CR_5

Avancée

Afin d'essayer d'obtenir un résultat plus probant, nous avons essayé d'augmenter la taille des blocs de 16 à 20. Même si la qualité de l'image déchiffrée est légèrement amoindrie comparée à une image dont la taille des blocs serait restée à 16 (PSNR : 38,878883), le résultat reste tout de même très convenable avec cette nouvelle taille de blocs (PSNR : 37,744373).

Image cadrée avec N = 20

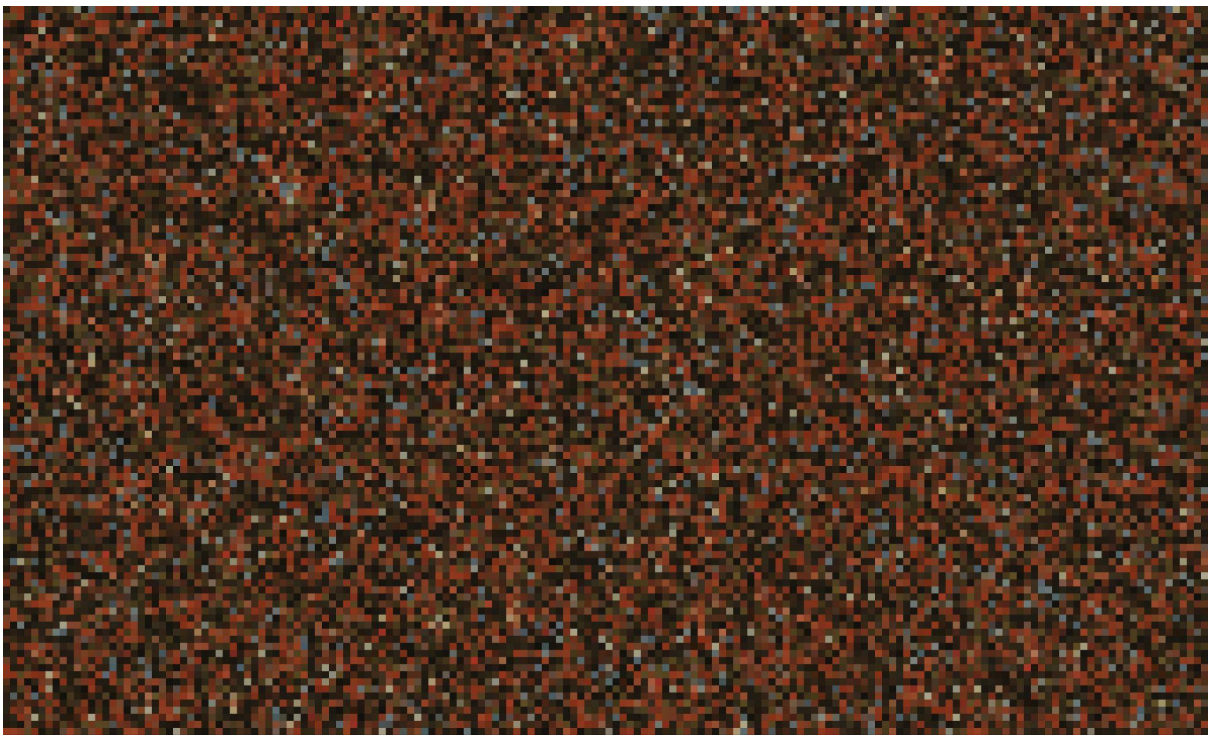

Image après retrait du cadre avec N = 20

Malheureusement, cette solution n'a pas été plus concluante.

Ainsi, sur un conseil de Monsieur Puech, nous sommes revenus à N = 16 et nous avons agrandi le "QR code" (damier en bas à droite du cadre) et le cadre, ces derniers n'étant pas assez gros pour permettre de retrouver facilement la taille d'un bloc. Nous sommes donc passées à un damier de 8*8 séparé du bord pour une meilleure visibilité. En ajoutant un fond blanc, nous obtenons une nouvelle image de base :

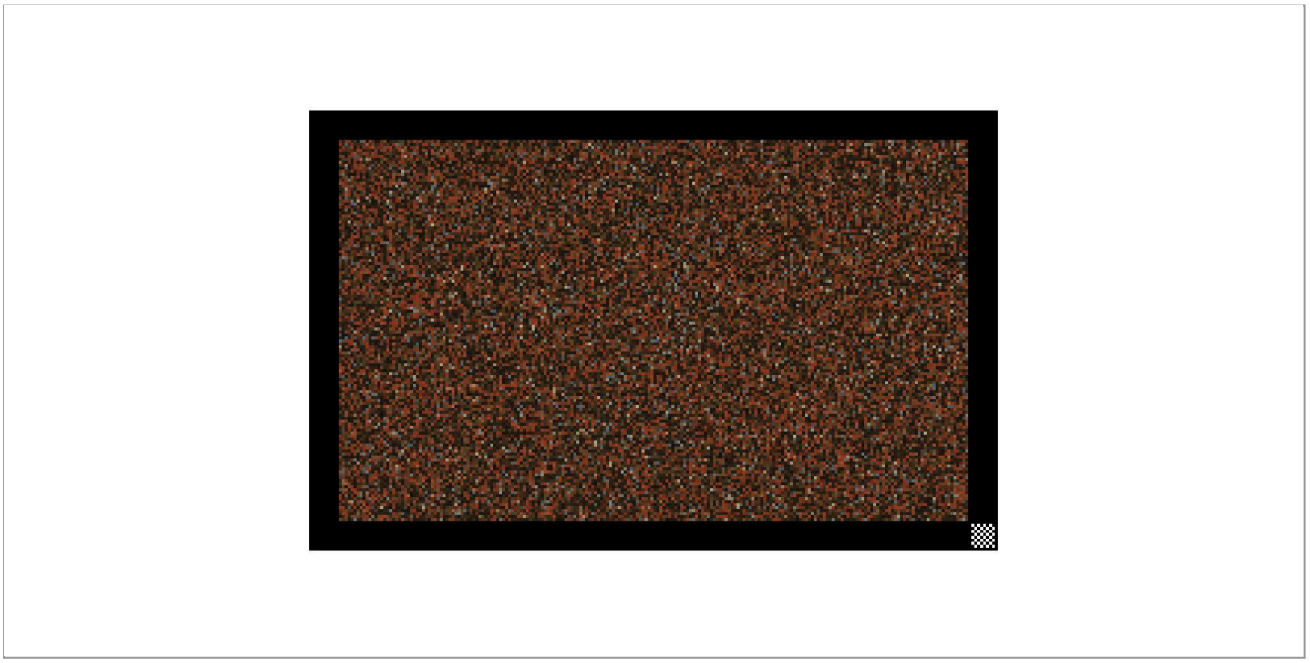

Image test avec cadre et fond

Alors, cette fois-ci, on remarque immédiatement que le nombre de blocs de l'image chiffrée est le même que le nombre de blocs de l'image recalée (ce qui est bon signe), et voici le résultat du déchiffrement de l'image recalée :

Image recalée déchiffrée

On note sans difficulté que le résultat n'est pas très éloigné de ce qu'on recherchait, simplement, pour une raison qui nous échappe encore, certains blocs sont devenus noirs.

Après correction de la dernière colonne avant déchiffrement qui était une ligne de pixel noir, on obitent :

Image recalée déchiffrée tentative 2

Et c’est enfin une victoire ! 🎉🎉🎉🎉🎉🎉🎉🎉

Maintenant, il nous faut tester cet algorithme avec une image imprimée !

Réflexion sur le recalage avec rotation :

Dans ce cas-là, il faut également faire une étape pour identifier la rotation lors de l’étape d’identification du cadre et la corriger. Toutefois, en appliquant une transformation affine comme nous le voulions à la base, il y aura probablement des problèmes liés au fait que nous obtenons, en appliquant une transformation affine sur des entiers (positions du pixels dans l’image), des flottants. Il faudra donc faire une correspondance entre les flottants et les entiers des pixels.