# VECTORISATION D'IMAGES PAR OPTIMISATION DIRECTE DE BEZIGONS

## Motivation et vue d'ensemble de la méthode décrite dans le papier de recherche

La vectorisation d'image est un sujet crucial pour de nombreux graphistes qui sont souvent amenés à devoir transformer une image pixélisée en image vectorielle. Ceci demande un travail conséquent si le graphiste n'est pas aidé par un algorithme. Dans beaucoup de cas, ces algorithmes utilisent des bezigons, c'est-à-dire un chemin fermé constitué de courbes de Bézier, pour représenter les contours des formes vectorielles. Cependant, la plupart d'entre eux n'utilisent pas les propriétés intrinsèques de ces bezigons mais s'en servent uniquement pour approximer des courbes trouvées auparavant. Le papier de recherche proposé par Ming Yang, Hongyang Chao, Chi Zhang, Jun Guo, Lu Yuan, et Jian Sun cherche à utiliser ces propriétés afin de vectoriser un clipart de la façon la plus fidèle et esthétique possible.

Pour cela, ils procèdent selon les étapes suivantes :

1. Trouver une première approximation correcte de l'image et initialiser les bezigons à optimiser.
2. Optimiser chacun des morceaux $P_j$ de bezigons $B$ selon l'équation :

$$P_j^* = \arg\min_{P_j} \left[E_{data}(P_j, B) + E_{prior}(P_j, B)\right]$$

Dans l'équation précédente, $E_{data}$ représente la différence entre l'image de départ et l'image obtenue en rastérisant le bezigon. Un défi majeur est donc de trouver une fonction de rastérisation qui permette l'optimisation des paramètres. La fonction utilisée dans le papier utilise une décomposition en ondelettes discrètes qui permet d'obtenir une $E_{data}$ différentiable.

$E_{prior}$ est une énergie conçue selon des critères pré-établis et pénalisant les bezigons présentant certains aspects. Elle se décompose en 4 énergies : $E_{spt}$ pénalise les bezigons qui s'auto-intersectent, $E_{apt}$ pénalise les petites variations d'angles, $E_{hpt}$ pénalise les poignées trop proches des nœuds, $E_{lpt}$ pénalise les courbes tordues qui rajoutent de la longueur au bezigon. Les équations qui décrivent ces énergies sont disponibles dans le papier.

## Implémentation du procédé et des différentes énergies

### Initialisation des bezigons

Le papier ayant pour but l'optimisation, l'initialisation est laissée à l'utilisateur qui peut choisir soit tous les points (nœuds et poignées), soit juste les nœuds du futur bezigon. Dans ce cas, les poignées sont déterminées automatiquement, soit en les prenant sur la ligne reliant deux nœuds consécutifs, soit en utilisant les nœuds précédant et suivant comme guide (cf Fig 1).

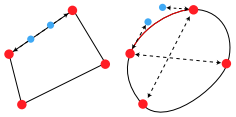

Figure 1 – Deux façons de déterminer les poignées (bleu) à partir des nœuds (rouge)

### $E_{DATA}$

Pour déterminer $E_{data}$, il faut au préalable rastériser le bezigon. Pour cela, le papier utilise une décomposition en ondelettes, décrite en [1]. Cette méthode a été implémentée en C++ à partir d'une implémentation en Python déjà existante. Pour un bezigon donné, un paramètre d'échelle et un paramètre de translation, on peut calculer le coefficient d'ondelette avec les formules données en [1]. Toutefois, cette méthode s'est avérée extrêmement coûteuse et lente : le calcul d'un coefficient de cadrant nécessite de réaliser les intersections entre toutes les courbes de Bézier et le cadrant considéré, et ceci doit être fait pour toutes les translations $k_x, k_y \in [\![0, 2^s]\!]$ et tous les paramètres d'échelle $s$ tel que $2^s < I.width$.

Nous avons donc considéré une autre méthode de rastérisation citée très succintement dans [1]. Elle utilise le principe suivant : pour un pixel $P$ de l'image, on projète un rayon qui va de ce point vers un autre point dont on sait qu'il est à l'extérieur de la forme (par exemple le bord gauche). On peut alors compter le nombre d'intersections entre les contours de la forme et ce rayon : si ce nombre est impair, alors $P$ est à l'intérieur de la forme et donc à colorier. Pour calculer les intersections entre ce rayon horizontal (à $y = h$, avec $h \in \mathbb{N}$) et les courbes de Bézier du bezigon, nous avons dans un premier temps créé une liste de points correspondant à l'interpolation cubique des courbes de Bézier. Il suffit alors de trouver les points consécutifs de la liste tels que $h$ soit entre les ordonnées de ces points pour qu'il y ait intersection. Pour améliorer la vitesse et la robustesse de ce procédé, nous avons ensuite calculé analytiquement l'intersection entre la ligne $y = h$ et les courbes de Bézier du bezigon pour déterminer les intersections. Différents cas de bords ce sont présentés, notamment celui dû aux points où la droite intersecte la courbe en un point de façon tangente (qu'il ne faut pas compter). Pour le contrer, il a fallu trouver une façon de discriminer de tels points : une façon de faire est de trouver un point du bezigon "juste avant" et un point "juste après" (en prenant en compte les

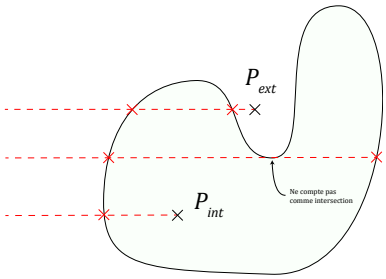


Figure 2 – Procédé de rastérisation du bezigon

éventuels changements de courbes de Bézier). Si ces deux points sont du même côté de la droite, c'est que le point en question ne doit pas être compté comme intersection. Cette méthode reste assez lente et présente la plus grande piste d'amélioration de notre implémentation (cf. Section 4).

### $E_{PRIOR}$

L'implémentation des énergies constitutives de $E_{prior}$ ont tout d'abord nécessité le calcul de la longueur d'une courbe de Bézier. Pour cela, nous avons tout d'abord subdivisé la courbe en différents segments et calculé la longueur de ces segments. En deuxième approximation, sachant que la courbe de Bézier est en fait polynomiale de degré 3, nous avons utilisé la méthode de quadrature de Gauss-Legendre qui permet d'approximer : $\int_{t_0}^{t_1} \sqrt{x'^2 + y'^2} \approx \sum_{1}^{3} w_i \sqrt{x'^2 + y'^2}(t_i)$, où les $w_i$ et $t_i$ sont donnés par respectivement : $\frac{5}{9}, \frac{8}{9}, \frac{5}{9}$ et $-\sqrt{3/5}, 0, \sqrt{3/5}$ dans le cas de courbes de Bézier cubiques.

L'énergie $E_{spt}$ fut la plus complexe à implémenter. Celle-ci nécessite de calculer la longueur de chaque courbe mais surtout d'en trouver les intersections. Le papier [2] décrit une méthode permettant de les trouver : dans l'image, on récupère les boîtes encadrant chacune des courbes de Bézier et on calcule leur recouvrement. Si ce recouvrement est positif, chacune des courbes est subdivisée puis le processus est réitéré jusqu'à obtenir des boîtes suffisamment petites pour être considérées ponctuelles. S'il est nul, c'est qu'il n'y a pas d'intersection. Pour récupérer les boîtes encadrant une courbe de Bézier nous avons considéré les points où l'une des dérivées en $x$ ou en $y$ s'annule et les points aux bords de la courbe, puis nous avons récupéré les minima en $x$ et en $y$ pour former le coin inférieur gauche et le point supérieur droit.

### Optimisation

Une fois les énergies implémentées, il faut optimiser le bezigon pour trouver le minimum de l'énergie totale. Le papier laisse entendre que l'optimisation peut se faire de façon linéaire. Bien que partiellement adaptée en raison de notre méthode de rastérisation, une descente de gradient classique nous a permis d'obtenir des résultats corrects. Comme indiqué dans le papier, l'optimisation peut se faire par morceaux de bezigon, c'est-à-dire 5 points. Le procédé est donc le suivant : pour chaque morceau de bezigon, déterminer le gradient par rapport à ces 5 points, mettre à jour le bezigon avec les nouveaux 5 points obtenus après la descente de gradient, et réitérer pour un nombre d'itérations donné. Pour empêcher les points de sortir du cadre de l'image suite à un gradient trop élevé, nous avons fixé une limite au déplacement des points.

## Résultats et choix des différents paramètres

### Les résultats obtenus

Malgré nos efforts, les résultats obtenus ne sont pas à la hauteur des logiciels disponibles sur le marché. Toutefois, nous avons pu obtenir des résulats honnêtes (cf. Fig 3) et surtout retrouver les résultats du papier de recherche en ce qui concerne les différentes énergies implémentées. Mais avant cela, il a fallu trouver les paramètres de l'optimisation. (Le programme renvoie un .svg à chaque étape de l'optimisation de la couleur choisie par l'utilisateur).

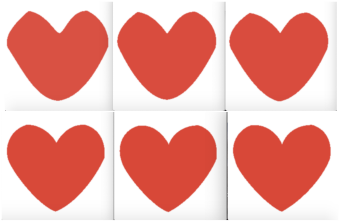

Figure 3 – Vectorisation d'un clipart de cœur

### Choix du taux d'apprentissage optimal et du $\epsilon$ lors du calcul du gradient

À chaque itération, l'un des paramètres du Bezigon est mis à jour selon $x[i] = x[i] - \alpha \times \frac{\partial E}{\partial x}[i]$, donc un taux d'apprentissage trop grand fera trop se déplacer les points tandis qu'un taux d'apprentissage trop petit nécessitera trop d'itérations avant convergence. L'importance d'un bon taux d'apprentissage peut s'illustrer sur la figure suivante :

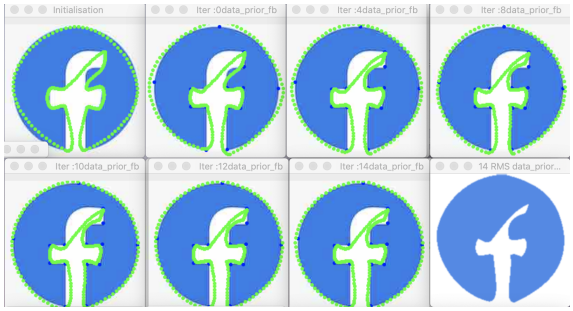

Figure 4 – $\alpha$ trop faible ($\approx 0.1$)

Figure 5 – $\alpha$ trop élevé ($\approx 100$)

Comme illustré, lorsque $\alpha$ n'est pas assez élevé (les résultats ne convergent toujours pas après 14 itérations) et lorsque $\alpha$ est trop élevé les points s'éloignent trop de leur position d'origine.

Le gradient de l'énergie est approximé par $\frac{\partial E}{\partial x} = \frac{E(x+\epsilon) - E(x-\epsilon)}{2\epsilon}$. Pour le choix du $\epsilon$, il faut prendre en compte les deux énergies $E_{data}$ et $E_{prior}$. Notre méthode de rastérisation n'est pas assez fine pour prendre en compte les changements de variable trop petits (les pixels passent directement de blanc à la couleur souhaitée), ainsi pour $\epsilon$ trop faible $E_{data}$ n'est pas modifiée et le bezigon ne peut pas se conformer à l'image (ce problème ne survient pas en utilisant la rastérisation par ondelettes car le pixel prend une nuance dépendant de son taux d'occupation par la forme). Au contraire pour $\epsilon$ trop grand, les variations approximées de $E_{prior}$ ne sont plus représentatives des vraies variations. Ces résultats sont illustrés en Figure 6.

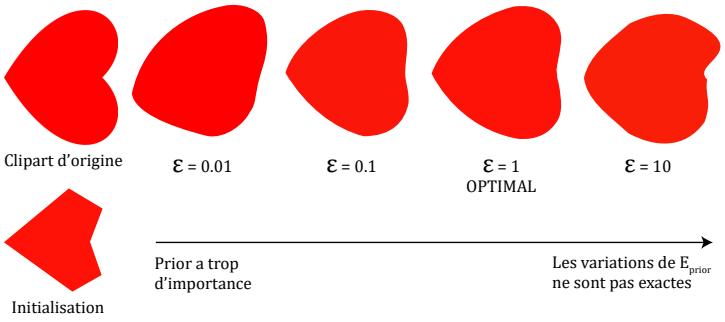


FIGURE 6 – Illustration du choix de $\epsilon$

## LES DIFFÉRENTES ÉNERGIES

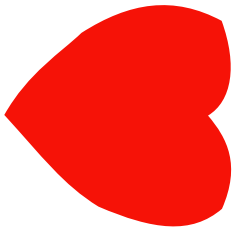

FIGURE 7 – Uniquement $E_{data}$

Nous avons également pu mettre en évidence l'importance des 5 énergies citées dans le papier. Tous les résultats présentés sont ceux obtenus pour l'image de cœur présentée auparavant, à la $10^{ème}$ itération du procédé d'optimisation. Tout d'abord, $E_{data}$ permet bien de s'approcher de l'image d'origine (cf. Fig 7). $E_{spt}$ permet de retirer l'auto-intersection du bezigon, mais cela demande un nombre conséquent d'itérations, si bien que sur la figure le bezigon obtenu à la $10^{ème}$ itération n'est pas très proche du cœur d'origine. $E_{apt}$ permet bien de retirer les petites variations d'angle (cf. Fig 9). $E_{hpt}$, qui doit pénaliser les petites poignées, a une importance moins flagrante que les autres (cf. Fig 10). Enfin $E_{lpt}$ permet de retirer les éventuelles torsions mais tout comme pour $E_{spt}$, cela demande un nombre conséquent d'itérations.

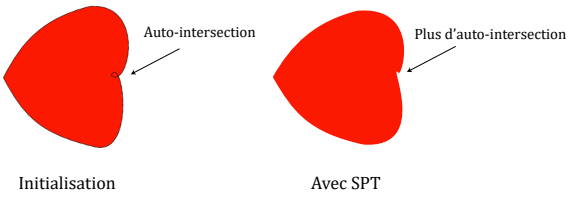


FIGURE 8

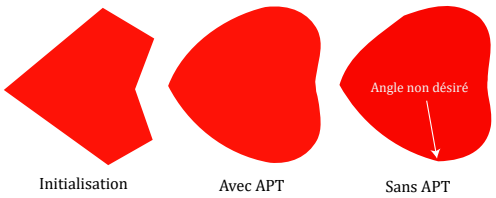


FIGURE 9

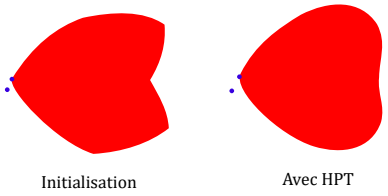


FIGURE 10

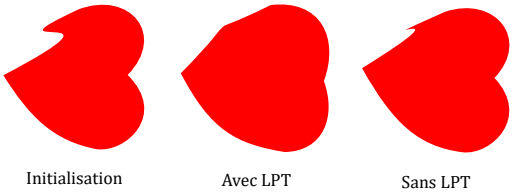


FIGURE 11

## Conclusion et pistes d'amélioration

En conclusion, ce projet de vectorisation nous a permis d'obtenir de premiers résultats très satisfaisants, et ce à partir d'images de mauvaises qualités. Les résultats, exportables au format svg, sont obtenus grâce à l'optimisation de différentes énergies se basant sur l'image de base et sur des considérations graphiques propres à la vectorisation d'images par courbes de Bézier.

Notre implémentation présente néanmoins de nombreuses pistes d'amélioration. En effet, son implémentation est coûteuse et lente. La rastérisation d'images est d'autant plus lente que l'image d'origine est de grande dimension, ce qui nous force à de petits formats en entrée. Il est néanmoins important de constater que les résultats obtenus pouvaient être bons en partant de ces tailles réduites, à compter que l'image ne soit pas de trop mauvaise qualité. Par ailleurs, notre méthode de rastérisation, comme cela a été détaillé dans les sections précédentes, n'est pas assez fine pour rendre compte des petits déplacements des points de contrôle des différentes courbes de Bézier. Une grande piste d'amélioration serait de réussir à implémenter de façon optimale la rastérisation par ondelette présentée en [1].

D'autres pistes d'améliorations sont également à considérer : nous nous limitons pour l'instant à la vectorisation d'image à partir d'un seul bezigon, alors que des formes plus complexes nécessitent plusieurs de ces surfaces pour être vectorisées (par exemple des lettres fermées comme a ou b, où le trou doit être considéré comme un bezigon). Pour cela, il faudrait seulement inclure dans le procédé plusieurs étapes d'initialisation durant lesquelles un nouveau bezigon serait rentré puis optimisé.

Enfin, nous pourrions automatiser l'initialisation de ces courbes à l'aide de méthodes de "curve fitting" comme détaillée en [3] : celles-ci nous permettraient de rendre plus pratique le programme et pallier à la difficulté d'ordonner les points saillants détectés sur l'image afin de relier les courbes de Bézier entre elles. Il faut pour cela parvenir à récupérer les contours de l'image et les transformer en courbes digitalisées. Une autre façon de faire serait de partir des contours puis de récupérer certains points d'intérêt en utilisant l'algorithme de Ramer-Douglas-Peucker avant de fit des courbes de Bézier avec une méthode des moindres carrés.

## Références bibliographiques


**[1]** J. Manson and S. Schaefer. Wavelet Rasterization, 2011

**[2]** A Primer on Bézier Curves §29, https ://pomax.github.io/bezierinfo/, 2011

**[3]** A. Glassner, Graphics Gems page 612, 1990