Musée Sécurisé Virtuel

Projet Image - M2 IMAGINE 2022

Maëva DALILA

Contexte et Objectif

Chiffrer une image à l'aide d'une clé secrète puis la déchiffrer

Outils : OpenCV C++

Chaîne de traitement

4 étapes :

1) Création de l'oeuvre à l'aide d'une clé secrète
2) Acquisition et détection de la feuille
3) Détection de l'oeuvre
4) Déchiffrement de l'oeuvre à l'aide de la clé secrète

Création de l'oeuvre à l'aide d'une clé secrète

Chiffrement par permutation

```
// Algorithme de Fisher-Yates
/*
pour i de n - 1 descendant_à 1 :
      j ← nombre aléatoire entier 0 ≤ j
≤ i
      échanger a[j] et a[i]
*/
```

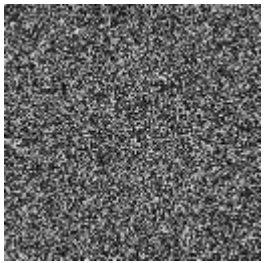
Déchiffrement

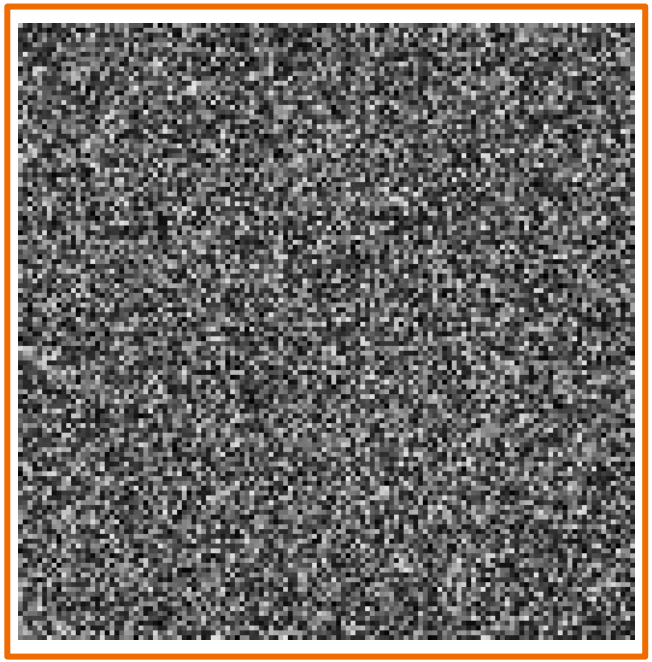
Recréer la séquence de permutation à l'aide de la clé

Reconstruire à l'aide des indices et du vecteur

Création de l'oeuvre chiffrée à l'aide d'une clé secrète

Input : Image moyennée par bloc





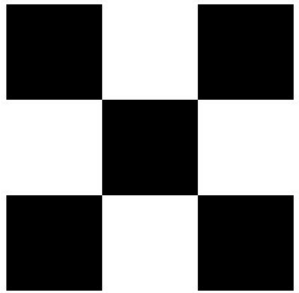
Output : Image Chiffrée par permutation

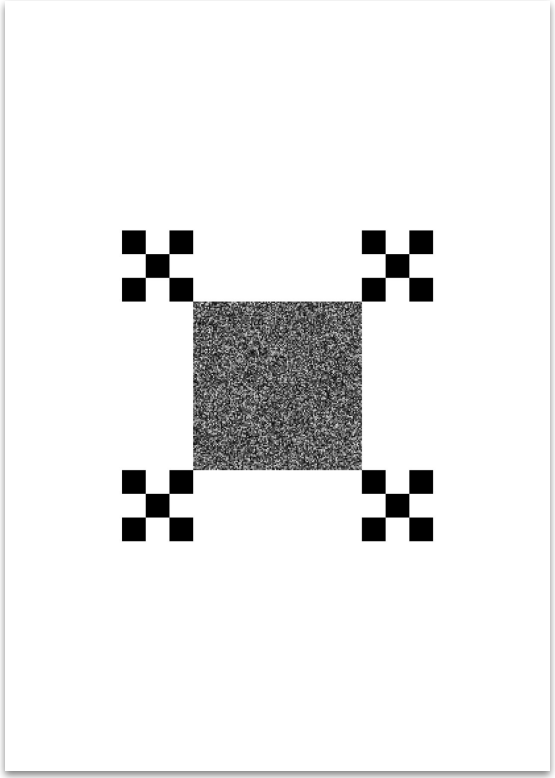
Création de l'oeuvre à afficher

- Création d'une image de taille A4 DPI = 300

  (W,H) = 2480 x 3508

- Faciliter la détection : Choix d'un pattern
  5 carrés de 72 pixels de côté





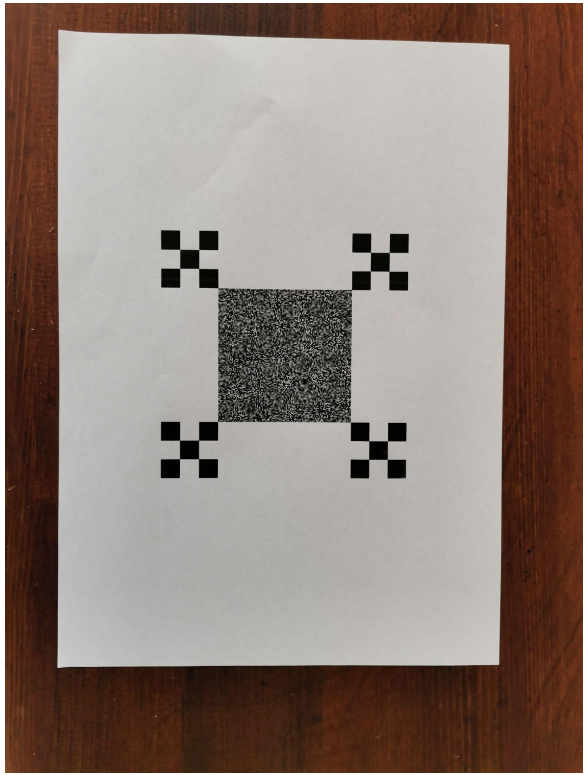
Acquisition et détection de la feuille

OpenCV (C++)

Attributs image :

- Zone très texturée = oeuvre chiffrée
- Patterns
- Bords de la feuille

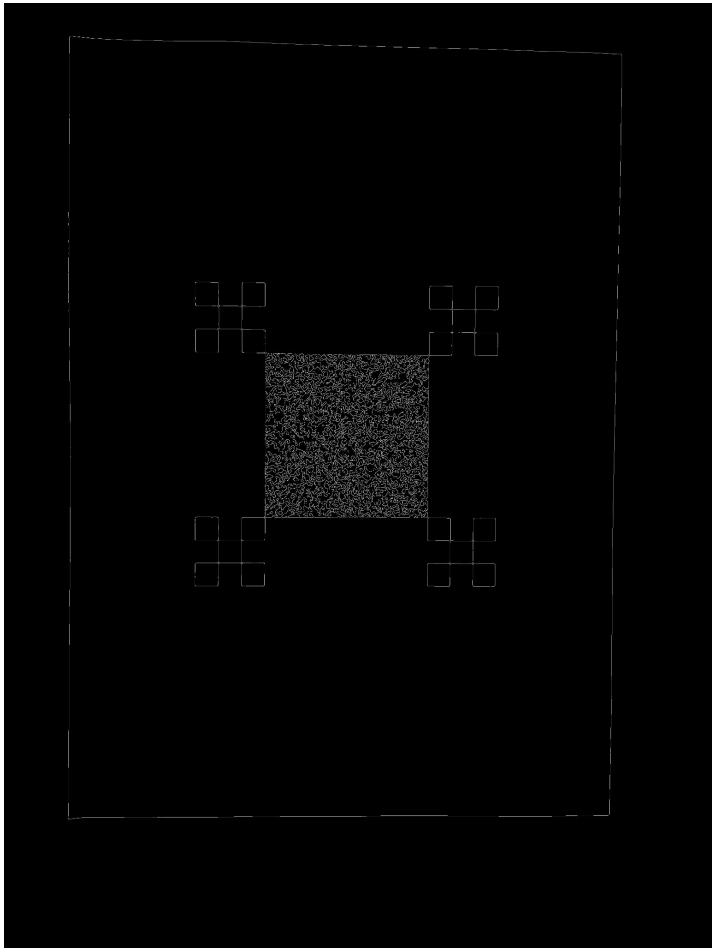
Détecter la feuille et corriger la perspective



Acquisition et détection de la feuille

Détection de rectangles

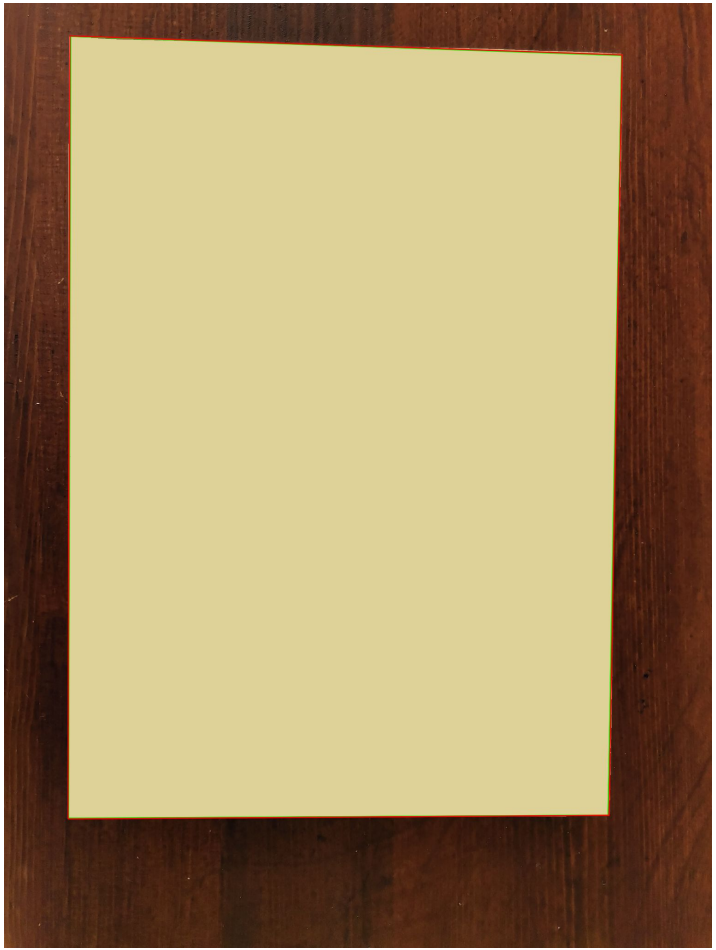
- Median Filter
- Filtre de Canny => edges
- Dilatation => edges continus
- findContours : détermination des contours
- Feuille = surface la plus grande



Acquisition et détection de la feuille

Détection de rectangles

- Median Filter
- Filtre de Canny => edges
- Dilatation => edges continus
- findContours : détermination des contours
- Feuille = surface la plus grande



Acquisition et détection de la feuille

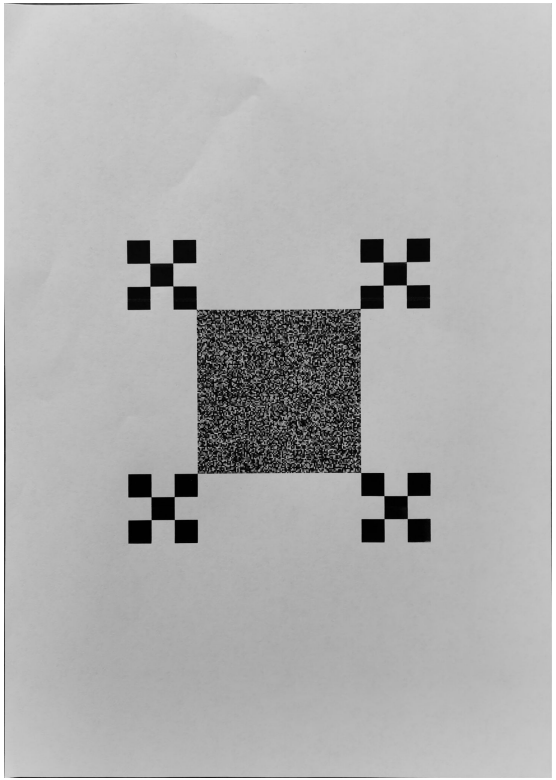
Détection de rectangles

- Median Filter
- Filtre de Canny => edges
- Dilatation => edges continus
- findContours : détermination des contours
- Feuille = surface la plus grande

Correction Perspective

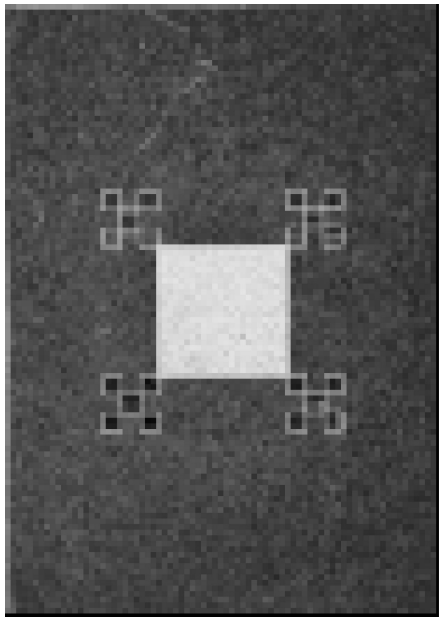
- $$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- 4 points feuille (Repère feuille)
- 4 points feuille A4

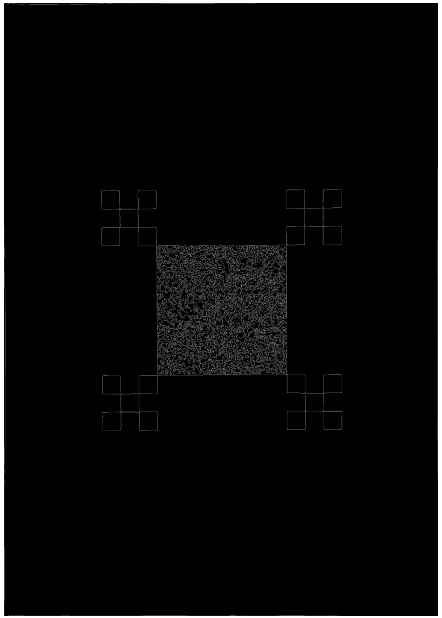


Détection de l'oeuvre (Méthode 1)

Eviter le pattern : calculer l'entropie de l'image



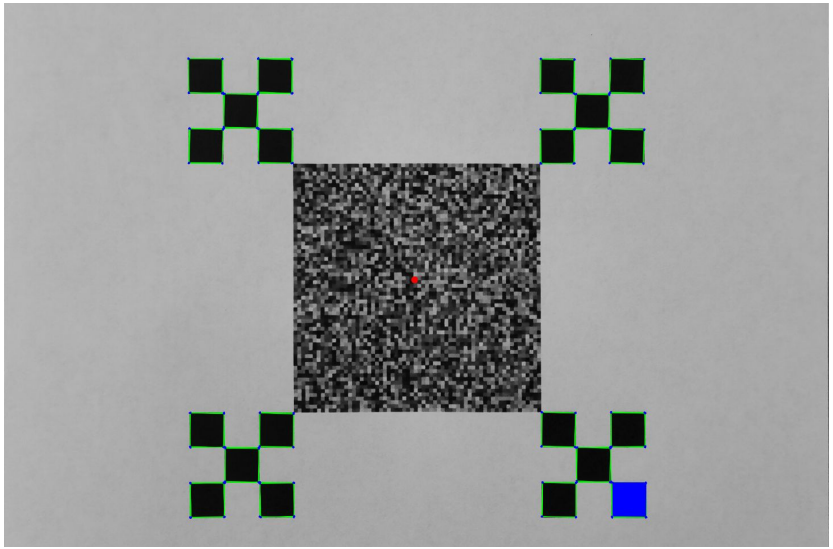
Histogramme des contours détectés par le filtre de Canny : seuil



Détection de l'oeuvre (Méthode 2)

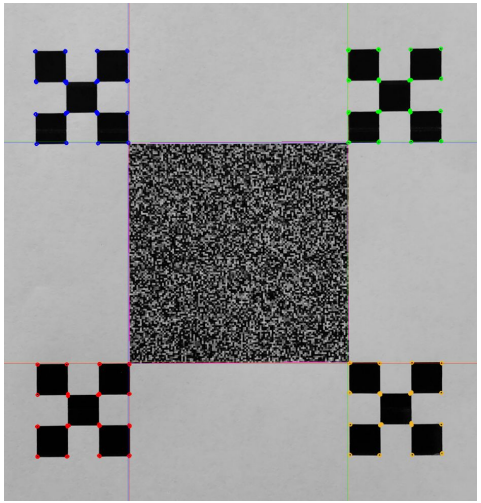
Détection des carrés présents dans l'image

Tri sur les carrés : area ~10% médiane



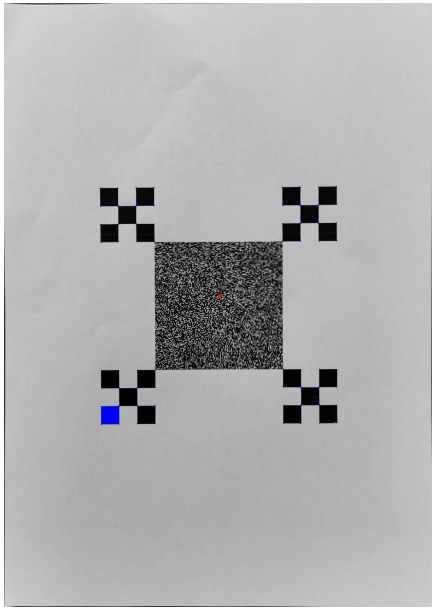
Classes les carrés selon leur coin d'appartenance

Déterminer de 2 droites (H, V) => intersection

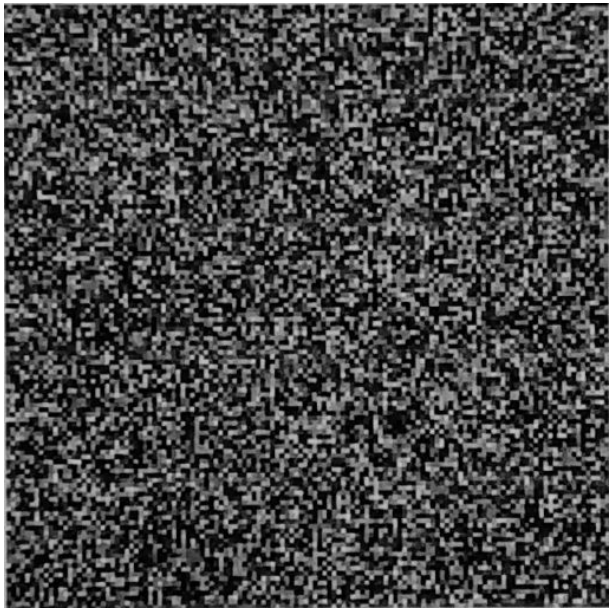


Détection de l'oeuvre

Calcul du ratio (pattern)

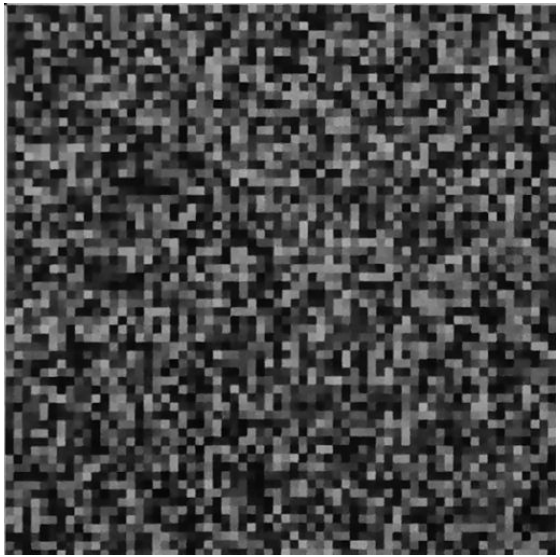


Exemple 4x4 Image rescaled



Déchiffrement de l'oeuvre à l'aide de la clé secrète

Exemple blockSize = 8x8 = 64



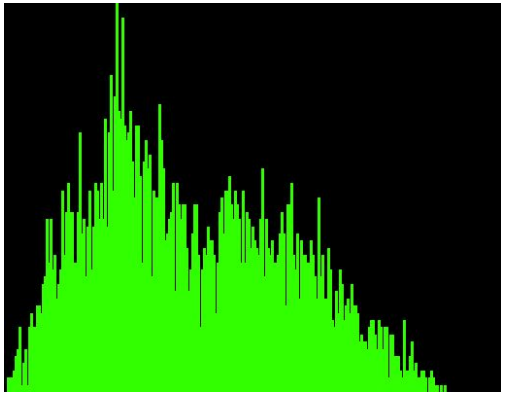
pixel = mean(4 pixels centraux)

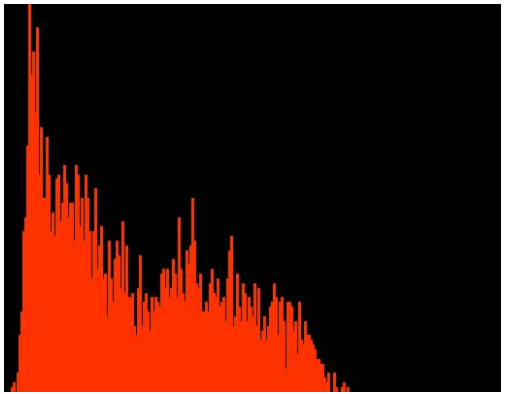
sequence de permutation

Mesures des Performances

pSNR pour un moyennage sur un bloc[8x8]

PSNR(dB)	**Ref**	**Blocky**	**Déchiffrée**
Ref	X	19.48	15.89
Blocky	X	X	18.53

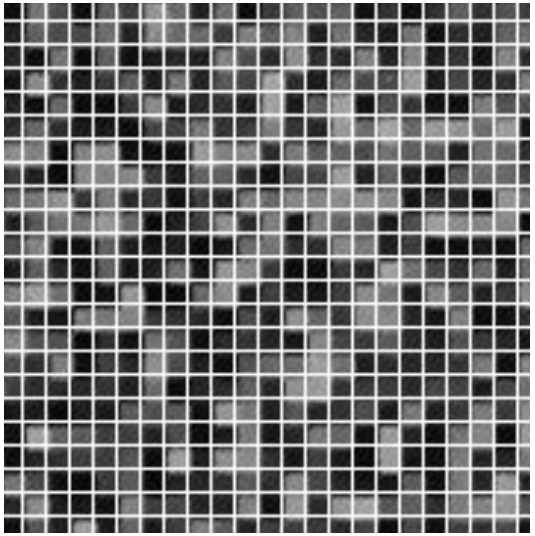




Conclusion

Dans des conditions optimales : bonne reconnaissance

Qualité de l'oeuvre vue par l'utilisateur: très moyenne



Limitations :

Déchiffrement extrêmement dépendant de la précision de la détection

Détection : conditions de prise de vue

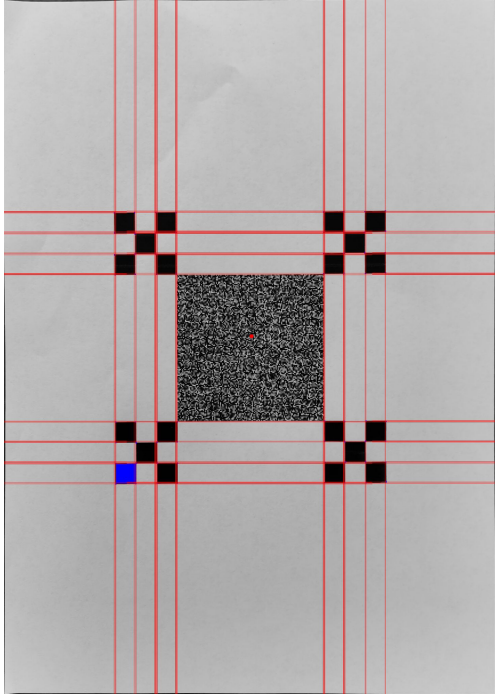
Conclusion

Améliorations:

- Détection : Hough Line transformation à améliorer (tuning à faire ~manque de précision)

- Côté utilisateur : Création d'une application via OpenCV

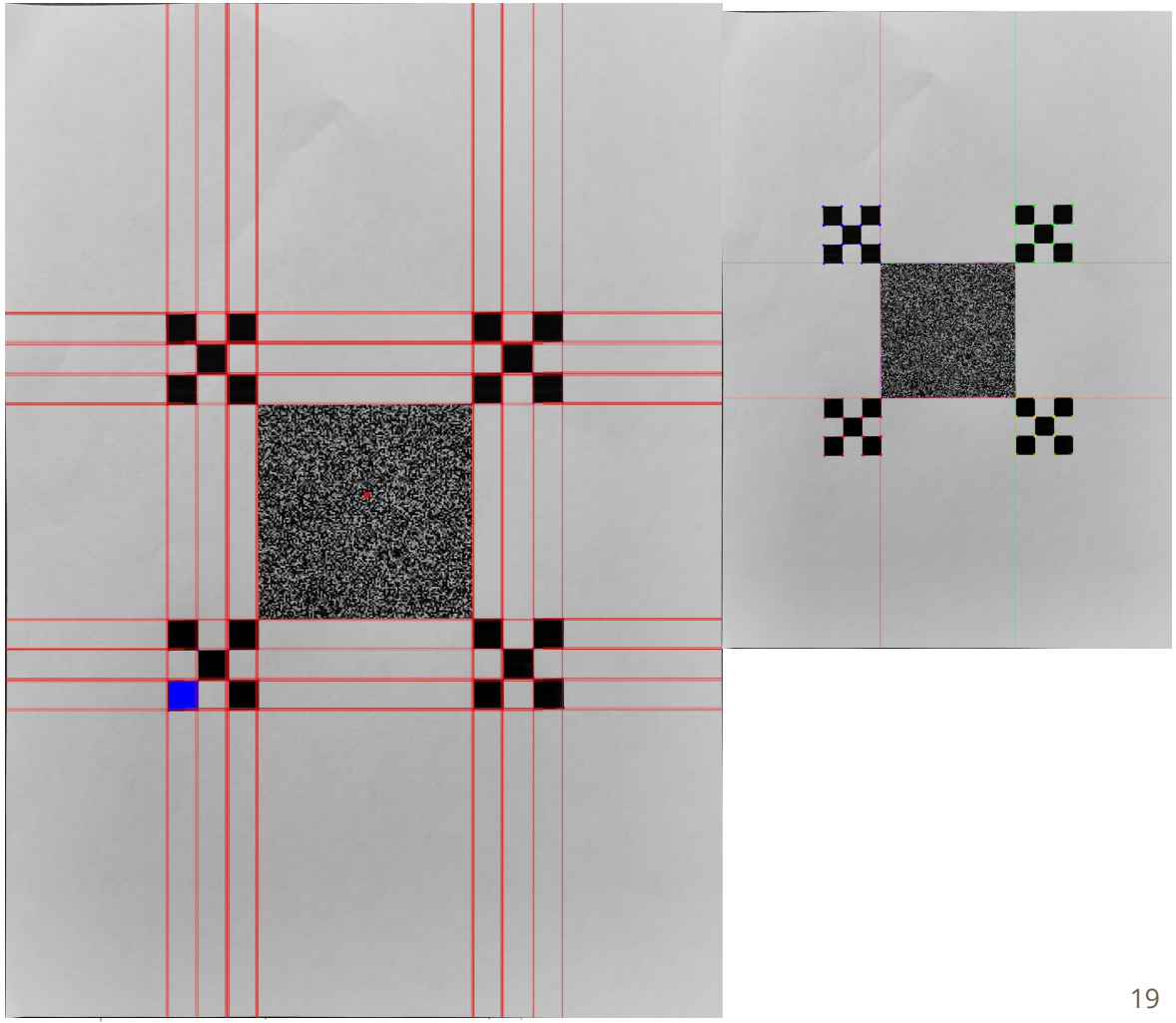
- Couleurs & Palette de couleurs



Merci de votre attention.

Avez-vous des questions ?

Slides Complémentaires
Hough Transformation



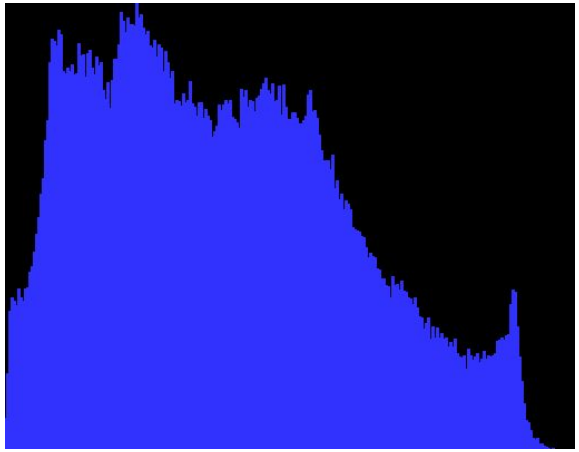
Slides Complémentaires
Histograms

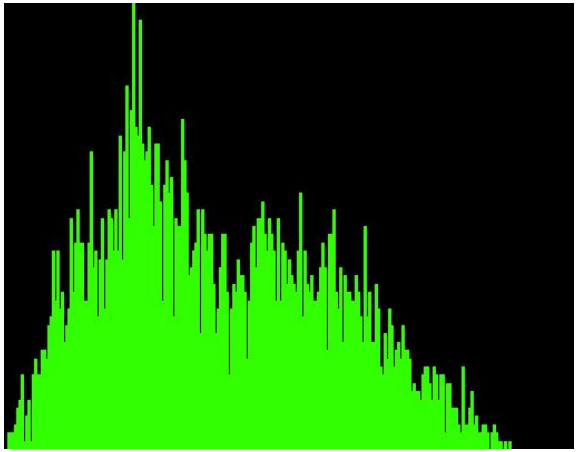
3 histograms

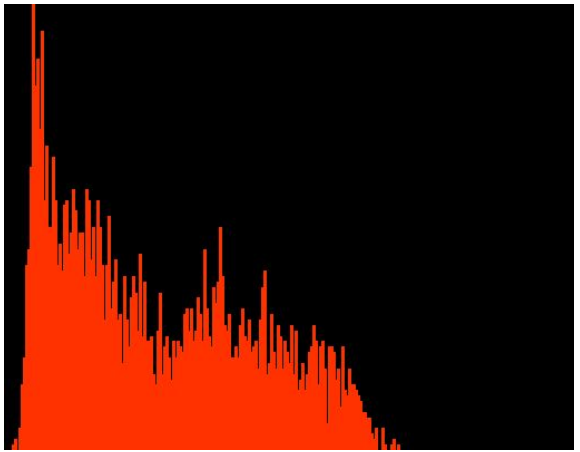
Reference

Reference moyennée par bloc

Reference moyennée par bloc chiffrée puis déchiffrée







Création de l'oeuvre à l'aide d'une clé secrète

Chiffrement par permutation

Etape 1	Etape 2	Etape 3	Etape 4	Etape 5
Input	**Générateur de nombre pseudo-aléatoire**	**Séquence de permutation S**	**Permutation des pixels**	**Déchiffrement : reconstruction de l'image originale**
Définition d'une image à chiffrer I = M x N pixels	Input : Clé secrète K(seed), Type des éléments : position Taille : M*N	Création de la séquence de permutation Utilisation de l'Algorithme de Fisher-Yates	Création d'une image chiffrée par permutation	Input : - clé secrète K - données permutées

Sommaire

- Objectifs
- Chaîne de traitement
- Conclusion
  - Difficultés rencontrées
  - Pistes d’améliorations