. Le système de reconnaissance de gestes .....	23
1.1.4. Le système de correction de documents.....	24
1.2. <i>Besoins</i> .....	25
2. Système d'apprentissage et de reconnaissance de gestes.....	26
2.1. <i>Présentation</i> .....	26

2.1.1. Vocabulaire utilisé .....	26
2.1.2. Fonctionnalités principales .....	26
2.2. <i>Programmation</i> .....	27
2.2.1. Diagramme de classes .....	27
2.2.2. Apprentissage .....	28
2.2.2.1. <i>Description des classes</i> .....	28
1. CExemple .....	28
2. CClasse .....	28
3. CEnsemble .....	29
2.2.2.2. <i>Algorithme d'apprentissage</i> .....	29
2.2.2.3. <i>Enregistrement</i> .....	30
1. Enregistrement auxiliaire.....	30
2. Enregistrement de l'application globale.....	30
2.2.3. Reconnaissance .....	31
2.2.3.1. <i>CAReconnaître</i> .....	31
2.2.3.2. <i>Algorithme de reconnaissance</i> .....	31
2.2.3.3. <i>Enregistrement</i> .....	31
2.3. <i>Application finale</i> .....	32
2.3.1. Lancement de l'application .....	32
2.3.1.1. <i>Nouveau</i> .....	32
2.3.1.2. <i>Ouvrir</i> .....	32
2.3.2. Fenêtre principale .....	33
2.3.2.1. <i>Structure générale</i> .....	33
2.3.2.2. <i>Commandes utilisateur</i> .....	34
1. Description d'un ensemble.....	34
2. Gestion des classes.....	34
✓ Supprimer.....	34
✓ Renommer.....	35
✓ Propriétés .....	35
✓ Ajouter .....	35
3. Gestion des exemples .....	35
✓ Supprimer.....	35
✓ Renommer.....	35
✓ Propriétés .....	35
✓ Ajouter .....	35
4. Informations sur la sélection .....	35
2.3.2.3. <i>Zone de reconnaissance</i> .....	36
2.4. <i>Problèmes rencontrés</i> .....	36
2.5. <i>Résultats</i> .....	36
2.6. <i>Améliorations</i> .....	37
3. <i>Editeur d'organigrammes</i> .....	38
3.1. <i>Présentation</i> .....	38
3.1.1. Association geste-commande .....	38
3.1.2. Construction des formes .....	39
3.1.2.1. <i>Taille fixe</i> .....	39
3.1.2.2. <i>Taille variable</i> .....	40
3.1.3. Commandes utilisateur .....	41
3.2. <i>Programmation</i> .....	43
3.2.1. Diagramme de classes .....	44
3.2.2. Description des classes .....	44
3.2.2.1. <i>CElement</i> .....	44
3.2.2.2. <i>CLigne</i> .....	45
3.2.2.3. <i>CRectangle</i> .....	45
3.2.2.4. <i>CCarre</i> .....	45
3.2.2.5. <i>CLosange</i> .....	45
3.2.2.6. <i>CTriangle</i> .....	46
3.2.2.7. <i>CCercle</i> .....	46

3.2.2.8. <i>CEllipse</i> .....	46
3.2.2.9. <i>CFleche</i> .....	46
3.2.2.10. <i>CCroix</i> .....	47
3.2.2.11. <i>CTexte</i> .....	47
3.2.2.12. <i>CInconnu</i> .....	47
3.2.3. Les méthodes.....	47
3.2.4. Algorithme .....	47
3.2.5. Enregistrement.....	48
3.3. <i>Application finale</i> .....	48
3.3.1. Lancement de l'application .....	48
3.3.2. Association geste-commande .....	48
3.3.2.1. <i>Modifications du système d'apprentissage et de reconnaissance de gestes</i> .....	48
3.3.2.2. <i>Choix des ensembles associés à l'organigramme</i> .....	50
3.3.3. Fonctionnalités du menu .....	51
3.3.4. Création d'organigrammes .....	52
3.3.4.1. <i>Composants graphiques</i> .....	52
3.3.4.2. <i>Insertion de texte</i> .....	52
3.3.5. Sélection.....	53
3.3.6. Commandes gestuelles .....	54
3.3.6.1. <i>Modification de la couleur</i> .....	54
1. Modifier la couleur d'une forme .....	54
2. Modifier de la couleur courante.....	54
3.3.6.2. <i>Suppression</i> .....	55
1. Supprimer une seule forme.....	55
2. Supprimer toutes les formes .....	55
3.3.6.3. <i>Modification de la taille</i> .....	55
1. Agrandissement ou réduction fixe.....	55
2. Taille variable.....	55
3. Autre taille.....	55
4. Cas particulier du texte .....	56
3.3.6.4. <i>Déplacement de la forme</i> .....	56
1. Déplacer selon l'axe horizontal ou l'axe vertical .....	56
2. Déplacements variables.....	56
3.4. <i>Améliorations</i> .....	56
<b>Conclusion .....</b>	<b>58</b>
<b>Table des matières .....</b>	<b>59</b>
<b>BIBLIOGRAPHIE .....</b>	<b>62</b>
<b>ANNEXES .....</b>	<b>64</b>
I. Enregistrement lors de l'apprentissage .....	65
A. <i>Structure des fichiers</i> .....	65
B. <i>Exemple de fichier</i> .....	66
II. Enregistrement lors de la reconnaissance .....	70
A. <i>Structure des fichiers</i> .....	70
B. <i>Exemple de fichier</i> .....	71

## BIBLIOGRAPHIE

---

- [1] **Paper-less Editing and Proofreading of Electronic Documents**  
Jacques André & Hélène Richy - 1999
- [2] **Projet IMADOC : Interprétation et Reconnaissance d'Images et de Documents**  
INSA de Rennes : Jacques André, Guy Lorette, Hélène Richy,... - 1999
- [3] **Interface-stylo de correction en ligne de documents électroniques : Application aux pages Web**  
Jérôme Azé, Guy Lorette & Hélène Richy - 2000
- [4] **An approach to Graffiti Script Recognition**  
Asim Banerjee, Amey Bordikar & Pankaj Doke - 2002
- [5] **Finding extremal polygons**  
J.E. Boyce & D.P. Dobkin - 1985
- [6] **Towards an art based mathematical editor, that uses on-line handwritten symbol recognition**  
Juan López Coronado & Yannis A. Dimitriadis - 1994
- [7] **Construction d'un graphe structurel représentatif d'une forme**  
Patrice Dargenton, Hubert Emptoz & Nicole Vincent - 1993
- [8] **Appariement de deux graphes structurels quelconques pour la reconnaissance de lettres manuscrites**  
Patrice Dargenton, Hubert Emptoz & Nicole Vincent - 1994
- [9] **La Théorie et la Technique de REMUS (Reconnaissance de l'Ecriture ManUScrite)**  
INT (B. Dorizzi) et LIP6 (P. Gallinari)
- [10] **Entre manipulation directe et reconnaissance de l'écriture : les gestes écologiques**  
Daniel Etienne, Christophe Mertz & Jean-Luc Vinot - 2000
- [11] **L'interprétation de tableaux dans une situation d'interaction utilisateur-machine**  
Claudie Faure & Luc Julia - 1995
- [12] **A Simple Approach to Recognise Geometric Shapes Interactively**  
Manuel J. Fonseca & Joaquim A. Jorge - 1999
- [13] **Using Fuzzy Logic to Recognize Geometric Shapes Interactively**  
Manuel J. Fonseca & Joaquim A. Jorge - 2000
- [14] **Experimental Evaluation of an On-line Scribble Recognizer**  
Manuel J. Fonseca & Joaquim A. Jorge - 2000
- [15] **Determining the minimum-area encasing rectangle for an arbitrary closed curve**  
H. Freeman & R. Shapira - 1975
- [16] **Integrating Gesture and Snapping into a User Interface Toolkit**  
Tyson R. Henry, Scott E. Hudson & Gary L. Newell - 1990
- [17] **PDA and Gesture Use in Practice: Insights for Designers of Pen-based User Interfaces**  
James A. Landay, Allan Christian Long & Lawrence A. Rowe
- [18] **Extending an Existing User Interface Toolkit to Support Gesture Recognition**  
James A. Landay & Brad A. Myers - 1993
- [19] **Implications for a Gesture Design Tool**  
James A. Landay, Allan Christian Long & Lawrence A. Rowe - 1999
- [20] **Vectorisation et Reconnaissance de gestes graphiques : du Off-line au On-line**  
Jean-Yves Ramel & Nicole Vincent
- [21] **Computational geometry in C**  
Joseph O'Rourke - 1998

- [22] **Specifying Gestures by Example**  
Dean Rubine - 1991
- [23] **Amélioration des tracés issus d'une tablette graphique et Correction de documents texte à l'aide de gestes simples**  
Projet de Fin d'Etudes de Nicolas Hervouet
- [24] **Handwriting Recognition**  
[www.amug.org/amug/sigs/newton/nanug/PenReport/Handwriting.html](http://www.amug.org/amug/sigs/newton/nanug/PenReport/Handwriting.html)
- [25] **MyScript**  
Site de la société Vision Objects de Nantes : [www.visionobjects.com](http://www.visionobjects.com)
- [26] **Palm**  
Site de la société Palm Computing : [www.palm.com](http://www.palm.com)
- [27] **PenOffice et CalliGrapher**  
Site de la société PhatWare Corporation : [www.phatware.com](http://www.phatware.com)
- [28] **PenReader**  
Site de la société Paragon Software : [www.penreader.com](http://www.penreader.com)

# ANNEXES

## I. Enregistrement lors de l'apprentissage

- A. Structure des fichiers
- B. Exemple de fichier

## II. Enregistrement lors de la reconnaissance

- A. Structure des fichiers
- B. Exemple de fichier

# I. Enregistrement lors de l'apprentissage

## A. Structure des fichiers

Toutes les informations concernant les données statistiques liées à l'apprentissage d'un ensemble de gestes (caractéristiques de Rubine, matrices de variance-covariance,...) sont sauvegardées par défaut dans un fichier texte ('**.dat**'), appelé '**app\_**' suivi du nom de l'ensemble utilisé. Ce document est enregistré dans le répertoire '**Statistiques/Apprentissage/**'.

Le programmeur peut ainsi consulter ces données afin de comprendre quand une reconnaissance ne fonctionne pas correctement. On détaille ci-dessous la structure d'un fichier de ce type :

```
Type d'actions associées à cet ensemble de gestes : <type de commandes>

pour chaque classe c de l'ensemble

    CLASSE : <nom de la classe c> (<description>) ↵
    -----

    pour chaque exemple e de la classe c
        Vecteur caractéristique de l'exemple : '<nom de e>' : ↵
        <liste des 13 caractéristiques de Rubine de l'exemple e>
    fin

    -----

    Vecteur caractéristique moyen de la classe <nom de c> : ↵
    <liste des caractéristiques moyennes de la classe c>

    Matrice de variance-covariance de la classe <nom de c> : ↵
    <matrice de variance-covariance de la classe c>

    *****

fin

-----
-----

MATRICE DE VARIANCE-COVARIANCE MOYENNE : ↵
<liste des valeurs de la matrice de variance-covariance>

MATRICE INVERSE : ↵
<liste des valeurs de la matrice inverse de la matrice de variance-covariance>

-----
-----

pour chaque classe c de l'ensemble
    POIDS DE LA CLASSE <nom de la classe c> (<description>) ↵
    <liste des poids associés à chaque caractéristiques de la classe c>
fin
```

## B. Exemple de fichier

L'ensemble présenté ci-dessous contient les 17 gestes associés à la construction d'organigrammes (cercles, ellipses, rectangles,...).

```
Type d'actions associées à cet ensemble de gestes : 1

CLASSE : ellipseV (Aucune description)
-----

Vecteur caractéristique de l'exemple : 'exemple1' :
-0.47 ; -0.88 ; 168.58 ; 1.31 ; 36.14 ; -0.25 ; -0.97 ; 322.70 ; -5.23 ; 5.45 ; 2.90 ;
213200.00 ; 0.55 ;

Vecteur caractéristique de l'exemple : 'exemple2' :
-0.58 ; -0.81 ; 160.26 ; 1.19 ; 9.85 ; -0.41 ; -0.91 ; 340.80 ; -5.03 ; 6.15 ; 2.23 ;
195200.00 ; 0.60 ;

Vecteur caractéristique de l'exemple : 'exemple3' :
-0.36 ; -0.93 ; 181.47 ; 1.01 ; 28.16 ; -0.96 ; -0.28 ; 413.77 ; -1.69 ; 8.36 ; 11.82 ;
195200.00 ; 0.55 ;

...

Vecteur caractéristique de l'exemple : 'exemple15' :
-0.27 ; -0.96 ; 145.50 ; 1.23 ; 5.00 ; 0.00 ; -1.00 ; 300.28 ; -4.45 ; 4.54 ; 1.79 ; 291600.00
; 0.44 ;

-----

Vecteur caractéristique moyen de la classe ellipseV :
-0.43 ; -0.88 ; 174.01 ; 1.18 ; 16.10 ; 0.04 ; -0.29 ; 362.10 ; -3.70 ; 6.59 ; 5.65 ;
343810.37 ; 0.53 ;

Matrice de variance-covariance de la classe ellipseV :
0.54 ; -0.22 ; -9.60 ; 0.10 ; -1.18 ; 0.41 ; 0.23 ; -16.83 ; 0.30 ; -1.17 ; -1.99 ; -67630.34
; -0.02
-0.22 ; 0.10 ; 0.98 ; -0.05 ; -1.34 ; -0.15 ; -0.02 ; 2.03 ; -0.36 ; 0.22 ; -0.05 ; 58556.19 ;
0.01
-9.60 ; 0.98 ; 4226.66 ; -2.77 ; 982.43 ; 15.94 ; 28.75 ; 8413.51 ; 250.30 ; 295.89 ; 862.56 ;
4111036.88 ; 1.57
0.10 ; -0.05 ; -2.77 ; 0.12 ; 0.35 ; 0.27 ; -0.04 ; -16.20 ; 0.16 ; 0.08 ; 0.09 ; -82707.07 ;
0.01
-1.18 ; -1.34 ; 982.43 ; 0.35 ; 1429.33 ; 16.39 ; -9.35 ; 1100.21 ; -34.86 ; -25.17 ; -77.92 ;
-6772367.41 ; 0.40
0.41 ; -0.15 ; 15.94 ; 0.27 ; 16.39 ; 7.27 ; 1.19 ; -47.39 ; -2.96 ; -5.50 ; -15.18 ; -
318951.56 ; -0.15
0.23 ; -0.02 ; 28.75 ; -0.04 ; -9.35 ; 1.19 ; 6.45 ; 83.21 ; 3.76 ; 4.20 ; 11.76 ; 73998.87 ;
0.03
-16.83 ; 2.03 ; 8413.51 ; -16.20 ; 1100.21 ; -47.39 ; 83.21 ; 19943.98 ; 690.97 ; 785.27 ;
2322.14 ; 10012254.26 ; 5.45
0.30 ; -0.36 ; 250.30 ; 0.16 ; -34.86 ; -2.96 ; 3.76 ; 690.97 ; 69.49 ; 69.77 ; 218.46 ; -
632428.93 ; 0.20
-1.17 ; 0.22 ; 295.89 ; 0.08 ; -25.17 ; -5.50 ; 4.20 ; 785.27 ; 69.77 ; 77.33 ; 233.73 ; -
828066.37 ; 0.49
-1.99 ; -0.05 ; 862.56 ; 0.09 ; -77.92 ; -15.18 ; 11.76 ; 2322.14 ; 218.46 ; 233.73 ; 727.67 ;
-2756256.37 ; 1.10
-67630.34 ; 58556.19 ; 4111036.88 ; -82707.07 ; -6772367.41 ; -318951.56 ; 73998.87 ;
10012254.26 ; -632428.93 ; -828066.37 ; -2756256.37 ; 371616530979.09 ; -31097.81
-0.02 ; 0.01 ; 1.57 ; 0.01 ; 0.40 ; -0.15 ; 0.03 ; 5.45 ; 0.20 ; 0.49 ; 1.10 ; -31097.81 ;
0.02

*****

CLASSE : rectangleH (Aucune description)
-----

Vecteur caractéristique de l'exemple : 'exemple1' :
-0.24 ; 0.97 ; 278.10 ; 0.35 ; 10.00 ; 0.00 ; 1.00 ; 670.52 ; 5.18 ; 9.19 ; 7.76 ; 547600.00 ;
1.27 ;

Vecteur caractéristique de l'exemple : 'exemple2' :
```

```

-0.08 ; 1.00 ; 293.03 ; 0.26 ; 67.00 ; -1.00 ; 0.00 ; 648.52 ; 4.53 ; 8.36 ; 7.96 ; 846400.00
; 1.15 ;

Vecteur caractéristique de l'exemple : 'exemple3' :
-0.25 ; 0.97 ; 210.09 ; 0.41 ; 26.31 ; 0.99 ; -0.15 ; 497.28 ; 5.07 ; 9.29 ; 7.64 ; 384400.00
; 1.10 ;

Vecteur caractéristique de l'exemple : 'exemple4' :
-0.08 ; 1.00 ; 200.55 ; 0.38 ; 8.60 ; 0.81 ; 0.58 ; 502.51 ; 4.71 ; 6.27 ; 5.98 ; 466000.00 ;
1.10 ;

...

Vecteur caractéristique de l'exemple : 'exemple15' :
-0.07 ; 1.00 ; 141.48 ; 0.61 ; 14.32 ; -0.91 ; 0.42 ; 364.39 ; 4.80 ; 7.42 ; 7.99 ; 134444.44
; 0.94 ;

-----

Vecteur caractéristique moyen de la classe rectangleH :
-0.07 ; 0.99 ; 211.22 ; 0.40 ; 22.35 ; 0.14 ; 0.31 ; 510.17 ; 4.74 ; 7.66 ; 7.10 ; 503936.30 ;
1.08 ;

Matrice de variance-covariance de la classe rectangleH :
0.11 ; 0.01 ; -13.43 ; 0.00 ; -1.49 ; -0.03 ; -0.03 ; -25.76 ; -0.10 ; -1.01 ; -0.11 ;
47065.82 ; -0.04
0.01 ; 0.00 ; -1.89 ; 0.00 ; 0.30 ; -0.02 ; -0.01 ; -4.26 ; -0.02 ; -0.10 ; -0.03 ; -330.35 ;
-0.01
-13.43 ; -1.89 ; 27321.74 ; -38.16 ; 4631.98 ; 171.21 ; 10.18 ; 56428.37 ; -38.56 ; 266.93 ;
254.87 ; 64369783.00 ; 47.65
0.00 ; 0.00 ; -38.16 ; 0.08 ; -7.48 ; -0.24 ; 0.01 ; -76.03 ; 0.09 ; -0.05 ; -0.07 ; -
153461.79 ; -0.06
-1.49 ; 0.30 ; 4631.98 ; -7.48 ; 3366.60 ; 5.78 ; -47.07 ; 6957.15 ; -23.71 ; 56.09 ; 19.59 ;
8056768.66 ; 4.72
-0.03 ; -0.02 ; 171.21 ; -0.24 ; 5.78 ; 11.56 ; -0.80 ; 380.23 ; -1.17 ; 1.31 ; -2.10 ; -
215087.75 ; 0.60
-0.03 ; -0.01 ; 10.18 ; 0.01 ; -47.07 ; -0.80 ; 1.70 ; 70.81 ; -0.54 ; 0.03 ; -0.79 ;
236585.40 ; 0.13
-25.76 ; -4.26 ; 56428.37 ; -76.03 ; 6957.15 ; 380.23 ; 70.81 ; 119737.53 ; -62.61 ; 532.46 ;
559.45 ; 135527178.40 ; 100.32
-0.10 ; -0.02 ; -38.56 ; 0.09 ; -23.71 ; -1.17 ; -0.54 ; -62.61 ; 4.16 ; 3.14 ; 6.43 ; -
474445.81 ; 0.00
-1.01 ; -0.10 ; 266.93 ; -0.05 ; 56.09 ; 1.31 ; 0.03 ; 532.46 ; 3.14 ; 31.73 ; 12.44 ; -
539590.24 ; 1.07
-0.11 ; -0.03 ; 254.87 ; -0.07 ; 19.59 ; -2.10 ; -0.79 ; 559.45 ; 6.43 ; 12.44 ; 23.19 ; -
214753.74 ; 0.40
47065.82 ; -330.35 ; 64369783.00 ; -153461.79 ; 8056768.66 ; -215087.75 ; 236585.40 ;
135527178.40 ; -474445.81 ; -539590.24 ; -214753.74 ; 519026652463.43 ; 54836.18
-0.04 ; -0.01 ; 47.65 ; -0.06 ; 4.72 ; 0.60 ; 0.13 ; 100.32 ; 0.00 ; 1.07 ; 0.40 ; 54836.18 ;
0.14

*****

CLASSE : rectangleV (Aucune description) ...
CLASSE : triangleV (Aucune description) ...
CLASSE : ligneV (Aucune description) ...
CLASSE : ligneH (Aucune description) ...
CLASSE : croix (Aucune description) ...
CLASSE : flècheGD (Aucune description) ...
CLASSE : flècheDG (Aucune description) ...
CLASSE : flècheHB (Aucune description) ...
CLASSE : flècheBH (Aucune description) ...
CLASSE : carré (Aucune description) ...
CLASSE : cercle (Aucune description) ...
CLASSE : ellipseH (Aucune description) ...
CLASSE : losange (Aucune description) ...

```

## Reconnaissance de gestes graphiques - Annexes

### MATRICE DE VARIANCE-COVARIANCE MOYENNE :

```

0.02 ; 0.00 ; -0.43 ; 0.00 ; -0.21 ; 0.01 ; -0.00 ; -0.89 ; -0.02 ; 0.01 ; 0.01 ; -1253.05 ; -
0.00
0.00 ; 0.02 ; 0.06 ; 0.00 ; 0.09 ; -0.00 ; 0.01 ; 0.25 ; -0.03 ; -0.02 ; -0.04 ; 9395.09 ; -
0.00
-0.43 ; 0.06 ; 1173.85 ; -0.52 ; 698.07 ; 1.34 ; 0.48 ; 1931.80 ; 6.65 ; 5.76 ; 6.93 ;
4491364.21 ; 0.98
0.00 ; 0.00 ; -0.52 ; 0.01 ; -0.24 ; 0.00 ; 0.00 ; -1.11 ; 0.00 ; 0.01 ; -0.01 ; -194.95 ; -
0.00
-0.21 ; 0.09 ; 698.07 ; -0.24 ; 685.66 ; -0.19 ; -0.59 ; 857.50 ; 4.26 ; 0.63 ; -1.93 ;
2324066.45 ; 0.30
0.01 ; -0.00 ; 1.34 ; 0.00 ; -0.19 ; 0.33 ; -0.05 ; 1.01 ; 0.04 ; 0.02 ; -0.03 ; -6709.09 ;
0.00
-0.00 ; 0.01 ; 0.48 ; 0.00 ; -0.59 ; -0.05 ; 0.18 ; 2.20 ; -0.03 ; 0.07 ; 0.15 ; 12033.64 ;
0.00
-0.89 ; 0.25 ; 1931.80 ; -1.11 ; 857.50 ; 1.01 ; 2.20 ; 3771.19 ; 10.20 ; 10.75 ; 21.72 ;
8462922.15 ; 1.76
-0.02 ; -0.03 ; 6.65 ; 0.00 ; 4.26 ; 0.04 ; -0.03 ; 10.20 ; 2.79 ; 0.73 ; 2.22 ; 29668.65 ;
0.01
0.01 ; -0.02 ; 5.76 ; 0.01 ; 0.63 ; 0.02 ; 0.07 ; 10.75 ; 0.73 ; 3.44 ; 6.47 ; -89615.08 ;
0.07
0.01 ; -0.04 ; 6.93 ; -0.01 ; -1.93 ; -0.03 ; 0.15 ; 21.72 ; 2.22 ; 6.47 ; 18.77 ; -58257.45 ;
0.05
-1253.05 ; 9395.09 ; 4491364.21 ; -194.95 ; 2324066.45 ; -6709.09 ; 12033.64 ; 8462922.15 ;
29668.65 ; -89615.08 ; -58257.45 ; 162150507218.02 ; -3065.87
-0.00 ; -0.00 ; 0.98 ; -0.00 ; 0.30 ; 0.00 ; 0.00 ; 1.76 ; 0.01 ; 0.07 ; 0.05 ; -3065.87 ;
0.01

```

### MATRICE INVERSE :

```

53.9100083316 ; -1.0762425028 ; 0.0124090060 ; -3.7909153425 ; -0.0086292167 ; -1.9345226247 ;
0.4535737831 ; 0.0072903203 ; 0.4278029776 ; -0.3470075042 ; -0.0041501457 ; -0.0000004650 ;
2.4973251626
-1.0751526142 ; 45.7756749406 ; 0.0665177036 ; -5.2594509996 ; -0.0421457583 ; -0.6797877048 ;
-2.9742910141 ; -0.0285960934 ; 0.3931954824 ; -0.1685511275 ; 0.1004483305 ; -0.0000021462 ;
10.3090334808
0.0121827938 ; 0.0671144653 ; 0.0237500870 ; -0.0762204337 ; -0.0126005515 ; -0.0713120215 ; -
0.0101261608 ; -0.0091337031 ; -0.0030141920 ; -0.0141967358 ; 0.0065020644 ; -0.0000000158 ;
-0.2553567014
-3.7665898553 ; -5.2460081706 ; -0.0812207954 ; 153.1518490828 ; 0.0362694050 ; -2.4821142452
; -3.6016403223 ; 0.0841675397 ; -0.3706327158 ; -1.8687239861 ; 0.6421363562 ; -0.0000024338
; 11.5884263427
-0.0085256605 ; -0.0423336905 ; -0.0125943046 ; 0.0341660198 ; 0.0087907731 ; 0.0427955715 ;
0.0196369274 ; 0.0043522386 ; -0.0005283485 ; 0.0043102404 ; -0.0015263070 ; 0.0000000031 ;
0.1688453579
-1.9348111848 ; -0.6829847338 ; -0.0713286910 ; -2.4936070978 ; 0.0428270062 ; 3.4974756215 ;
0.9840397986 ; 0.0239801130 ; -0.0480173582 ; 0.0476354532 ; -0.0086793751 ; 0.0000002214 ; -
0.6419482088
0.4513814375 ; -2.9742629447 ; -0.0098886598 ; -3.6108041329 ; 0.0195174372 ; 0.9838224775 ;
6.3814050080 ; -0.0030032416 ; 0.0791305367 ; -0.0802978403 ; -0.0258316910 ; -0.0000002025 ;
-1.1095114217
0.0073293345 ; -0.0288580125 ; -0.0091463507 ; 0.0814673945 ; 0.0043605595 ; 0.0240222992 ; -
0.0029200258 ; 0.0040031255 ; 0.0008204512 ; 0.0075983797 ; -0.0035700710 ; -0.0000000121 ;
0.0064576499
0.4277185320 ; 0.3912619063 ; -0.0029940885 ; -0.3774639647 ; -0.0005380228 ; -0.0478735981 ;
0.0794519034 ; 0.0008106909 ; 0.4115683777 ; 0.0097490305 ; -0.0530018428 ; -0.0000000653 ;
0.1820320899
-0.3455632635 ; -0.1689319547 ; -0.0154150946 ; -1.8486095492 ; 0.0049555169 ; 0.0495602985 ;
-0.0836941173 ; 0.0079424072 ; 0.0096566153 ; 1.2544029640 ; -0.4132660737 ; 0.0000003630 ; -
8.0782705785
-0.0045752155 ; 0.1016392252 ; 0.0068459415 ; 0.6421755297 ; -0.0017078864 ; -0.0093681258 ; -
0.0250607813 ; -0.0036671179 ; -0.0529675173 ; -0.4136953351 ; 0.1974840371 ; -0.0000000854 ;
2.3970430807
-0.0000004594 ; -0.0000021516 ; -0.0000000159 ; -0.0000025077 ; 0.0000000034 ; 0.0000002206 ;
-0.0000001996 ; -0.0000000124 ; -0.0000000650 ; 0.0000003341 ; -0.0000000769 ; 0.0000000000 ;
0.0000038249
2.5648060997 ; 10.4376324765 ; -0.2315822908 ; 11.6494867788 ; 0.1571911533 ; -0.7093698279 ;
-1.0531620556 ; -0.0019687080 ; 0.1859630481 ; -8.2734971894 ; 2.4461025598 ; 0.0000032035 ;
193.2475498585

```

POIDS DE LA CLASSE ellipseV (Aucune description)  
-194.66 ; -24.76 ; -41.91 ; 0.28 ; 203.34 ; -0.28 ; -4.51 ; -7.75 ; 0.08 ; -2.91 ; 0.11 ; 0.35 ; -0.00 ; 27.22 ;

POIDS DE LA CLASSE rectangleH (Aucune description)  
-145.78 ; 1.55 ; 52.39 ; -0.25 ; 80.64 ; -0.06 ; -3.15 ; -7.17 ; 0.25 ; 1.82 ; -1.94 ; 0.44 ; -0.00 ; 133.21 ;

POIDS DE LA CLASSE rectangleV (Aucune description)  
-224.51 ; -1.03 ; 49.37 ; -0.45 ; 174.73 ; 0.05 ; -3.59 ; -10.06 ; 0.33 ; 1.58 ; -2.52 ; 0.75 ; -0.00 ; 148.04 ;

POIDS DE LA CLASSE texte (Aucune description)  
-102.07 ; 2.97 ; 41.13 ; -0.28 ; 128.57 ; 0.21 ; 0.27 ; -3.09 ; 0.15 ; -0.37 ; -4.00 ; 2.08 ; -0.00 ; 83.62 ;

POIDS DE LA CLASSE triangleV (Aucune description)  
-133.96 ; 25.18 ; 42.23 ; 0.09 ; 144.49 ; -0.18 ; -6.92 ; -10.71 ; 0.08 ; 1.04 ; -2.06 ; 0.71 ; -0.00 ; 89.35 ;

POIDS DE LA CLASSE triangleA (Aucune description)  
-156.06 ; 11.11 ; -44.35 ; 0.04 ; 169.74 ; -0.15 ; -5.28 ; -6.38 ; 0.15 ; -2.70 ; -1.48 ; 0.99 ; -0.00 ; 56.89 ;

POIDS DE LA CLASSE ligneV (Aucune description)  
-209.82 ; -6.46 ; 34.55 ; 0.15 ; 228.12 ; 0.14 ; -3.96 ; -1.56 ; -0.01 ; -0.45 ; -3.99 ; 1.49 ; -0.00 ; 45.84 ;

POIDS DE LA CLASSE ligneH (Aucune description)  
-52.99 ; 53.76 ; -3.28 ; 0.21 ; 5.90 ; 0.17 ; 0.76 ; 2.12 ; -0.10 ; -0.09 ; -1.74 ; 0.54 ; 0.00 ; 23.08 ;

POIDS DE LA CLASSE croix (Aucune description)  
-109.96 ; -37.75 ; 26.43 ; -0.78 ; 129.48 ; 0.47 ; 1.59 ; -0.66 ; 0.36 ; -1.61 ; -1.63 ; 0.89 ; 0.00 ; 71.79 ;

POIDS DE LA CLASSE flècheGD (Aucune description)  
-83.76 ; 50.77 ; 1.19 ; -0.24 ; 41.63 ; 0.31 ; 0.90 ; -0.60 ; 0.10 ; -1.93 ; -1.94 ; 1.77 ; 0.00 ; 87.67 ;

POIDS DE LA CLASSE flècheDG (Aucune description)  
-75.92 ; -44.88 ; 9.61 ; -0.18 ; 39.32 ; 0.22 ; -2.67 ; -2.66 ; 0.05 ; 1.04 ; -1.78 ; 1.04 ; 0.00 ; 107.96 ;

POIDS DE LA CLASSE flècheHB (Aucune description)  
-197.61 ; -3.92 ; 43.71 ; -0.41 ; 194.57 ; 0.30 ; -2.50 ; -2.83 ; 0.20 ; 2.14 ; -3.32 ; 1.80 ; -0.00 ; 105.71 ;

POIDS DE LA CLASSE flècheBH (Aucune description)  
-212.40 ; -1.13 ; -44.48 ; -0.43 ; 208.56 ; 0.34 ; -2.60 ; -9.41 ; 0.22 ; -3.20 ; -3.56 ; 1.95 ; 0.00 ; 110.26 ;

POIDS DE LA CLASSE carré (Aucune description)  
-149.06 ; 0.30 ; -39.41 ; -0.63 ; 128.91 ; 0.23 ; -0.29 ; -0.27 ; 0.34 ; 0.90 ; -2.07 ; 0.90 ; 0.00 ; 128.53 ;

POIDS DE LA CLASSE cercle (Aucune description)  
-100.90 ; -41.21 ; 8.06 ; -0.10 ; 111.54 ; -0.11 ; -3.20 ; -6.10 ; 0.20 ; 1.97 ; 3.02 ; -1.18 ; -0.00 ; 16.33 ;

POIDS DE LA CLASSE ellipseH (Aucune description)  
-70.85 ; 48.42 ; -19.23 ; 0.54 ; 49.45 ; -0.41 ; -5.87 ; -2.78 ; -0.07 ; -1.30 ; 0.73 ; 0.20 ; 0.00 ; 16.71 ;

POIDS DE LA CLASSE losange (Aucune description)  
-204.84 ; -37.89 ; 43.45 ; 0.47 ; 114.94 ; -0.44 ; -4.95 ; -12.04 ; -0.06 ; 0.99 ; 1.95 ; -0.22 ; -0.00 ; 130.94 ;

## II. Enregistrement lors de la reconnaissance

### A. Structure des fichiers

Dès lors qu'un geste est reconnu, on sauvegarde le nom de l'ensemble de gestes testé, le vecteur caractéristique du geste à reconnaître, la valeur de toutes les fonctions d'évaluation et enfin le nom du geste reconnu avec la valeur de la fonction d'évaluation maximum :

```
RECONNAISSANCE DE L'ENSEMBLE '<nom de l'ensemble de gestes utilisé>' ↵

vecteur caractéristique de la forme à reconnaître : ↵
<liste des 13 caractéristiques de Rubine>

valeur des fonctions d'évaluation de chaque classe : ↵
<liste des fonctions d'évaluation>

la forme est reconnue comme étant un geste '<nom du geste reconnu>' (fonction
d'évaluation = <valeur de la fonction d'évaluation maximum>) ↵

...

```

Un fichier de ce type est enregistré par défaut dans le répertoire '**Statistiques/Reconnaissance/**' et s'intitule '**rec\_**' suivi du nom de l'ensemble de gestes étudié.

## B. Exemple de fichier

On expose ci-dessous un extrait d'un fichier de reconnaissance. L'ensemble de gestes testé correspond à celui présenté dans l'exemple précédent : les 17 formes disponibles pour la construction d'un organigramme.

```
vecteur caractéristique de la forme à reconnaître :
0.21 ; -0.98 ; 123.04 ; 0.77 ; 10.77 ; 0.93 ; 0.37 ; 315.29 ; 1.92 ; 10.85 ; 17.31 ; 193600.37
; 0.98 ;

valeur des fonctions d'évaluation de chaque classe :
75.10 ; 26.41 ; 36.65 ; 45.84 ; 45.61 ; 115.50 ; -31.35 ; -22.52 ; 44.84 ; 54.94 ; 34.74 ;
25.46 ; 102.10 ; 140.47 ; 44.71 ; 55.02 ; 5.06 ;

la forme est reconnue comme étant un geste 'carré' (fonction d'évaluation = 140.47)

-----

vecteur caractéristique de la forme à reconnaître :
-0.75 ; -0.66 ; 200.32 ; 0.83 ; 19.42 ; -0.98 ; -0.21 ; 473.42 ; 7.21 ; 7.21 ; 3.09 ;
384400.73 ; 0.55 ;

valeur des fonctions d'évaluation de chaque classe :
108.77 ; 28.08 ; 36.33 ; 15.64 ; 45.20 ; 92.03 ; -11.54 ; -86.75 ; 48.56 ; -58.39 ; 36.38 ;
15.67 ; 36.02 ; 86.26 ; 134.36 ; 25.61 ; 45.48 ;

la forme est reconnue comme étant un geste 'cercle' (fonction d'évaluation = 134.36)

-----

vecteur caractéristique de la forme à reconnaître :
0.71 ; -0.71 ; 143.84 ; 0.23 ; 34.01 ; 1.00 ; -0.03 ; 267.37 ; -3.14 ; 13.15 ; 11.23 ;
96399.72 ; 0.99 ;

valeur des fonctions d'évaluation de chaque classe :
-47.40 ; -32.91 ; -81.93 ; -37.32 ; -24.21 ; 14.36 ; -156.79 ; 5.92 ; -59.62 ; 51.18 ; -21.45
; -108.74 ; -34.39 ; 22.44 ; -41.75 ; 60.11 ; -55.04 ;

la forme est reconnue comme étant un geste 'ellipseH' (fonction d'évaluation = 60.11)

-----

vecteur caractéristique de la forme à reconnaître :
-0.33 ; -0.94 ; 146.74 ; 1.32 ; 6.08 ; -0.16 ; 0.99 ; 304.68 ; -5.07 ; 6.44 ; 3.66 ; 141388.60
; 0.66 ;

valeur des fonctions d'évaluation de chaque classe :
210.89 ; 9.14 ; 59.55 ; 68.96 ; 78.63 ; 195.80 ; 89.11 ; -48.74 ; 92.60 ; 10.79 ; 27.41 ;
63.57 ; 173.17 ; 138.71 ; 107.01 ; 73.14 ; 46.56 ;

la forme est reconnue comme étant un geste 'ellipseV' (fonction d'évaluation = 210.89)

-----

vecteur caractéristique de la forme à reconnaître :
-0.02 ; 1.00 ; 157.89 ; 0.68 ; 12.04 ; 0.75 ; -0.66 ; 406.59 ; 6.30 ; 9.43 ; 9.87 ; 1082002.06
; 0.99 ;

valeur des fonctions d'évaluation de chaque classe :
-13.10 ; 149.44 ; 147.52 ; 110.50 ; 133.90 ; 14.55 ; 12.61 ; -46.65 ; 89.56 ; 27.41 ; 63.87 ;
101.56 ; -11.27 ; 62.13 ; 92.17 ; 9.51 ; 115.32 ;

la forme est reconnue comme étant un geste 'rectangleH' (fonction d'évaluation = 149.44)

-----

vecteur caractéristique de la forme à reconnaître :
-0.96 ; -0.28 ; 182.73 ; 0.68 ; 13.60 ; 0.59 ; 0.81 ; 406.42 ; 6.60 ; 6.60 ; 2.30 ; 510798.54
; 0.60 ;

valeur des fonctions d'évaluation de chaque classe :
46.62 ; 17.53 ; 6.06 ; 7.40 ; 10.86 ; 27.36 ; -38.35 ; -93.01 ; 40.81 ; -72.38 ; 40.56 ; -6.87
; -26.16 ; 45.04 ; 105.57 ; -12.66 ; 35.99 ;

la forme est reconnue comme étant un geste 'cercle' (fonction d'évaluation = 105.57)

-----

vecteur caractéristique de la forme à reconnaître :
1.00 ; 0.00 ; 161.30 ; 0.23 ; 133.73 ; 0.99 ; 0.10 ; 239.36 ; -2.14 ; 9.98 ; 17.26 ; 321110.45
; 0.76 ;
```

valeur des fonctions d'évaluation de chaque classe :  
-118.61 ; -34.55 ; -81.03 ; 9.56 ; -15.69 ; -38.02 ; -107.62 ; 46.03 ; -36.12 ; 85.65 ; -22.26 ; -63.87 ; -51.83 ; -19.54 ; -86.13 ; 24.84 ; -107.23 ;

la forme est reconnue comme étant un geste 'flècheGD' (fonction d'évaluation = 85.65)

vecteur caractéristique de la forme à reconnaître :  
-0.06 ; 1.00 ; 155.67 ; 1.22 ; 102.62 ; 0.26 ; 0.96 ; 274.43 ; 8.56 ; 10.06 ; 20.39 ; 439200.84 ; 0.88 ;

valeur des fonctions d'évaluation de chaque classe :  
45.84 ; 139.93 ; 184.42 ; 183.76 ; 169.39 ; 61.95 ; 158.92 ; -13.79 ; 152.66 ; 63.37 ; 96.86 ; 215.80 ; 88.55 ; 103.96 ; 103.12 ; 0.85 ; 117.20 ;

la forme est reconnue comme étant un geste 'flècheHB' (fonction d'évaluation = 215.80)

vecteur caractéristique de la forme à reconnaître :  
-0.99 ; -0.12 ; 159.84 ; 0.19 ; 105.93 ; -0.99 ; 0.13 ; 256.46 ; 6.85 ; 11.90 ; 22.25 ; 193600.37 ; 0.98 ;

valeur des fonctions d'évaluation de chaque classe :  
-73.28 ; 9.33 ; -34.90 ; 7.39 ; -28.79 ; -54.57 ; -99.00 ; -65.38 ; 23.79 ; -19.00 ; 99.14 ; -23.68 ; -55.16 ; 16.86 ; 24.51 ; -55.98 ; 18.24 ;

la forme est reconnue comme étant un geste 'flècheDG' (fonction d'évaluation = 99.14)

vecteur caractéristique de la forme à reconnaître :  
-0.24 ; -0.97 ; 187.93 ; 1.28 ; 150.48 ; -0.08 ; -1.00 ; 297.52 ; 0.23 ; 11.74 ; 15.62 ; 280399.20 ; 0.99 ;

valeur des fonctions d'évaluation de chaque classe :  
185.49 ; 46.09 ; 110.51 ; 116.52 ; 105.23 ; 191.47 ; 117.62 ; -10.66 ; 136.09 ; 78.31 ; 95.20 ; 134.90 ; 239.40 ; 186.97 ; 107.08 ; 49.70 ; 74.50 ;

la forme est reconnue comme étant un geste 'flècheBH' (fonction d'évaluation = 239.40)

vecteur caractéristique de la forme à reconnaître :  
0.00 ; 1.00 ; 191.13 ; 0.57 ; 13.60 ; -0.96 ; -0.29 ; 489.51 ; 4.71 ; 7.25 ; 6.76 ; 639998.17 ; 0.82 ;

valeur des fonctions d'évaluation de chaque classe :  
-16.24 ; 134.32 ; 119.61 ; 87.33 ; 122.17 ; 11.22 ; -3.41 ; -49.53 ; 67.19 ; 8.40 ; 40.47 ; 68.76 ; -41.77 ; 33.35 ; 87.52 ; 22.20 ; 88.92 ;

la forme est reconnue comme étant un geste 'rectangleH' (fonction d'évaluation = 134.32)

vecteur caractéristique de la forme à reconnaître :  
-0.30 ; -0.95 ; 152.74 ; 0.78 ; 22.20 ; -0.59 ; -0.81 ; 379.50 ; 7.81 ; 7.81 ; 3.33 ; 193600.37 ; 0.55 ;

valeur des fonctions d'évaluation de chaque classe :  
79.03 ; 1.23 ; 7.20 ; -2.47 ; 27.29 ; 84.02 ; -44.77 ; -64.32 ; 20.59 ; -36.59 ; 15.74 ; -7.58 ; 39.24 ; 89.40 ; 98.51 ; 29.26 ; -1.15 ;

la forme est reconnue comme étant un geste 'cercle' (fonction d'évaluation = 98.51)



# Reconnaissance de gestes graphiques

## Résumé :

Le rapport suivant a été réalisé dans le cadre d'un Projet de Fin d'Etudes en coordination avec le laboratoire RFAI. Une méthode statistique et géométrique a été implémentée afin de concevoir un système d'apprentissage et de reconnaissance de gestes graphiques. Les résultats de la reconnaissance étant plutôt concluants, un éditeur d'organigrammes à base de commandes gestuelles a également été créé.

## Mots-clés :

Reconnaissance de l'écrit  
Apprentissage et reconnaissance de gestes graphiques  
Méthode statistique et géométrique  
Traitement en ligne  
Editeur d'organigrammes  
Microsoft Visual Studio C++

---

## Abstract:

This report deals with a statistical method used to implement a system of gesture recognition. It describes a first toolkit for training new gestures. Then, a second soft analyses pen strokes and converts them into geometric shapes or user commands with the aim of create flow charts.

## Keywords:

Handwriting recognition  
Gesture training and recognition  
Statistical pattern recognition  
On-line  
Editor of flow chart  
Microsoft Visual Studio C++