Dimitri Charneux
Rémi Kroll

Rendu tp: codage d'un contour

Durant ce TP, nous allons étudier des algorithmes permettant de déterminer les contours d'une forme afin de connaître les avantages et les inconvénients de chacun d'entre eux. Nous étudions, dans un premier temps, les descripteurs de Fourier, puis ensuite, l'algorithme de la corde.

Avant d'utiliser ces algorithmes, nous allons comprendre comment utiliser les fonctions disponibles en R pour manipuler les contours d'une forme. Nous avons besoin de savoir comment charger le contour d'une forme en mémoire, et comment afficher ces contours sous forme d'un graphique.

Code R

Les formes sont disponibles dans les fichiers « .txt ou « .png ». Ces deux types de fichiers vont influencer la façon dont on va charger le contour de la forme.

Les fichiers de forme en texte contiennent la liste des points des contours sous la forme « x y ». Pour extraire cette liste, on va dans un premier temps lire la table, puis, pour faciliter la manipulation des contours, on enregistre les points dans une liste de nombre complexe en prenant comme réel, les x des points, et en imaginaire, les y.

Les fichiers de forme d'une image en « png » sont, contrairement aux fichiers « txt », des fichiers binaires. On utilise une fonction de la librairie « EBImage » pour lire et extraire les coordonnées du contour. Cette fonction revoit une liste de matrices contenant les coordonnées des contours de l'image. Après cette étape, il suffit de passer les coordonnées sous formes complexes de la même manière que pour un fichier « txt ».

Les points du contour dans la variable « cont » sont codés sous la forme d'une liste de nombres complexes. Les nombres complexes sont créés à partir du x comme nombre réel et du y comme partie imaginaire du nombre.

```
plot (cont, main = nom, type = "o", asp = 1, col = "black", ylim = rev (range
(Im (cont))))
```

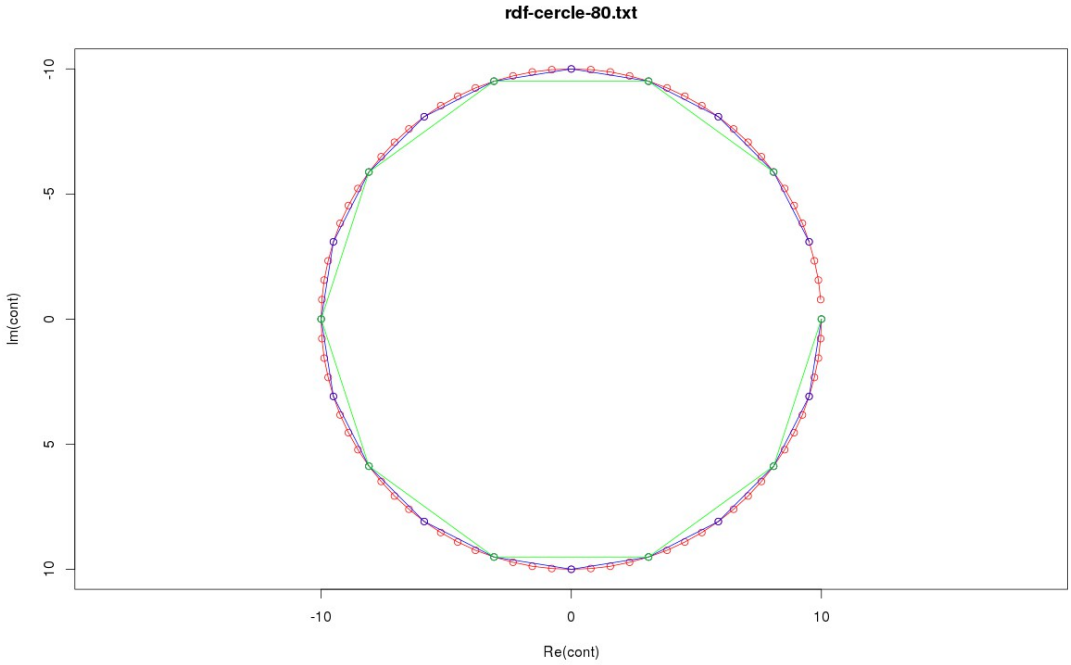
La méthode plot permet d'afficher sous forme graphique la liste des points contenus dans cont. Le paramètre main correspond au titre du graphique, le type permet d'entourer chaque points, asp donne le rapport entre x et y (avec un rapport de 1, l'échelle est la même pour les x comme pour les y). Le dernier paramètre « ylim » permet de déterminer les valeurs maximales et minimales en y.

```
      rev (range (Im (cont)))
```

« Im » permet de récupérer une liste avec seulement la partie imaginaire de la liste « cout ». « range » permet de récupérer le min et le max de la liste. Enfin « rev » renverse la liste pour donner

les deux nombres dans le bon ordre.

Pour afficher le contour d'un cercle en rouge, il suffit de changer le nom du fichier par le fichier souhaité et remplacer « Black » par « Red ».




```
con4 = cont[c(TRUE, FALSE, FALSE, FALSE)]
con8 = con4[c(TRUE,FALSE)]
lines(con4,type = "o", col="blue")
lines(con8, type = "o", col= "green")
```

Les lignes de codes ci-dessus permettent de n'afficher qu'un point sur quatre (con4) et un point sur huit (con8) du contour. « cont » contient la liste des points du contour. Avec la fonction « c » de R, nous pouvons extraire certains éléments de la liste. Ici, nous donnons une expressions de booléens pour lui signifier la manière dont on veut extraire l'ensemble des points. Par exemple, cont[c(TRUE,FALSE)] est une expression qui prend un élément sur deux de la liste « cont ».

Après observation du comportement des courbes, nous pouvons en déduire, que le contour du cercle est de plus en plus approximatif, mais, est encore reconnaissable avec réduction par huit des points de la forme.

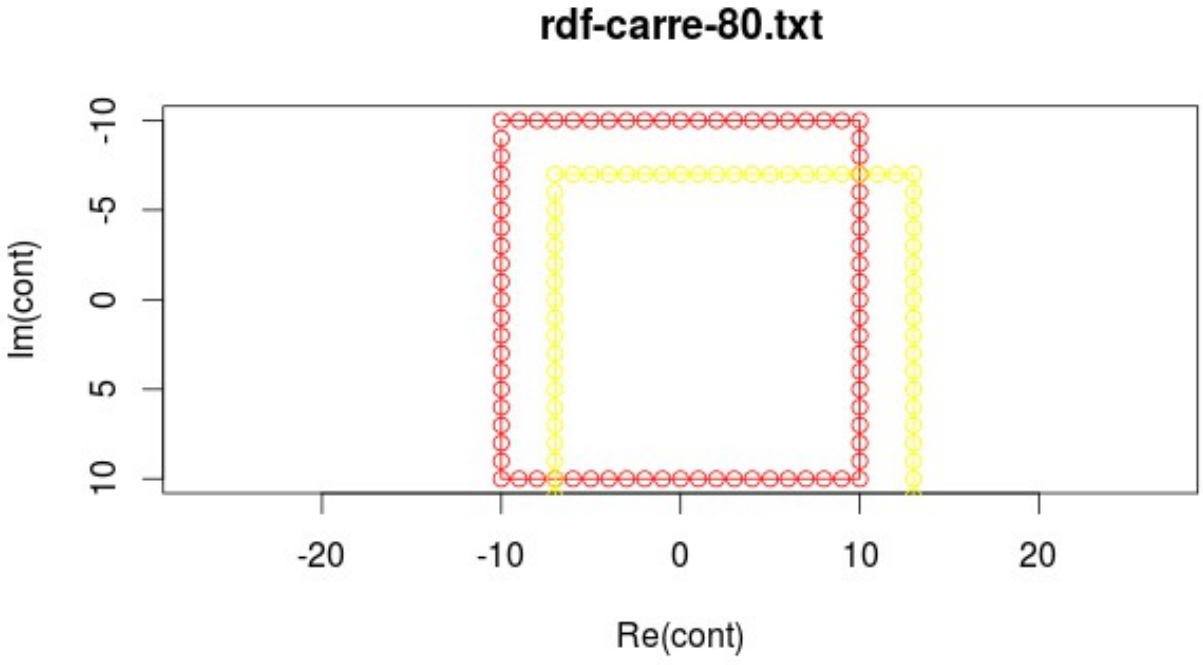
Dans cette partie nous avons pu comprendre comment manipuler le contour d'une forme sous R (l'affichage d'une forme, la modification de ses points..). Maintenant, nous allons utiliser les descripteurs de Fourier pour simplifier le codage d'une forme.

Descripteur de Fourier

On calcule le descripteur de Fourier de la manière suivante : « fourier = fft(cont, FALSE)/length(cont) ». On utilise la fonction fft de R puis on la normalise en divisant par le nombre de points.

« lines(fft(fourier, TRUE), type = "o", col= "yellow") » permet de faire la transformation de Fourier inverse à la variable « fourier » calculé précédemment. On voit que notre nouvelle ligne jaune est superposé au contour rouge. une transformée de Fourier inverse permet donc de reconstituer exactement le contour initial du cercle.

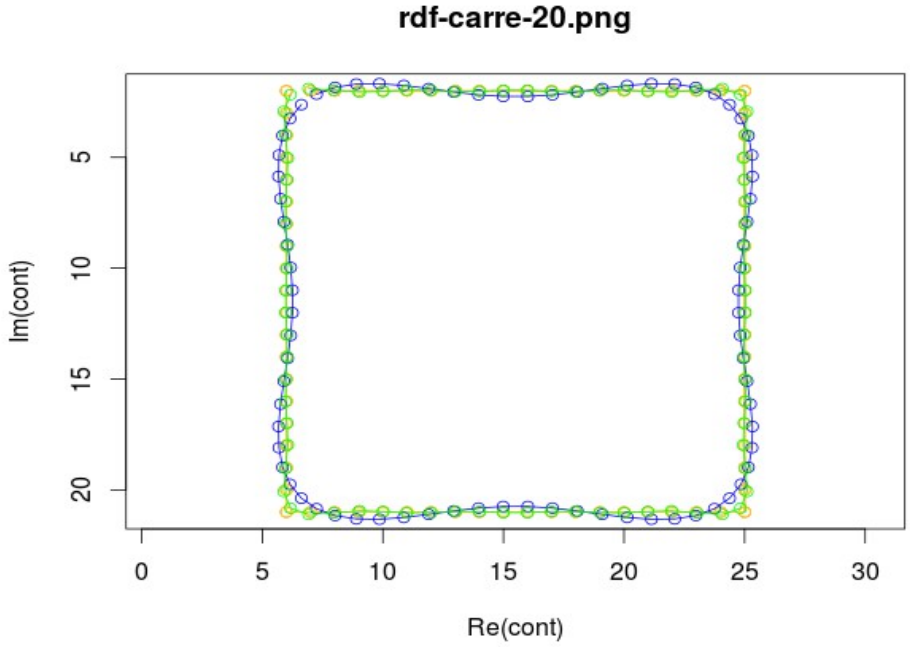
Le descripteur de Fourier Z0 correspond au centre de gravité de la forme. En ajoutant une constante complexe à ce descripteur, on constate que la forme est dessinée correctement mais qu'elle est décalée par rapport à l'image de base.



Filtrage des descripteurs de Fourier

Les descripteurs de Fourier vont nous servir pour simplifier le codage d'une forme. On utilise une méthode de filtrage des descripteurs de Fourier. Pour cela, nous avons créé une fonction « rdfAnnuleDescFourier » qui renvoie un vecteur de même dimension que le vecteur de descripteurs de Fourier précédemment calculé mais dont certaines composantes on été annulées. On passe donc le descripteur desc et le ratio de coefficients annulés compris entre 0 et 1. On remarque que la forme de l'objet est simplifiée.

Notre fonction va, dans un premier temps, mettre dans une attribut « tmp », combien de valeurs du descripteur « desc » on veut enlever. On va donc partir du milieu du « desc » et enlever du milieu moins l'attribut « tmp » jusqu'au milieu plus « tmp ». Ces valeurs là seront donc mises à nul pour qu'elles ne soient pas prises en compte lors du tracé du contour. Cette fonction peut servir à rendre la signature plus compacte (cf image ci-dessous) mais il ne faut pas annuler trop de descripteurs sinon la forme ne sera plus reconnaissable.



Les descripteurs de Fourier permettent d'obtenir de bon résultat pour déterminer les contours d'une forme.

Passons maintenant à l'étude de la réduction d'une chaîne de contour par l'algorithme de la corde.

Réduction d'une chaîne de contour

La réduction de contour grâce à l'algorithme de la corde fonctionne de la manière suivante : on trace une droite partant des deux extrémités de notre contour et pour chaque point de ce contour, on vérifie que la distance du point à la droite tracée est plus petite que la distance souhaitée. Ensuite, on recommence l'algorithme avec le point le plus éloigné jusqu'à ce que tout les points respectent la distance souhaité. Les points du contour réduit seront les points utilisés pour tracer les droites.

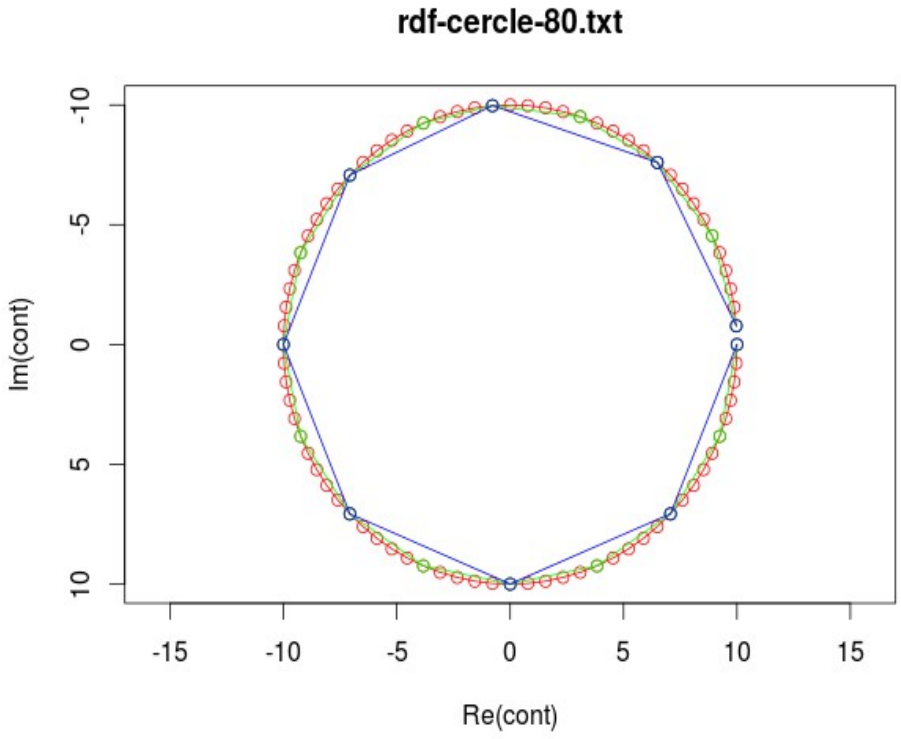
Pour rendre fonctionnel la fonction « redfAlgorithmeCorde » nous avons complété la fonction « rdfDistances » qui calcule les distances de chaque points avec la corde, c'est à dire la droite reliant les deux points aux extrémités du contour. La distance a été calculer de cette manière :

```
abs(Im((cont - debut) * Conj(fin - debut))) / Mod(fin - debut)
```

La variable « cont » de cette ligne représente le contour de la forme tandis que les variables « debut » et « fin » correspondent au premier et au dernier point de ce contour. Cette formule est une adaptation de la formule ci-dessous adapté au langage R.

$$d(\mathrm{A},(d)) = \mathrm{AA}_h = \frac{|ax_A + by_A + c|}{\sqrt{a^2 + b^2}}$$

Un exemple de contour par algorithme de la corde est présent ci-dessous.



L'image ci-dessus représente trois courbes. La rouge correspond au cercle tracé sans réduction. Ensuite, la courbe verte correspond à une réduction du contour par l'algorithme de la corde en prenant une distance maximale entre un point et la courbe de 0.5 pixel. Cette courbe représente très bien le cercle et ne contient qu'un cinquième des points. Pour finir, la courbe bleu représente la même réduction de contour mais cette fois ci avec une distance maximale de 2 pixels. Cette courbe ne contient que peu de points et le cercle pourrai être confondu avec un octogone.

Nous pouvons donc voir que l'algorithme de la corde reconnaît assez bien la forme d'un cercle à la condition d'avoir une distance entre les points du contours et de la corde assez petite. Il faut donc calculer un grand nombre de points pour avoir un résultat satisfaisant.

Après avoir vu les méthodes de réduction de contour par les descripteurs de Fourier et par l'algorithme de la corde, nous allons maintenant les comparer pour savoir laquelle est la plus efficace.

Comparaison des deux approches

Dans cette partie, nous allons comparé, sur différentes formes, la réduction de contour via la méthode de Fourier et via l'algorithme de la corde. Nous allons ensuite garder le contour réduit qui nous semble le plus efficace.

Rectangle horizontale :

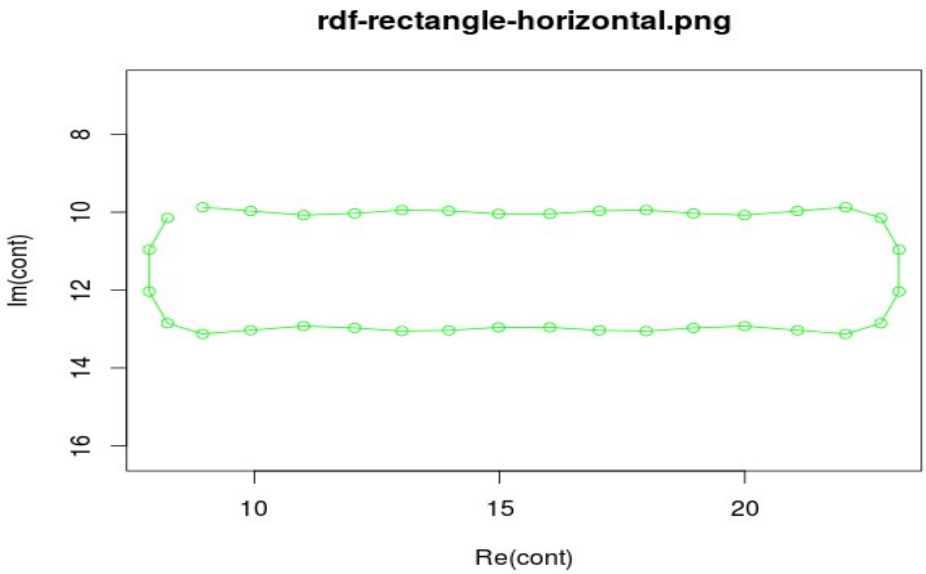


Illustration 1: contour d'un rectangle avec Fourier

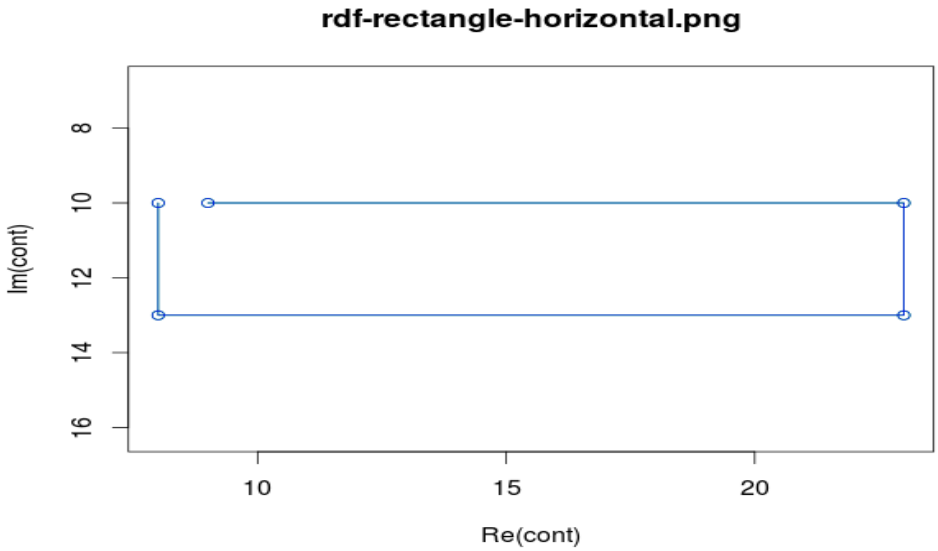


Illustration 2: contour d'un rectangle avec l'algorithme de la corde

Carré :

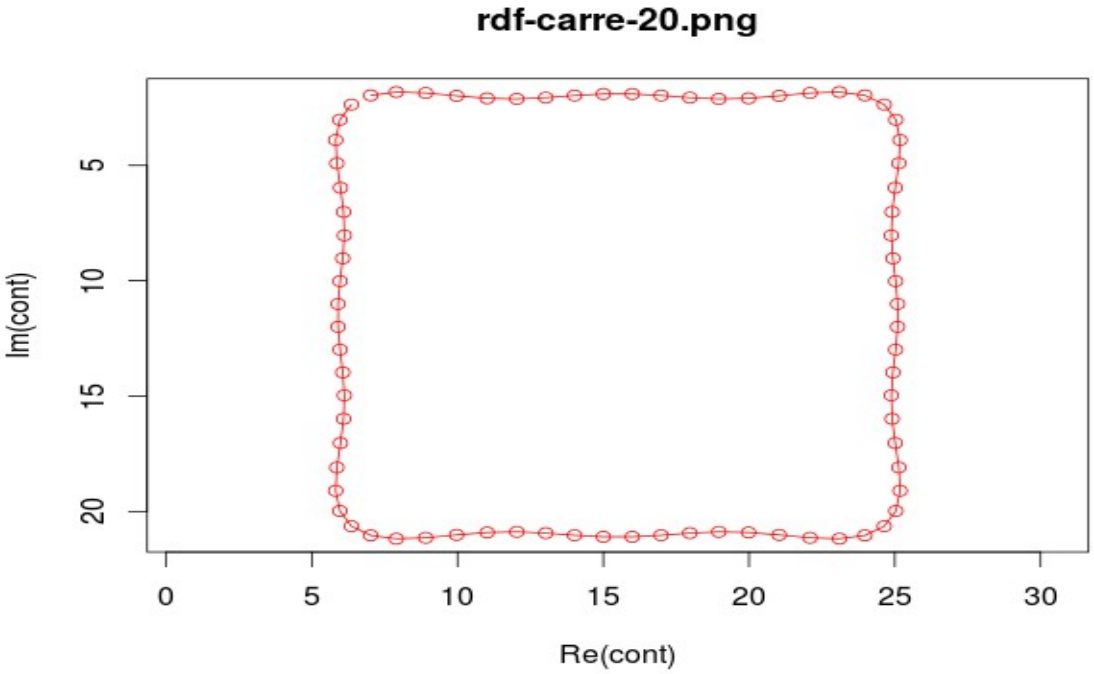


Illustration 3: contour d'un carré avec la méthode de Fourier

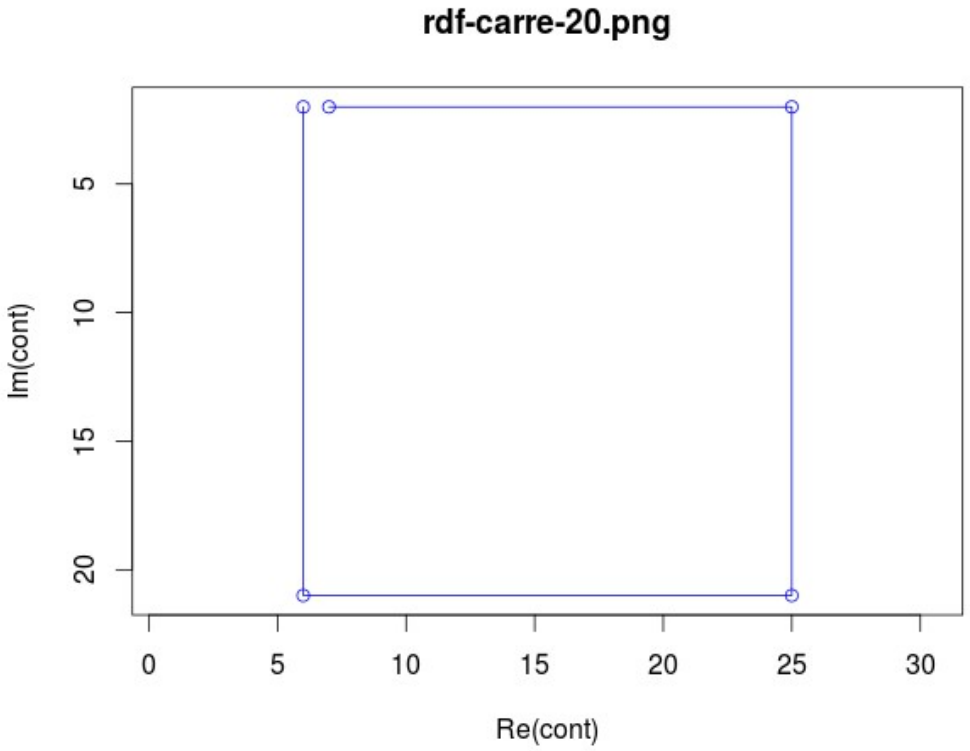


Illustration 4: contour d'un carré avec l'algorithme de la corde

Triangle :

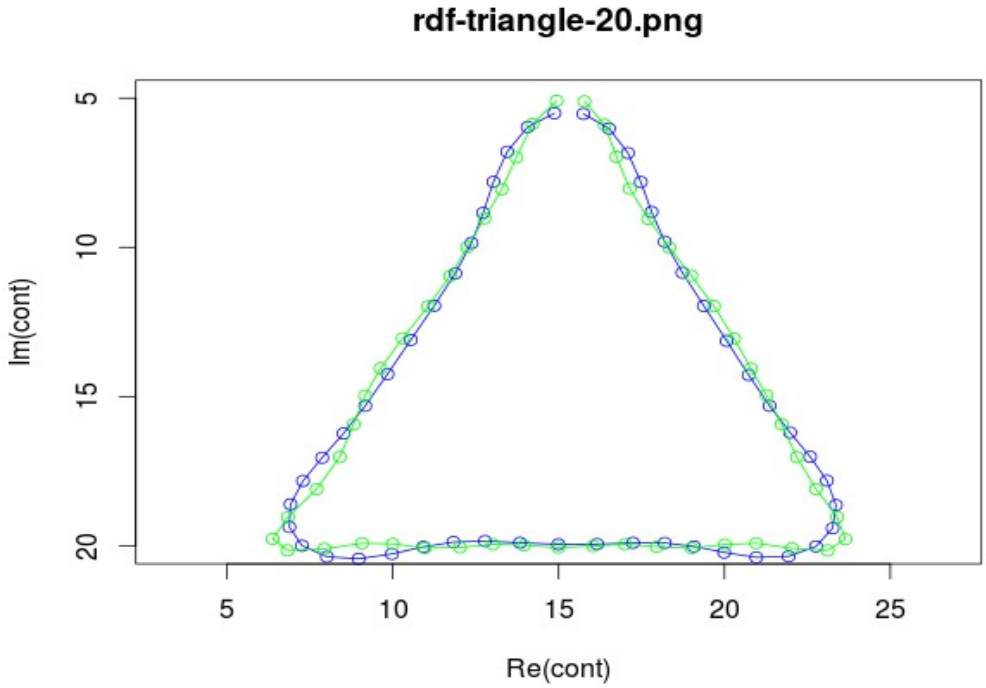


Illustration 5: contour d'un triangle avec la méthode de Fourier

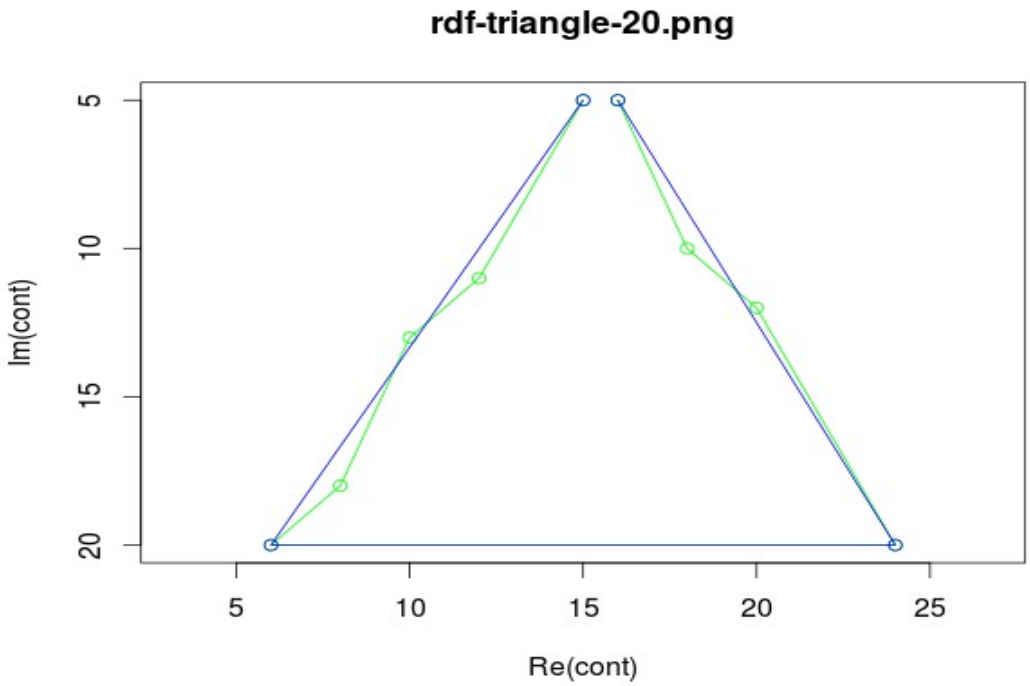


Illustration 6: contour d'un triangle avec l'algorithme de la corde

Concernant ces trois premières formes, on constate que l'algorithme de la corde est beaucoup plus efficace. En effet, avec très peu de points, on obtient un contour parfait de la forme, tandis qu'avec l'algorithme de Fourier, nous obtenons plus de points pour un contour reconnaissable

mais plus approximatif de la forme.

Croix :

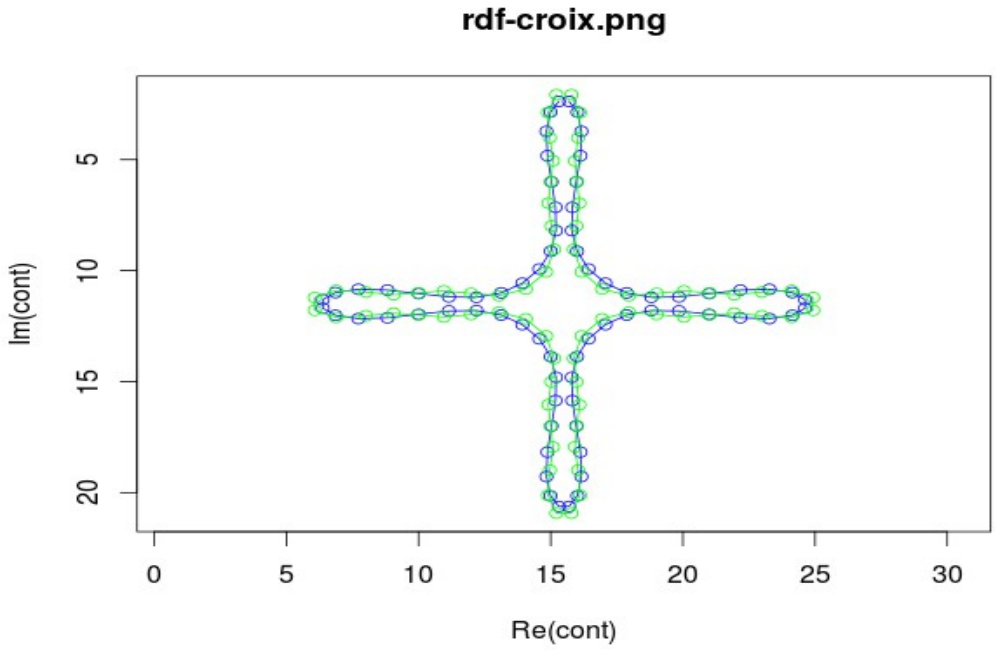


Illustration 7: contour d'une croix avec la méthode de Fourier

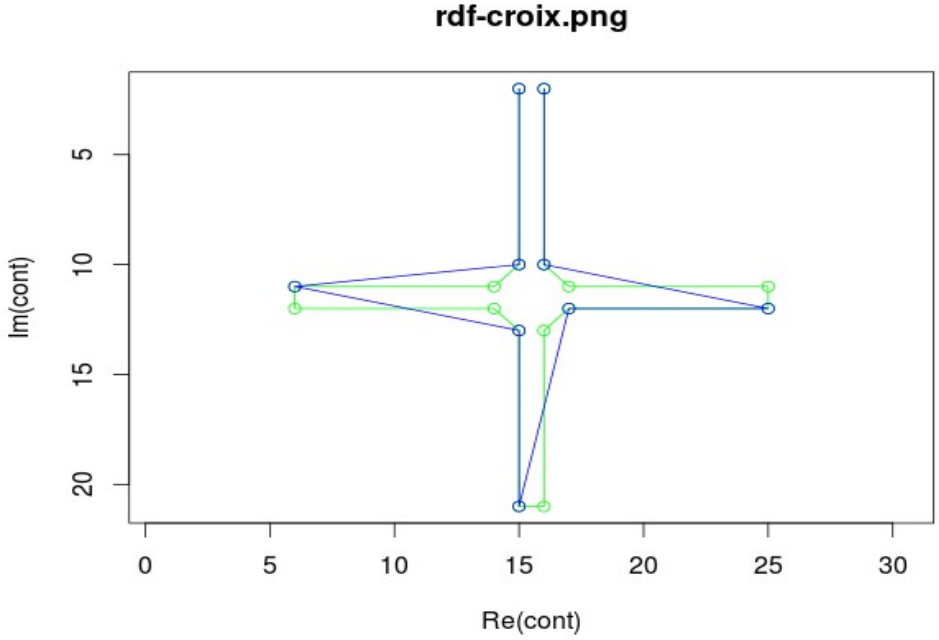


Illustration 8: contour d'une croix avec l'algorithme de la corde

Pour la croix, on obtient toujours un meilleur résultat avec l'algorithme de la corde. On remarque qu'en utilisant une distance maximal de 1 pixel (en bleu) , le contour est assez approximatif. En la réduisant à 0.5 pixel (en vert), on obtient un contour très reconnaissable et plus léger qu'avec la méthode de Fourier qui utilise beaucoup de points pour un résultat assez imprécis.

Patatoïde :

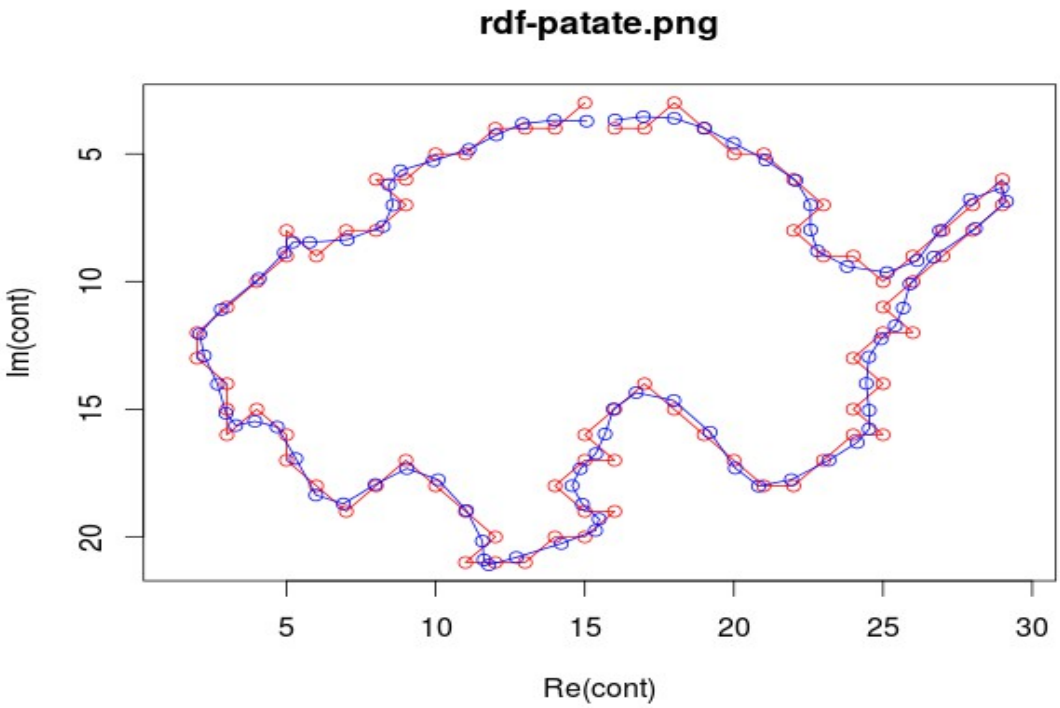


Illustration 9: contour d'une patate avec la méthode de Fourier

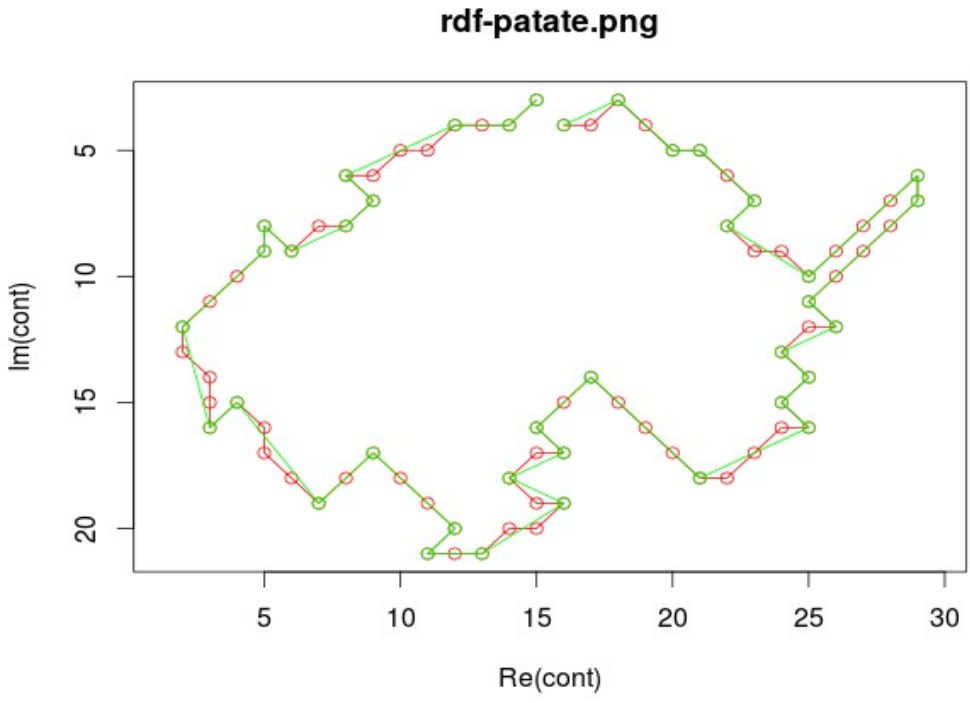


Illustration 10: contour d'une patate avec l'algorithme de la corde

Enfin, pour la patatoïde, Les deux algorithmes nous renvoient des contours assez proches de la forme originel. La méthodes des descripteurs de Fourier semble donc plus efficace sur des formes plus complexes.

Conclusion

Pour conclure, nous pensons que les deux algorithmes semblent être destinés à des formes différentes. L'algorithme de la corde semble être particulièrement efficace sur des formes simples comme le carré ou le triangle. Quant aux descripteurs de Fourier, ils semblent être plus compétent sur des contours complexes comme les patatoïdes tandis que sur des formes simples, ils utilisent trop de points pour un contour approximatif de la forme.