**CLUSTERING ET STATISTIQUES MULTIDIMENTIONNELLE**

Juan Pablo Aguirre N° Etudiant : 2199654

Juliana Castellanos N° Etudiante : xxxxxxxx

Sandra

Cintia Bru

Université Lyon 2 / ICOM

M1 Informatique 2019/2020

Table des matières

[1. Introduction 2](#_Toc33044542)

[2. Méthodes de binarisation 2](#_Toc33044543)

[2.1 Chargement d’images 2](#_Toc33044544)

[2.2 Fonction de binarisation globale 3](#_Toc33044545)

[2.3 Résultat de la binarisation globale 3](#_Toc33044546)

[2.4 Vote majoritaire 4](#_Toc33044547)

[2.5 Binarisation global vs binarisation global plus vote majoritaire 5](#_Toc33044548)

[2.6 Binarisation local, algorithme de Niblack 6](#_Toc33044549)

[2.7 Binarisation local vs binarisation global 7](#_Toc33044550)

[3. Fusion de QR code avec une image de fond 8](#_Toc33044551)

[3.1 Choisir et charger l’image de fond et le code QR 8](#_Toc33044552)

[3.2 Redimensionnement de l’image du fond 8](#_Toc33044553)

[3.3 Changement d’espace de couleur 9](#_Toc33044554)

[3.4 Assemblage 10](#_Toc33044555)

[3.5 Assemblage à couleur – Bonus 10](#_Toc33044556)

[4. Conclusion 11](#_Toc33044557)

# Introduction

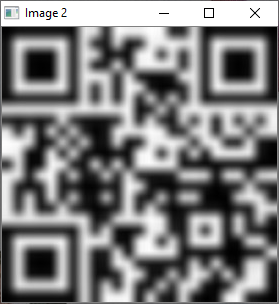
Dans le cadre du contrôle continu du cours infographie, je réalise ce rapport dans lequel j’étudie et compare les méthodes de binarisation local et global dans un code QR.

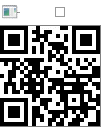
Je présente les algorithmes suivants : Chargement d’une image, binarisation global, vote majoritaire, binarisation local avec la méthode de Niblack, le changement de la taille d’une image et la fusion de deux images.

Pour le déroulement de chaque point, nous trouveront à chaque section le nom, le résultat et des images de l’implémentation.

# Présentation des données

## Contexte

Pour charger les images, une fonction a été créée qui reçoit deux paramètres, le chemin du fichier et un drapeau pour indiquer si vous voulez charger l'image en couleur ou en niveaux de gris. Cette fonction est chargée de valider si le fichier existe ou non. Enfin, la méthode renvoie l'objet « Mat » qui contient les matrices avec les données de l'image. Cette partie de code est créée pour être réutilisée en de multiples endroits.



**Image 1** : QR original

**Image 2** : QR après acquisition

## Les variables quantitatives

Pour binariser une image avec la méthode globale, il faut calculer le seuil en divisant par 2 la différence entre la valeur d'intensité maximale et minimale de la matrice.



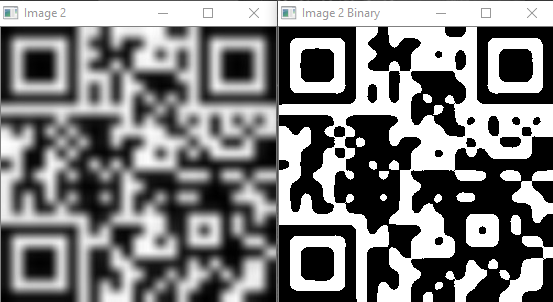
**Formule 1** : Calcule du seuil global

Par la suite, tous les pixels de l'image sont parcourus et leurs valeurs sont affectées par la règle suivante :

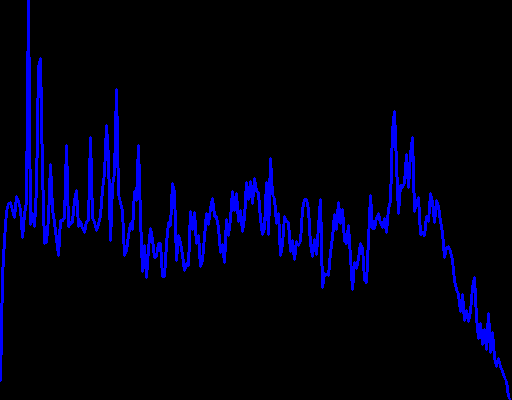
* Si la valeur du pixel est inférieure ou égale au seuil, le pixel devient noir, sinon il devient blanc.

## Les variables qualitatives

Comme nous pouvons le voir, le résultat est évidemment une image en noir et blanc. Mais il est évident que la binarisation globale ne fonctionne pas très bien pour les images qui ont des pixels avec des valeurs relativement homogènes, car la valeur est globale et les différentes parties isolées de l'image ne sont pas prises en compte par contre, c’est la même valeur du seuil qui est pris en compte.



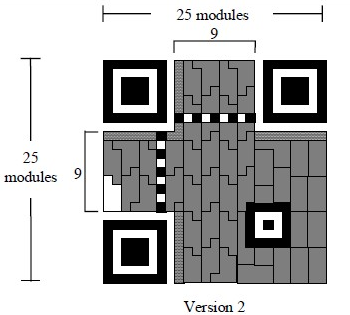
**Image 3** : Image floutée à gauche, image binarisé de façon global à droite



**Image 4** : Histogramme de l’image floutée

## Vote majoritaire

Sachant que les codes QR version 2 sont générés par une matrice de 25 x 25 modules dans lesquels chaque module est un carré de « K x K » pixels qui peut représenter la couleur noire ou blanche, nous pouvons calculer la taille d'un module à partir d'une image carrée en divisant le nombre de lignes ou de colonnes par 25.

  
 **Image 5** : Structure d’un code QR version 2.

A partir de la taille d'un module, nous parcourons l'image module par module aussi bien dans « X » que dans « Y » tout en comptant le nombre de pixels noirs et blancs à l'intérieur de chaque module, finalement les valeurs des pixels de chaque module sont affectées selon la majorité, c'est-à-dire que si il y a plus de pixels noirs, tout le module devient noir, s'il y a plus de pixels blancs, tout le module devient blanc.



**Image 6** : Résultat binarisation global et vote majoritaire

## Binarisation global vs binarisation global plus vote majoritaire

Après avoir appliqué les deux méthodes, on peut observer une grande différence dans les résultats où la binarisation global plus le vote majoritaire est beaucoup plus efficace que la simple mise en œuvre d'une binarisation global isolé.



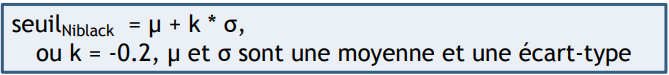
**Image 7** : Image original à gauche, binarisation global au centre et vote majoritaire à droite

Pour mettre en évidence, en utilisant le vote majoritaire nous constatons que en termes générales que l’image est bien reconstruite, mais nous évincions que des petites détails sont perdus, surtout les modules isolés.

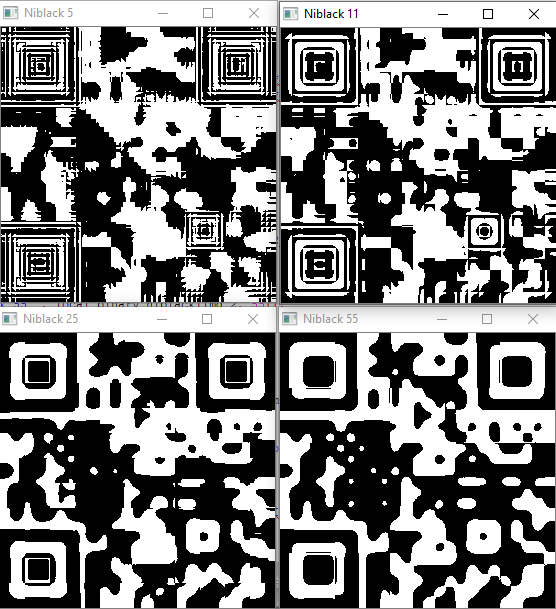
## Binarisation local, algorithme de Niblack

Cette fois-ci, il est proposé d'effectuer une binarisation locale en utilisant l'algorithme de Niblack qui consiste à parcourir l'image à l'aide d'une fenêtre glissante de taille « P x P », dans laquelle nous calculerons pour chaque fenêtre un seuil et binarisée localement . La taille de la fenêtre glissante doit absolument être un numéro multiple du nombre total de colonnes ou de lignes de l'image à parcourir.

Le seuil de Niblack est calculée en additionnant la moyenne d'intensité des pixels de la fenêtre avec la multiplication de l'écart type par « -0.2 ».



**Formule 2** : Seuil de Niblack

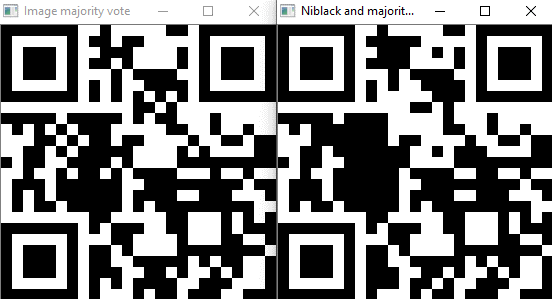


**Image 8** : Binarisation local avec différentes valeurs de fenêtre glissant : 5, 11, 25 et 55 respectivement

Nous pouvons mettre en évidence qu’au plus haut valeur de taille pour la fenêtre glissant, mieux c’est le résultat de la binarisation.

## Binarisation local vs binarisation global

Après avoir mis en œuvre des binarisations globales et locales et puis le vote majoritaire, nous pouvons constater que la binarisation globale fonctionne mieux parce qu’elle conserve plus de détails. Le problème avec la binarisation locale se trouve dans les zones où une couleur est beaucoup plus dominante que l’autre soit blanche ou noire, car le seuil locale sera fortement influencée.



**Image 9** : Binarisation local et global après le vote majoritaire respectivement

# Fusion de QR code avec une image de fond

Dans cette partie, nous avons l’objectif de fusionner deux images dont la première c’est un code QR et la deuxième une image de fond tout en permettant que le code reste lisible

## Choisir et charger l’image de fond et le code QR

Nous chargeons l’image de fond avec la même méthode mentionnée dans la section 2.1 mais cette fois-ci, nous passons « true » comme paramètre « is\_color » parce que le but c’est la charger avec toutes les matrices de l’espace de couleur.

## Redimensionnement de l’image du fond

Après avoir chargé l'image de fond, nous devons s’assurer qu'elle a exactement la même taille que le code QR. Pour ce faire, une fonction a été écrite qui reçoit l'image et la taille souhaitée en type de données Size(x, y).



**Image 10** : Image du fond de taille original et redimensionnée

## Changement d’espace de couleur

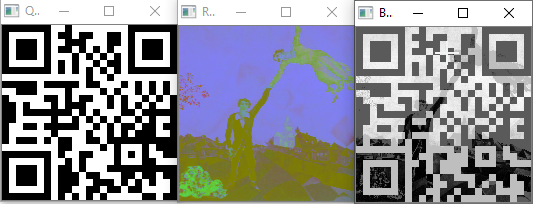
D’après l’article « Visually significant qr codes: image blending and statistical analysis » pour effectuer la fusion des images, il faut changer d’espace de couleur l’image du fond pour obtenir les valeurs de luminance, pour cela nous utiliserons l’espace YCB



**Image 10** : Image du fond dans l’espace YCB

## Assemblage

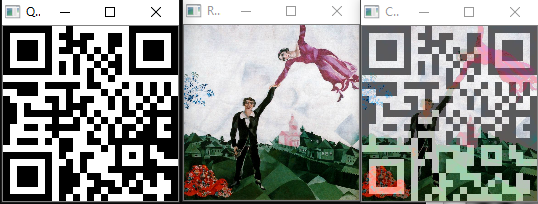
Pour l’assemblage, nous parcourrons de code QR, lorsque les pixels soient blancs nous affectons l’image assemblé par la valeur plus grande entre la valeur de luminosité du pixel dans la même position de l’image du fond et le seuil supérieur, dans le cas contraire où il y ait de pixels noirs, nous affectons l’image de sortie par la valeur plus petite entre la valeur de luminosité du pixel dans la même position de l’image du fond et le seul inferieur.



**Image 11** : Résultat de l’assemblage des images

## Assemblage à couleur – Bonus

Pour l’assemblage du code QR avec un fond à couleur, nous devons prendre la sortie de la méthode d’assemblage en niveaux de gris qui va représenter le canal Y de l’image assemblé à couleur, puis nous faisons la fusion avec la méthode « merge » et finalement, nous changeons l’espace de couleur de YCB à RGB.



**Image 12** : Résultat de l’assemblage des images à couleur

# Conclusion

Dans le cadre de ce travail pratique, des algorithmes de binarisations ont été codés et mis en œuvre, chacun avec ses différentes résultats qui ne sont pas du tout satisfaisants.

Pour améliorer les résultats des algorithmes de binarisation, nous pouvons utiliser différents techniques post-binarisation en fonction du cas particulière, dans le cas des codes QR le vote de majorité marche assez bien parc que l’image originelle est composée de modules carrés.

De plus, ce travail m’a servi pour explorer le langage de programmation C++ et la librairie openCV, dans laquel nous avons fait preuve d'une grande facilité lors de la manipulation d’images représentées par de matrices.