CR2/3 Forgery Breaker

Pour la partie Copy-move sans machine learning nous avons décidé de partir sur une méthode utilisant un changement de coordonnées log-polar. C'est une méthode par blocs qui utilise la fft après un changement de coordonnées qui permet de « se débarrasser » des rotations et des changements de taille.

Nous suivrons l'implémentation de [1] et [2] :

1) Changement en coordonnée log-polar :

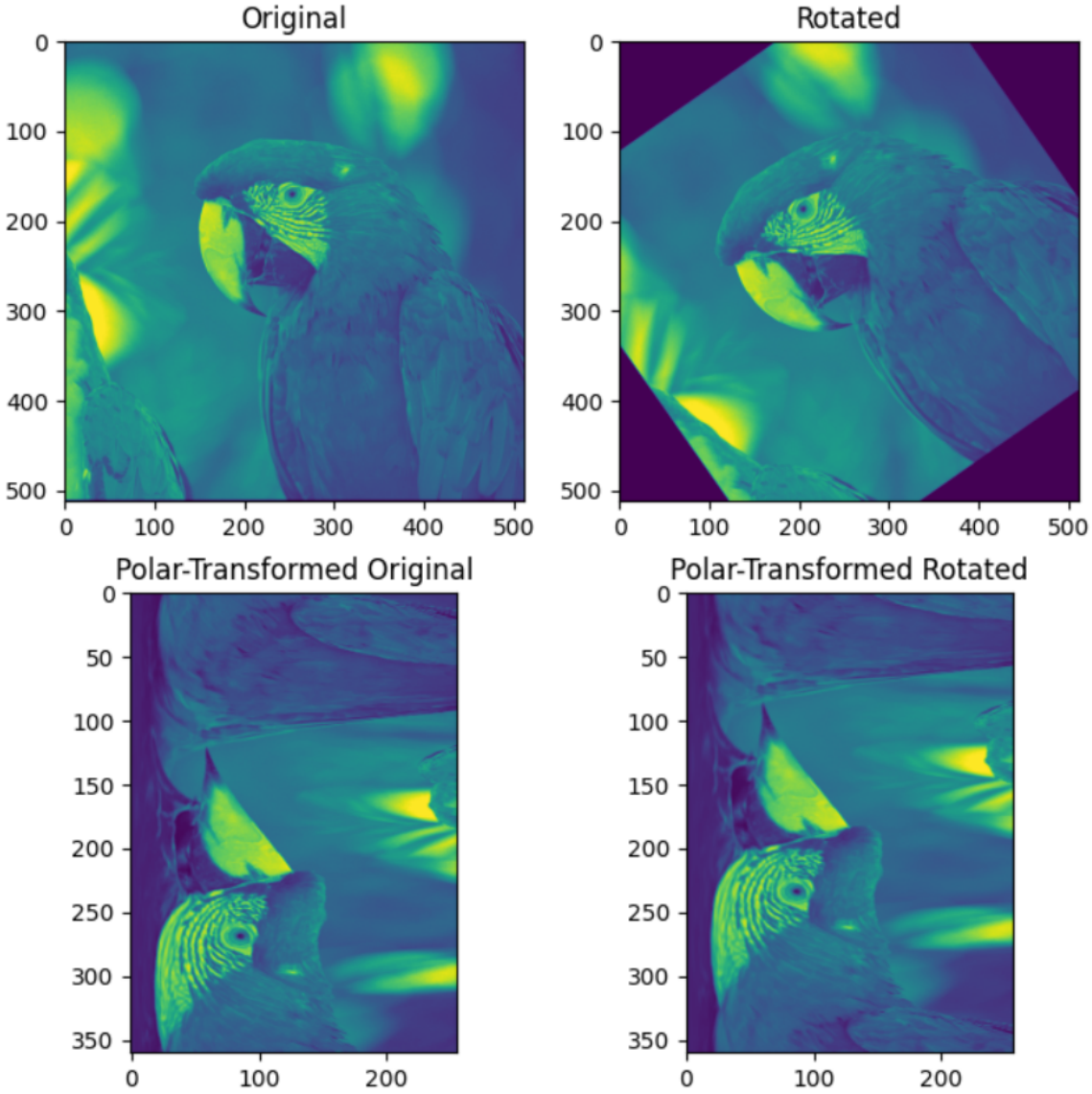
Le changement en log-polar s'effectue en passant de (x,y) dans w*h à p,t avec :

$$\rho = \log \sqrt{(x - x_0)^2 + (y - y_0)^2}$$

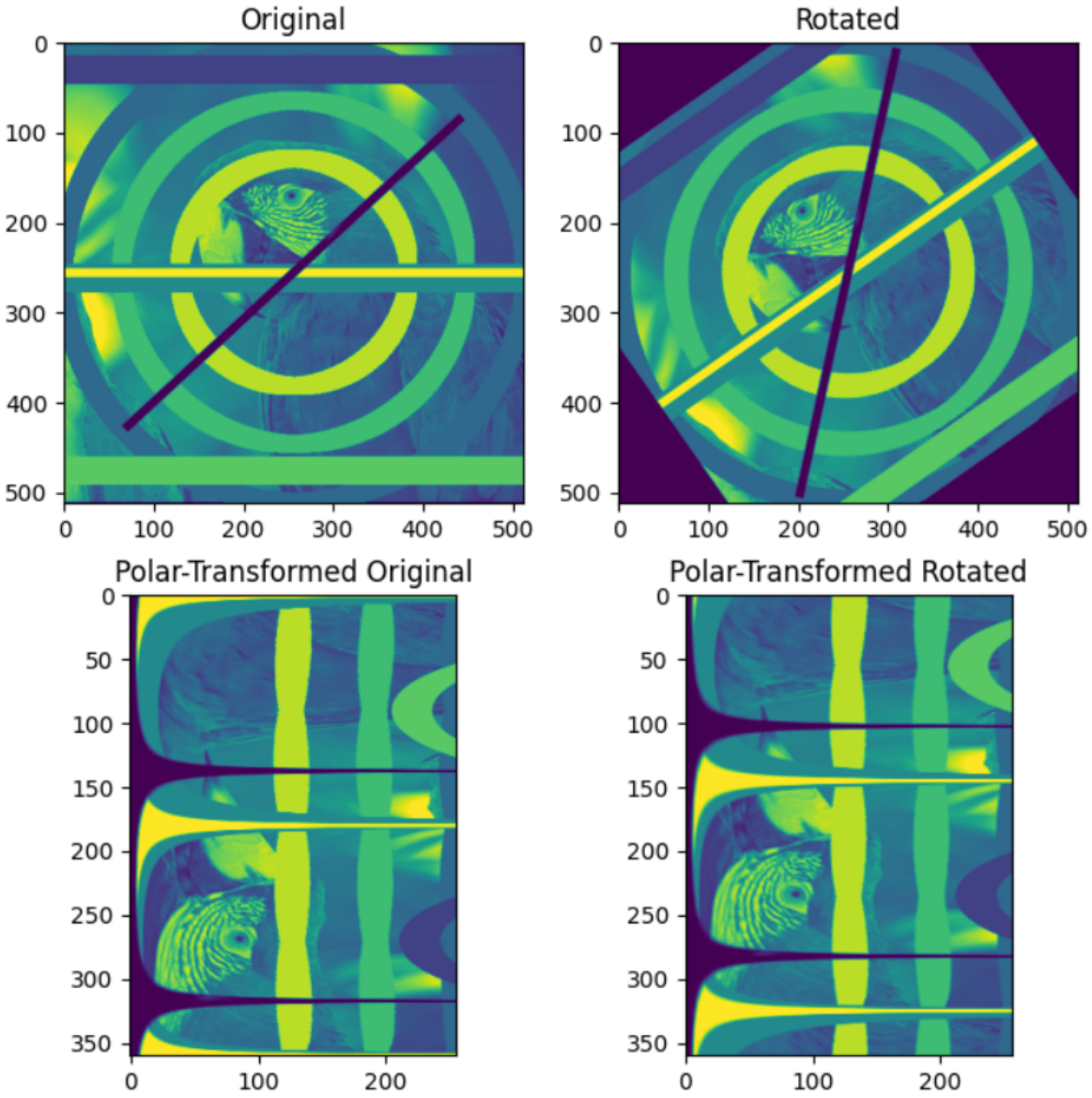
$$\theta = \arctan\left(\frac{y - y_0}{x - x_0}\right), \text{when } x \neq x_0$$

où (x_0,y_0) est le centre de l'image. ro est donc le logarithme de la distance de (x,y) à (x_0,y_0) tandis que teta est l'angle entre le milieu de l'image et la demi-droite $[(x_0,y_0),(x,y))$.

En appliquant cette transformation sur le planY de l'image « 14_Perroquet.ppm » et d'une copie qui a subi une rotation on obtient :


Original
Rotated
Polar-Transformed Original
Polar-Transformed Rotated

sur une image de test on obtient ceci :

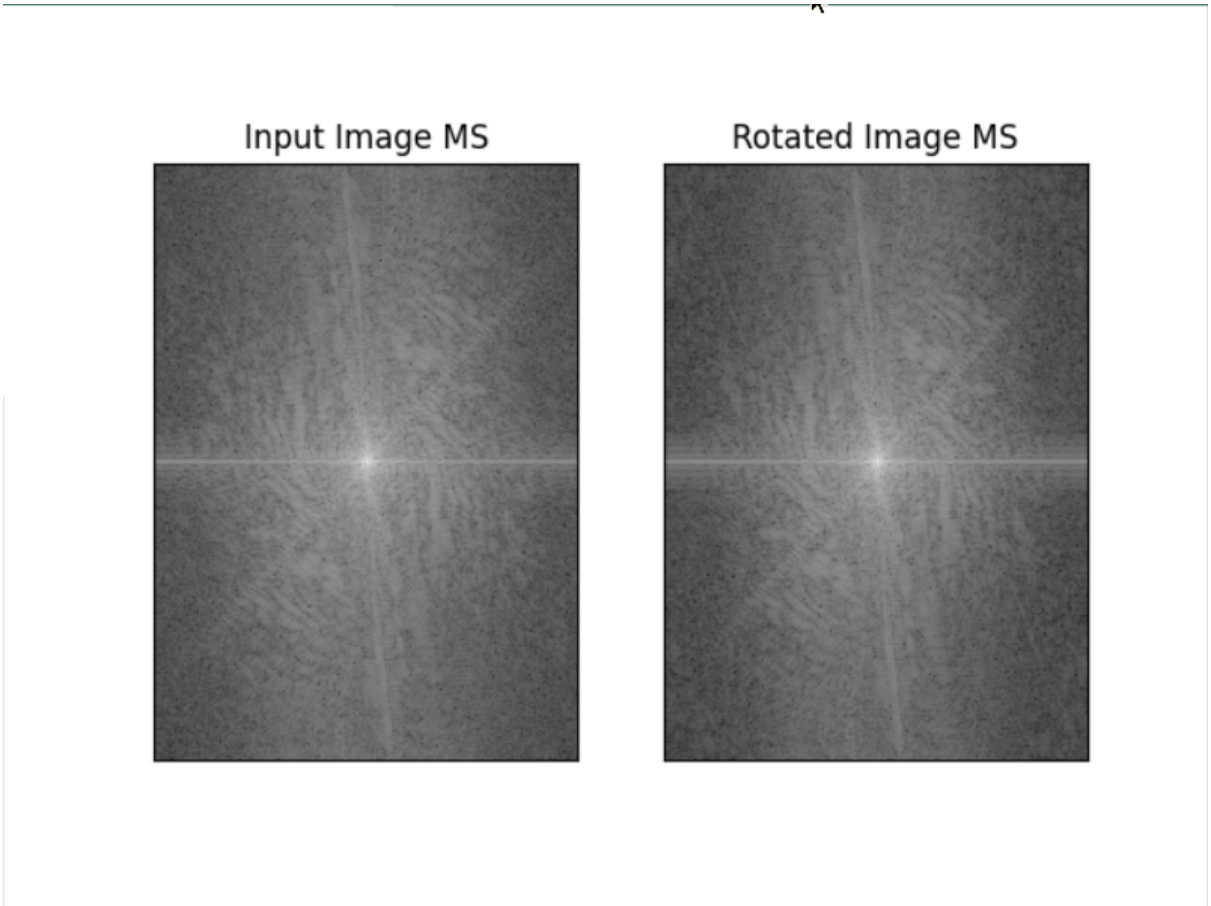


On remarque que dans les coordonnées log-polar les rotations et les scaling deviennent des translations.

2) Transformée de Fourier.

La transformée de Fourier discrète permet d'analyser un signal, ici elle nous permet de comparer deux images en coordonnées log-polar.

Par exemple la transformée de Fourier des deux images précédentes (après décalage) donne ceci :



Soit S l'image originale et S l'image transformée par rotation ou redimensionnement, alors

$$R\left(\rho_{\mathrm{R}}, \theta_{\mathrm{R}}\right) = S\left(\rho_{\mathrm{S}} + \log \lambda_0, \theta_{\mathrm{R}} + \theta_0\right)$$

dans les coordonnées log-polar. ($logLambda_0$ est la distance entre les deux points tandis que $Teta_R$+$Teta_0$ est la somme des deux angles).

En prenant la transformation de Fourier des deux membres on obtient :

$$F_{\mathrm{R}}(u,v) = F_{\mathrm{S}}(u,v)\exp\left[2\pi j\left(u\log\lambda_0 + v\theta_0\right)\right]$$

On définit le « normalised Cross-spectrum » G(u,v) (où u et v sont deux images) par :

$$G(u,v) = \frac{F_{\mathrm{R}}(u,v)F_{\mathrm{S}}^{*}(u,v)}{\left|F_{\mathrm{R}}(u,v)F_{\mathrm{S}}^{*}(u,v)\right|} :$$

où l'étoile* désigne le conjugué complexe. Comme la partie réelle des transformées de Fourier est la même (voir équation précédente) alors on a (?):

$$G(u,v) = \exp\left[-2\pi j\left(u\log\lambda_0 + v\theta_0\right)\right]$$

Si G(u,v) forme une sinusoïde alors U est la translation de V (dans les coordonnées log-polar) et la transformée de Fourier inverse de G(U,V) est une fonction delta (une fonction présentant un pic).

Cette propriété va nous permettre de repérer les copy-moves dans l'image.

3) L'algorithme.

L'algorithme fonctionne par « blocs » circulaire de rayon fixé. Il faut au préalable choisir une taille de rayon cohérente avec notre image : si la taille est trop grande, nous risquons de passer à côté de certaines duplications, et si elle est trop petite nous risquons d'obtenir des faux positifs.

1er étape : On découpe l'image en blocs circulaires de rayon 10px.

2ème étape : On applique la transformée log-polar sur chaque cercle et on applique la transformée de Fourier.

3ème étape : Pour chaque cercle, on crée une liste des transformées de Fourier inverse des « normalised Cross-spectrum » entre chaque paire de cercle avec tous les autres cercles.

4ème étape : On garde les paires qui ont une valeur maximale au-dessus d'un certain seuil (moyenne absolue des F^-1 G(u,v)). Et dans ce cas, on déclare que les deux sont identiques à rotation et redimensionnement près .

4) L'implémentation.

L'état actuel de l'implémentation peut être testé avec le script "main.py" présent dans le dossier "LogPolar_Detector", il présente actuellement la séparation en blocs d'une et la transformation en coordonnées log-polar de ces blocs.

5) Notre avancée :

L'implémentation de l'algorithme avance, nous pourrons bientôt passer à la partie test et ensuite à la partie réseaux de neurones.

Pour la répartition de la suite des tâches, Guillaume s'occupera de finir l'implémentation de l'algo tandis que Yann commencera la partie réseau de neurone. Une fois l'implémentation faite nous effectuerons différents tests sur les bases de données du CR1.

Annexe : sources utilisées

[1] : https://link.springer.com/chapter/10.1007%2F978-3-642-16693-8_11
[2]https://www.researchgate.net/publication/220139494_Log-Polar_Based_Scheme_for_Revealing_Duplicated_Regions_in_Digital_Images