TIPE

La stéganographie, comment sécuriser ses informations

BUCHON Valentin N° 29 220

Sommaire

01

Introduction à la stéganographie

Introduction à la stéganographie

Définition: La stéganographie, c'est l'art de dissimuler des données dans d'autres données. Il existe plusieurs techniques différentes qui permettent ce "tour de magie".

Objectif: Utiliser l'outil informatique pour cacher des données dans une images suivant 3 critères

-La robustesse

-La sécurité (et l’invisibilité)

-La capacité

02

Recherches personnelles et naïves

Recherches personnelles et naïves

Tableau de correspondances entre caractères et numéros dans la table ASCII

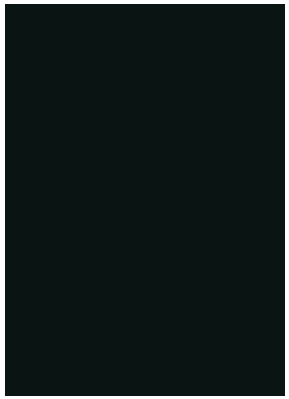
Caractère	Identifiant base 10	Identifiant base 2
T	84	01010100
I	73	01001001
P	80	01010000
E	69	01000101

Exemple: codage de « TIPE » en base 2 :
01010100 01001001 01010000 01000101

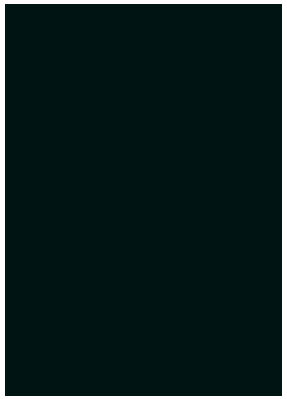
Recherches personnelles et naïves

Utilisation d'une image en nuances de gris:

Insertion du message dans les composantes rouges des pixels, avec un seuil fixé arbitrairement



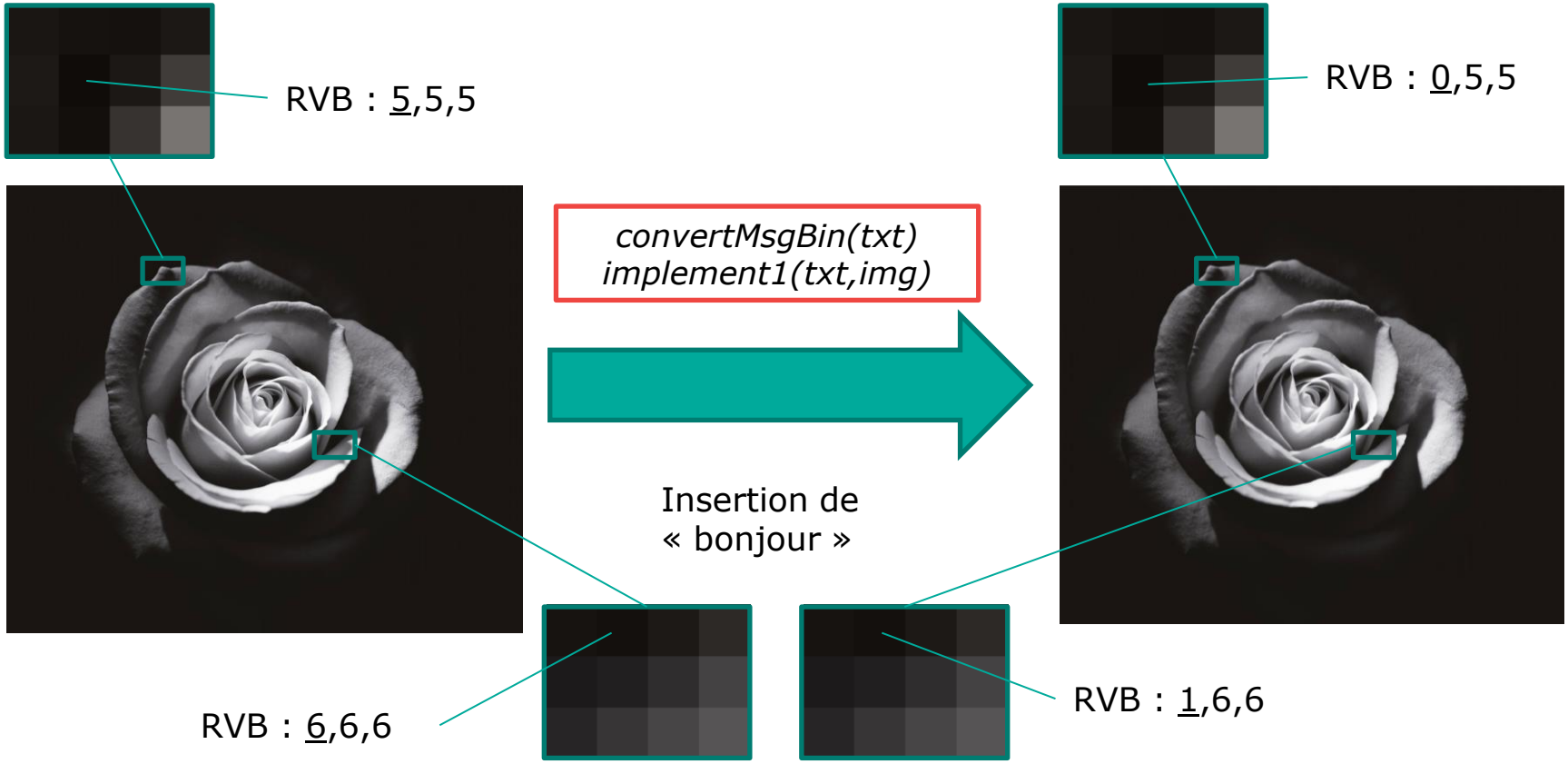
RVB: 10,10,10



RVB: 0,10,10

Différence de couleur peu perceptible par l'œil humain

Recherches personnelles et naïves



Recherches personnelles et naïves

Tableau de correspondances entre caractères et numéros dans la table ASCII

Caractère	Identifiant base 10	Identifiant base 2
T	84	01010100
I	73	01001001
P	80	01010000
E	69	01000101

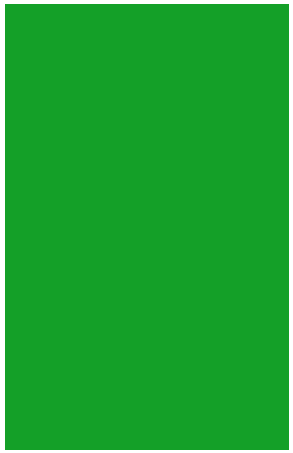
T = 0101/0100 = 5/4

Exemple: Codage de « TIPE » avec des demi octet :
5 4 4 9 5 0 4 9

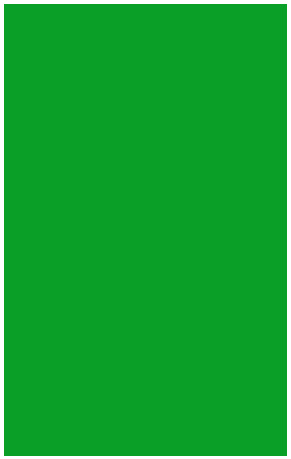
Recherches personnelles et naïves

Utilisation d'une image en couleur:

Insertion des demi octets (valeurs de 0 à 15) dans toutes les composantes RVB, en utilisant un seuil arbitraire



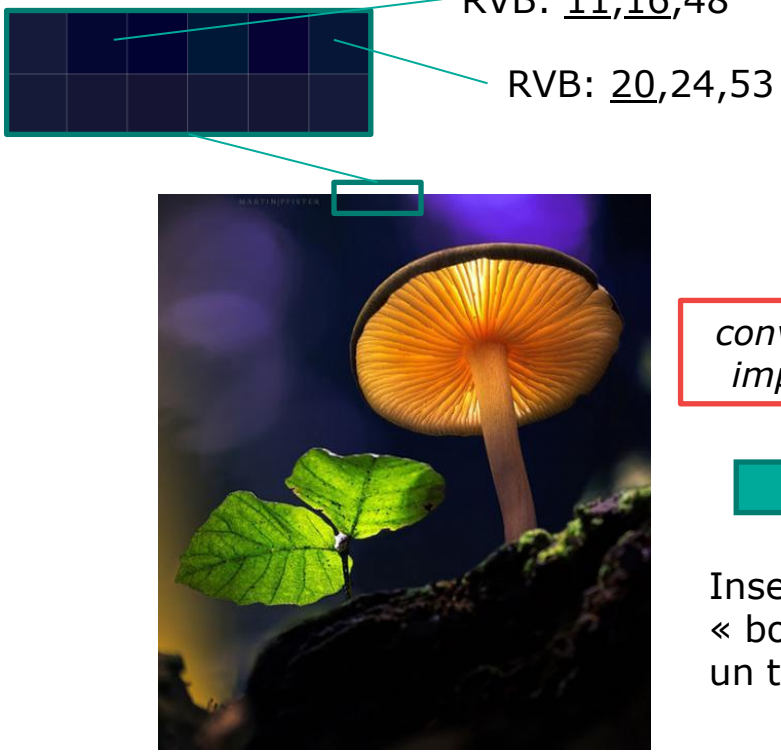
RVB: 20,160,40



RVB: 10,160,40

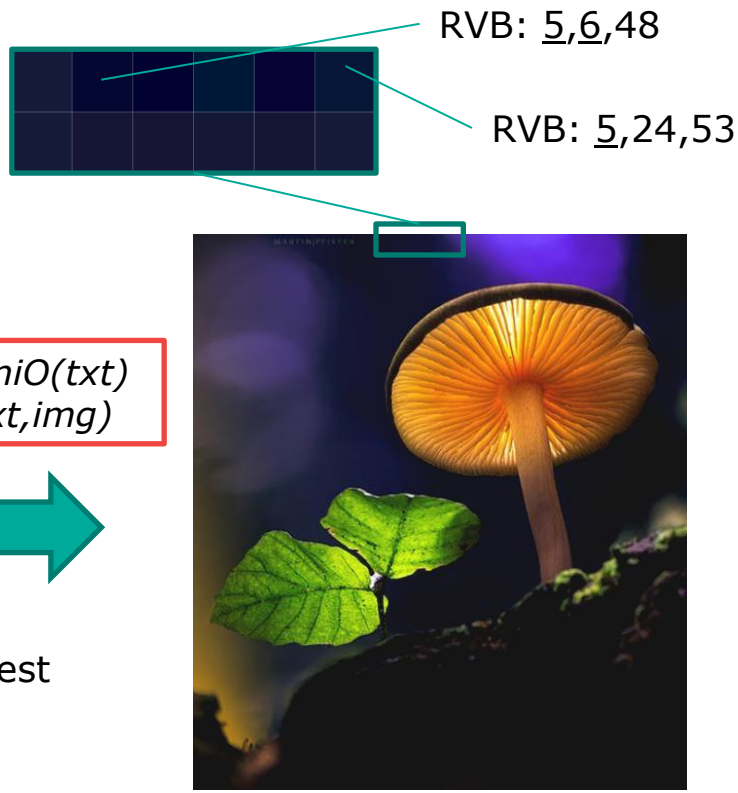
Différence de couleur peu perceptible par l'œil humain

Recherches personnelles et naïves



convertMsgDemiO(txt)
implement2(txt,img)

Insertion de
« bonjour ceci est
un test »



03

Méthode du LSB (Least Significant Bit)

Méthode du LSB

Amélioration de la méthode personnelle:

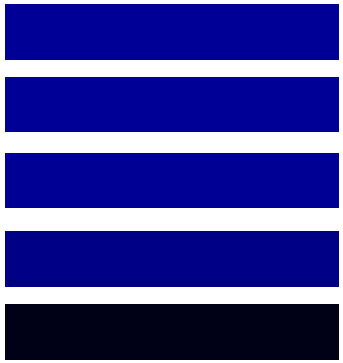
150 = 10010110

151 = 10010111

148 = 10010100

134 = 10000110

22 = 00010110



10010110

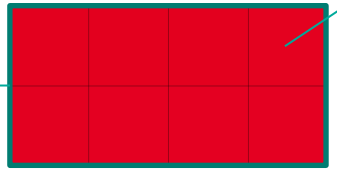
Bits de poids forts

Bits de poids faibles

Méthode du LSB

Image de surface

RVB: 227,0,30



RVB: 225,3,20

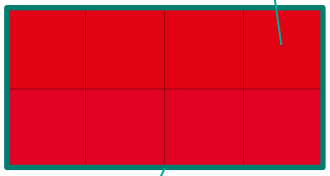
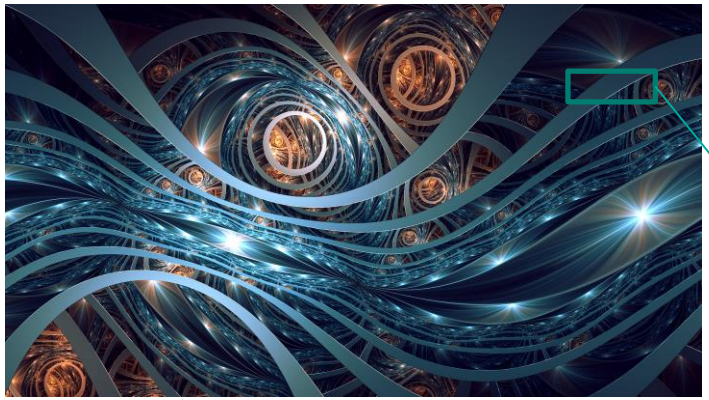
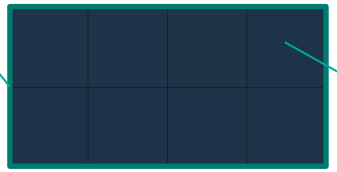


Image finale





RVB: 30,52,72

Image à cacher

Méthode du LSB

RVB: 227,0,30

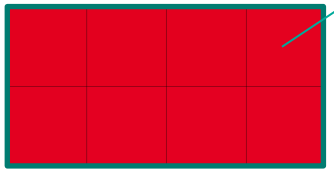


Image de surface

Insertion(img1,img2)

RVB: 225,3,20

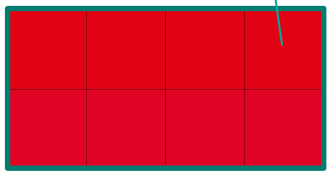


Image finale

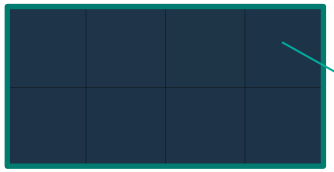


Image à cacher

RVB: 30,52,72

(R) 227 = 11100011
(V) 0 = 00000000
(B) 30 = 00011110

(R) 30 = 00011110
(V) 52 = 00110100
(B) 72 = 01001000

(R) 225 = 11100001
(V) 3 = 00000011
(B) 20 = 00010100

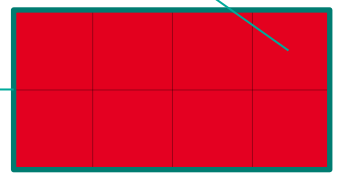
Méthode du LSB

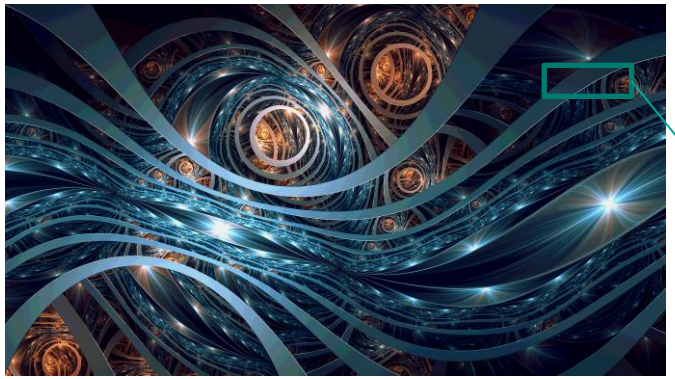
RVB: 225,3,20

Image avec message

Image de surface

RVB: 227,0,30





RVB: 30,52,72

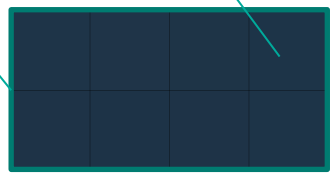


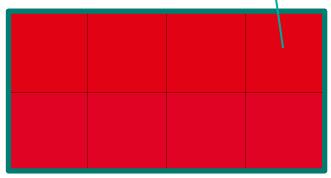
Image cachée

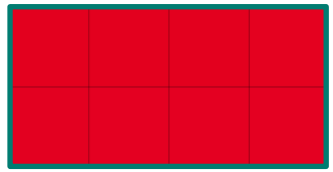
Méthode du LSB

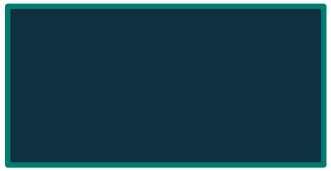
Couleur récupérée

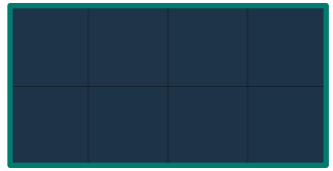
Couleur d'origine

RVB: 225,3,20









(R) 225 = 11100001
(V) 3 = 00000011
(B) 20 = 00010100

desinsertion(img)

(R) 224 = 11100000
(V) 0 = 00000000
(B) 16 = 00010000

RVB: 227,0,30

(R) 16 = 00010000
(V) 48 = 00110000
(B) 64 = 01000000

RVB: 30,52,72

Méthode du LSB

Image de surface

Image à cacher

Limite de la méthode: les dégradés de couleurs et les zones avec peu de détails

Image finale

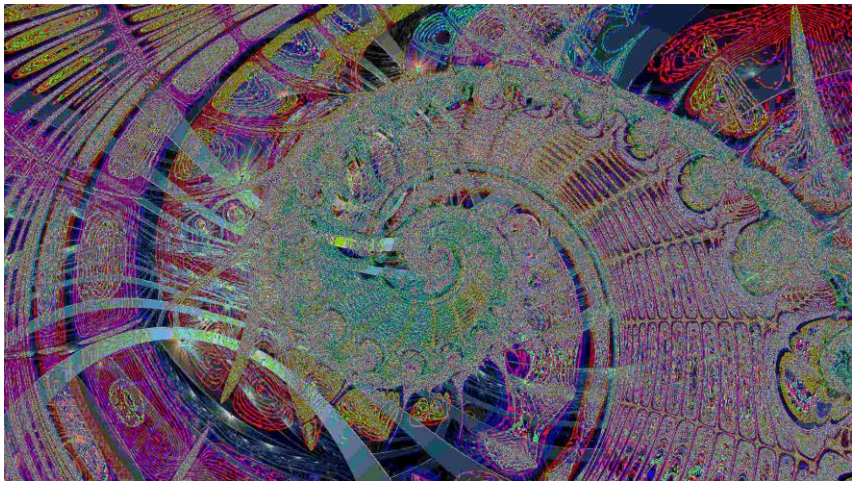
04

Le problème de la compression

Le problème de la compression

Image de surface

Image cachée

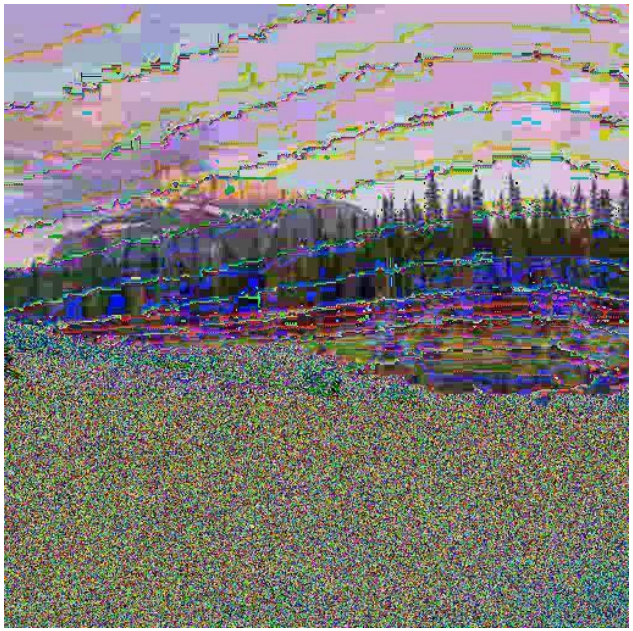


Enregistrement de l'image finale en JPG. Puis utilisation du procédé de récupération de l'image caché

Le problème de la compression

Image de surface

Image cachée



Le problème de la compression

Insertion d'un message via les méthodes naïves, suivie d'un enregistrement de l'image finale en PNG et JPG

RVB: 0,0,0

Format non destructeur
(PNG)

RVB: 72,72,72

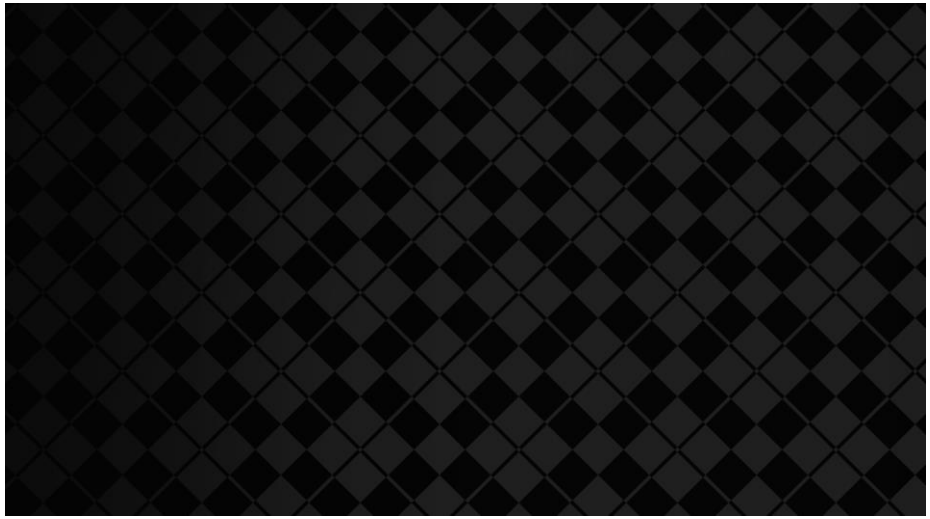
Format destructeur
(JPG)

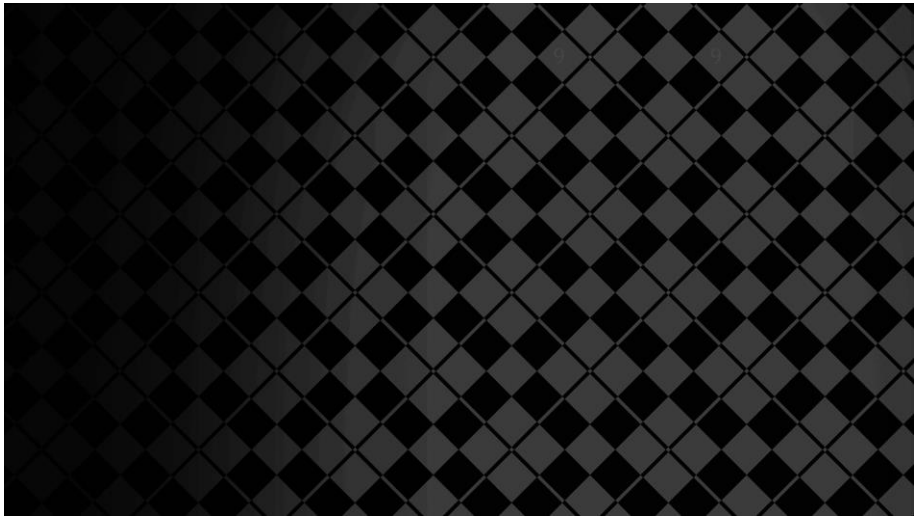
05

Recherches personnelles résistantes à la compression

Recherches personnelles

Insertion d'un motif dans une image





fondu(img1,img2)

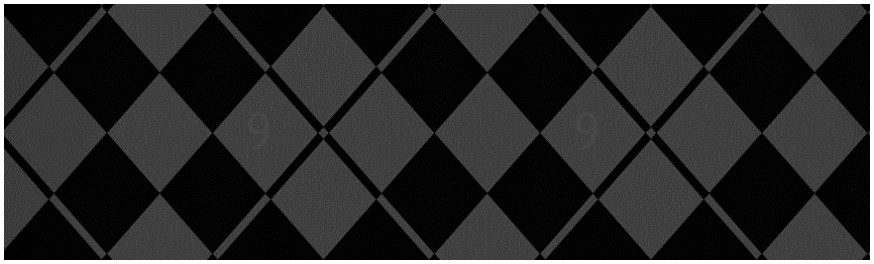
luminosite(img,value)
contrastePlus(img)
contrasteMoins(img)
pitch_color_plus(img,color,value)
pitch_color_moins(img,color,value)

Recherches personnelles

Insertion d’un motif dans une image

JPG

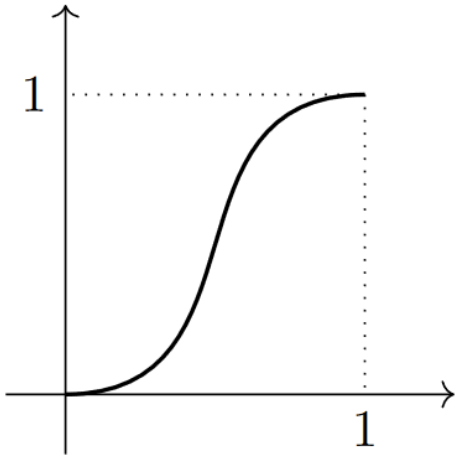
PNG



Recherches personnelles

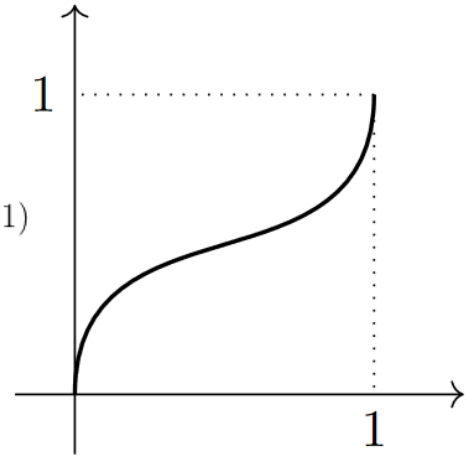
Création d'une fonction de modification du contraste:

$$\frac{1}{2} + \frac{1}{2}\sin(\pi x - \frac{\pi}{2})$$



Contraste Plus

$$\frac{1}{2} + \frac{1}{\pi}\arcsin(2x - 1)$$



Contraste Moins

Recherches personnelles

Utilisation des données EXIF (Exchangeable image file format), pour les modifier et y insérer des messages:

Données classiques:

Appareil photo	
Marque appareil photo	Apple
Modèle d'appareil photo	iPhone 8
Focale	F/1.8
Temps d'exposition	1/70 secondes
Sensibilité ISO	ISO-25
Compensation	0 étape
Distance focale	4 mm
Ouverture maxi	
Mode de contrôle de logiciel	Motif
Distance au sujet	
Mode flash	Pas de flash, obligatoire

Insertion de « TIPE Stéganographie 2021 »:

Appareil photo	
Marque appareil photo	TIPE
Modèle d'appareil photo	Steganographie
Focale	F/2021
Temps d'exposition	1/70 secondes
Sensibilité ISO	ISO-25
Compensation	0 étape
Distance focale	4 mm
Ouverture maxi	
Mode de contrôle de logiciel	Motif
Distance au sujet	
Mode flash	Pas de flash, obligatoire

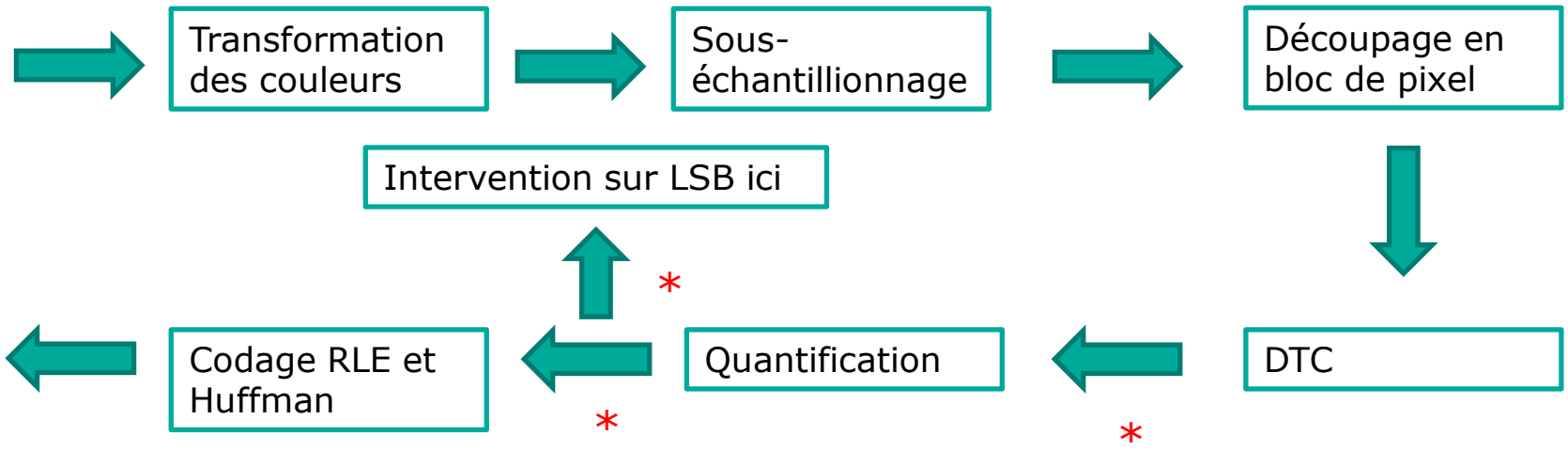
Bibliothèque : exif

06

Méthode JSPEG

Méthode JSPEG

Algorithme de compression JPGE:



Transformation des couleurs: Passage de RVB en YCrCb

Sous-échantillionnage: Suppression de chrominance

Découpage en bloc de pixel: Travailler sur des blocs de 8x8 indépendants, pour diminuer la complexité

DTC: Permet d'évaluer l'amplitude des changements d'un pixel à l'autre afin d'identifier les hautes et basses fréquences.

Quantification: atténuer les hautes fréquences qui ont été mises en évidence par la DTC

Le codage: encodage du bloc de pixel.

07

ANNEXES

Annexe

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image as img


def convertMsg(texte):
    motbin = ''
    for el in texte:
        motbin += '{0:08b}'.format(ord(el)) #donne le numbre binaire avec 8 bits
    return motbin

def implémentent(texte,Image):
    textebin = convertMsg(texte)
    taille = len(textebin)
    l = len(Image)
    c = len(Image[0])
    cpt = 0
    for j in range(0,l):
        for i in range(0,c):
            for p in range(0,3):
                if Image[j,i,p] <= 1: #Eviter les faux positifs à la lecture
                        Image[j,i,p] = 20
                if Image[j,i,p] < 15 and cpt < taille: #or Image[j,i,0] >= 240)
                        Image[j,i,p] = int(textebin[cpt])
                        cpt += 1
    if cpt < taille:
        return "tous le texte n'est pas dans l'image"
    return Image

def lecture(Image):
    listebin = []
    l = len(Image)
    c = len(Image[0])
    for j in range(0,l):
        for i in range(0,c):
            for p in range(0,3):
                if Image[j,i,p] <= 1:
                    listebin.append(Image[j,i,p])
    listebin = list(map(str,listebin))
    n = len(listebin)
    motbin = [ "".join(listebin[i:i+8]) for i in range(0,n,8)]
    mot = ""
    for el in motbin:
        mot += chr(int(el, 2))
    return mot

""" Image """

Image1=img.open("D:\\doc\\TIPE 2020\\Code\\Documents\\rose.jpg")

T = np.array(Image1)
texte = "oui"
Image_T = implémentent(texte,T)


resultat = img.fromarray(Image_T)

resultat.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\rose&texte(noir et blanc).png")
```

Insertion d'un message binaire dans une image en noir et blanc

Annexe

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image as img

def convertMsg(texte):
    motb15 = []
    motbin = ''
    for el in texte:
        motbin = '{0:08b}'.format(ord(el)) #donne le numbre binaire avec 8 bits
        motb15.append(int(motbin[0:4], 2))
        motb15.append(int(motbin[4:8], 2))
    return motb15

def implémentent(texte,Image):
    texteb15 = convertMsg(texte)
    taille = len(texteb15)
    l = len(Image)
    c = len(Image[0])
    cpt = 0
    for j in range(0,l):
        for i in range(0,c):
            for p in range(0,3): #pour chaque pixel
                if Image[j,i,p] <= 20 and cpt < taille:
                        Image[j,i,p] = int(texteb15[cpt])
                        print(j,i,p,int(texteb15[cpt]))
                        cpt += 1
                if Image[j,i,p] <= 20 and cpt >= taille:#Eviter les faux positifs à la lecture
                        Image[j,i,p] = 21
    if cpt != taille:
        return "tous le texte n'est pas dans l'image"
    return Image

def lecture(Image):
    listeb15 = []
    mot = ''
    l = len(Image)
    c = len(Image[0])
    for j in range(0,l):
        for i in range(0,c):
            for p in range(0,3):
                if Image[j,i,p] <= 15:
                    listeb15.append(Image[j,i,p])
    n = len(listeb15)
    listel = ['{0:04b}'.format(listeb15[i])+'{0:04b}'.format(listeb15[i+1]) for i in range(0,n-1,2)]
    for el in listel:
        mot += chr(int(el, 2))
    return mot



"""INSERTION"""

""" Ouverture """
Image1=img.open("D:\\doc\\TIPE 2020\\Code\\Documents\\champi.jpg")
Image2=img.open("D:\\doc\\TIPE 2020\\Code\\Documents\\champi.jpg")
f = open("D:\\doc\\TIPE 2020\\Code\\Documents\\a.txt","r")
doc = f.read()
texte = doc

""" Insertion """
Tab1=np.array(Image1)

res = implémentent("bonjour ceci est un test",Tab1)
resultat = img.fromarray(res)

resultat.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\champi&texte.png")


""" LECTURE """

""" Ouverture """
Image_avec_T=img.open("D:\\doc\\TIPE 2020\\Code\\.spyproject\\champi&texte.png")

Tab_T = np.array(Image_avec_T)
mot = lecture(Tab_T)
print(mot)
```

Insertion d'un message en demi octet

Annexe

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image as img

def insertion(Tab1,Tab2): #On rentre l'image 2 dans la 1
    l = len(Tab1)
    c = len(Tab1[0])
    for i in range(l):
        for j in range(c):
            for p in range(3):
                couleur1,couleur2 = '{0:08b}'.format(Tab1[i,j,p]),'{0:08b}'.format(Tab2[i,j,p])
#convertion en binaire de la couleur
                Tab1[i,j,p] = int(couleur1[0:4] + couleur2[0:4],2) #récupération bits de poid fort et
faible, concaténation + conversion base 10
    return Tab1

def desinsertion(Tab):
    l,c = len(Tab),len(Tab[0])
    Tab_poid_fort = np.ndarray((l,c,3),np.uint8)
    Tab_poid_faible = np.ndarray((l,c,3),np.uint8)
    for i in range(l):
        for j in range(c):
            for p in range(3):
                couleur_bin = '{0:08b}'.format(Tab[i,j,p])
                couleur1 = int(couleur_bin[0:4] + '0000',2)
                couleur2 = int(couleur_bin[4:8] + '0000',2)
                Tab_poid_fort[i,j,p] = couleur1
                Tab_poid_faible[i,j,p] = couleur2
    return Tab_poid_fort,Tab_poid_faible

""" Ouverture """
Image1=img.open("D:\\doc\\TIPE 2020\\Code\\Documents\\1920_1.jpg")
Image2=img.open("D:\\doc\\TIPE 2020\\Code\\Documents\\1920_2.png")

""" Insertion """
yop = np.array(Image1)
Tab1=np.array(Image1)
Tab2=np.array(Image2)

res = insertion(Tab1,Tab2)
resultat = img.fromarray(res)

resultat.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\1920_3_jpg.jpg")

""" Desinsertion """
Image=img.open("D:\\doc\\TIPE 2020\\Code\\.spyproject\\1920_3_jpg.jpg")
res = np.array(Image)
Tabfort,Tabfaible = desinsertion(res)
resfort = img.fromarray(Tabfort)
resfaible = img.fromarray(Tabfaible)
resfort.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\1920_3(1)_from_jpg.png")
resfaible.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\1920_3(2)_from_jpg.png")
```

Insertion d'une image dans une autre:

Annexe

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image as img


def brithness(img,value = 30):
    n,m = len(img),len(img[0])
    for i in range(n):
        for j in range(m):
            for k in range(3):
                if img[i,j,k] < 255 - value or img[i,j,k] > value:
                    img[i,j,k] += value
                else:
                    if img[i,j,k] > 255 - value:
                        img[i,j,k] = 255
                    else:
                        img[i,j,k] = 0
    return img

def fondu(img1,img2):
    n1,m1 = len(img1),len(img1[0])
    n2,m2 = len(img2),len(img2[0])
    for i in range(0,n1):
        for j in range(0,m1):
            for k in range(0,3):
                if i < n2 and j < m2:
                    test = img1[i,j,k] + img2[i,j,k]/90
                    if test < 255:
                        img1[i,j,k] = test
                else:
                    img1[i,j,k] = img1[i,j,k]
    return img1


def contrastePlus(image):
    return ((1+np.sin(np.pi*((image/255)-0.5)))/2)*255

def  contrasteMoins(image):
    return 255*(np.pi/2+np.arcsin(((image/255-0.5)*2)))/np.pi

def pitch_color_plus(color,coeff,image):
    n,m = len(image),len(image[0])
    for i in range(n):
        for j in range(m):
            image[i,j,color] = min(255,image[i,j,color]+coeff)
    return image

def pitch_color_moins(color,coeff,image):
    n,m = len(image),len(image[0])
    for i in range(n):
        for j in range(m);:
            image[i,j,color] = max(0,image[i,j,color]-coeff)
    return image

img1 = img.open("D:\\doc\\TIPE 2020\\Code\\Documents\\oiseau.jpg")
img2 = img.open("D:\\doc\\TIPE 2020\\Code\\Documents\\fleure.jpg")
T1 = np.array(img1)
T2 = np.array(img2)

""" FONDU """
res = fondu(T1,T2)
resultat = img.fromarray(res)
resultat.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\fondu_oiseau_fleure.png")

""" LUMINOSITE """
Image2 = brithness(res,40)
resultat = img.fromarray(res)
resultat.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\fondu_oiseau_fleure&lumi.png")

""" CONTRASTE """
Image2 = contrastePlus(Image2)
resultat = img.fromarray(res)
resultat.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\fondu_oiseau_fleure&lumi&contras.png")

""" PITCH BLEU + & ROUGE - """ #exemple
Image2 = pitch_color_plus(2,60,Image2)
Image2 = pitch_color_moins(1,40,Image2)
resultat = img.fromarray(res)
resultat.save("D:\\doc\\TIPE 2020\\Code\\.spyproject\\fondu_oiseau_fleure&lumi&contras&pb&mr.png")
```

Insertion d'un motif par fondu

Annexe

```
from exif import Image



with open("D:\\doc\\TIPE 2020\\Code\\Documents\\exif.jpg", 'rb') as image_file:
    my_image = Image(image_file)

my_image.model = "Steganographie"
my_image.make = "TIPE"
my_image.f_number = "2021"

with open("D:\\doc\\TIPE 2020\\Code\\Documents\\modified_image.jpg", 'wb') as new_image_file:
    new_image_file.write(my_image.get_file())
```

Insertion dans les données EXIF

Annexe

```python
from math import cos, pi, sqrt
from copy import deepcopy
from typing import List


def encoder_dct(ancien_tableau : List[List[int]]) -> List[List[int]]:

    # Le nouveau tableau aura le même nombre d'entrées (et de sous-
    # entrées) que le tableau d'origine, mais pas les mêmes valeurs,
    # c'est pourquoi on commence à copier le tableau d'origine (et
    # ses tableaux imbriqués) pour en créer un distinct.
    nouveau_tableau : List[List[int]] = deepcopy(ancien_tableau)
    for nouveau_y in range(8):
        for nouveau_x in range(8):
            nouvelle_valeur = 0
            for ancien_y in range(8):
                for ancien_x in range(8):
                    # Le cosinus retourne un facteur qui pondère
                    # la valeur selon si on est dans la strie ou non.

                    nouvelle_valeur += (
                        ancien_tableau[ancien_y][ancien_x] *
                        cos(((2 * ancien_y + 1) * nouveau_y * pi) / 16) *
                        cos(((2 * ancien_x + 1) * nouveau_x * pi) / 16)
                    )
            # Si on est au bord du tableau (aplat complet),
            # cosinus retournera toujours 1 et le nombre
            # pourrait être très gros : on va donc le réduire un peu
            if nouveau_y == 0:
                nouvelle_valeur /= sqrt(2)
            if nouveau_x == 0:
                nouvelle_valeur /= sqrt(2)
            nouvelle_valeur /= 4
            nouveau_tableau[nouveau_y][nouveau_x] = nouvelle_valeur

    return nouveau_tableau

def createQuatification(coeff): #Crée la matrice de quantification
    return [[1+(1+i+j)*coeff for j in range(8)] for i in range(8)]

def quantifCoeff(Tab,coeff): #Crée la matrice quantifiée
    matQuanti = createQuatification(coeff)
    return [[Tab[i][j]/matQuanti[i][j] for j in range(8)] for i in range(8)]

def convertMsg(texte):
    motbin = ''
    for el in texte:
        motbin += '{0:08b}'.format(ord(el)) #donne le numbre binaire avec 8 bits
    return motbin

def insertionMsg(texte,tableau,coeff): #insert le message après la quantification
    tabDCT = encoder_dct(tableau)
    tabQuanti = quantifCoeff(tabDCT,coeff)
    msgBin = convertMsg(texte)
    cpt = 0
    for i in range(8):
        for j in range(8):
            nb = tabQuanti[i][j]
            if abs(nb) > 1:
                nbBin = list('{0:08b}'.format(5))
                nbBin[-1] = msgBin[cpt]
                nbBinLSB = "".join(nbBin)
                tabQuanti[i][j] = int(nbBinLSB,2)
                cpt += 1
    return tabQuanti,cpt
```

Insertion dans DTC après quantification