**Projet Image – 25/01/2018**

**TPB08**

**Florian Cousin – Lysa Gramoli**

Table des matières

[I. Introduction du projet et description de la base de données 2](#_Toc504666864)

[II. Méthodologie finale 3](#_Toc504666865)

[A. Phase de prétraitement et de segmentation 3](#_Toc504666866)

[B. Phase de traitement et de classification 4](#_Toc504666867)

[III. Résultats expérimentaux 4](#_Toc504666868)

[IV. Fonctions Matlab 8](#_Toc504666869)

[A. Fonction de segmentation et de prétraitement 8](#_Toc504666870)

[B. Fonction de classement 8](#_Toc504666871)

[V. Bibliographie 9](#_Toc504666872)

# Introduction du projet et description de la base de données

Le but de ce projet image est de détecter des pièces de monnaie dans une image. Pour cela, nous possédons une base de données d’une cinquantaine d’images représentant une ou plusieurs pièces.

Pour obtenir cette base de données, nous avons pris une série de photos à l’aide d’un téléphone portable posé sur une pile de livres afin de minimiser les effets de flou dus aux éventuels micro-tremblements de la main. La distance entre les pièces de monnaie et la caméra est ainsi conservée. Les pièces ont été posées sur une feuille blanche afin d’uniformiser le fond. Ces photos ont ensuite été rognées et renommées.

Les pièces ont été prises 4 fois individuellement (2 recto et 2 verso). Il y a ensuite 9 photos avec deux pièces, 13 photos avec 3 pièces, 4 photos avec 4 pièces, 1 photo avec 5 pièces, 1 photo avec 6 pièces et 1 photo avec 8 pièces (tous les types de pièces sont inclues dans cette photo).

Après cela, nous avons mesuré le diamètre en pixels des pièces ainsi que la moyenne des couleurs pour chaque type de pièce (monnaie Européenne). Nous avons alors établi le facteur d’échelle dont la formule est donnée ci-dessous :

Nous avons alors en moyenne **f = 0.0558 mm/px** sur toute la base de données.

Nous avons ensuite calculé les moyennes RVB pour chaque couleur de pièces (Argent, Cuivre, Or), nous obtenons le tableau ci-dessous :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Moyenne Cuivre | | | Moyenne Or | | | Moyenne Or 2 | | | Moyenne Argent | | | Moyenne fond | | |
| R | V | B | R | V | B | R | V | B | R | V | B | R | V | B |
| 188,97 | 131,17 | 72,92 | 168,33 | 129,33 | 46,67 | 165,25 | 131,25 | 69,25 | 137,13 | 113,88 | 79,88 | 207,25 | 188 | 161,75 |

# Méthodologie finale

Pour traiter les images nous avons réalisé plusieurs étapes :

* Phase de prétraitement et de segmentation
* Phase de traitement et de classification

## Phase de prétraitement et de segmentation

La phase de prétraitement et de segmentation a pour but de segmenter et binariser l’image (blanc pour les pièces et noir pour le fond) afin de détecter facilement les pièces. Plusieurs méthodes ont été testées (Ligne de partage des eaux, Reconstruction, Top-Hat…).

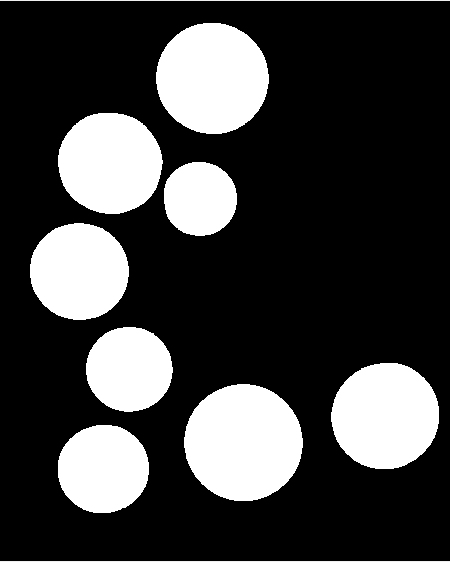
Celle retenue est la reconstruction géodésique en partant du négatif de l’image sur laquelle on a retiré le fond (en faisant une moyenne des pixels de fond).

Pour la segmentation, nous avons utilisé une segmentation multi-seuil (fonction multithresh (I, 2)) qui permet de segmenter automatiquement les images en plusieurs intervalles (3 dans notre cas).

Nous utilisons une fonction de Matlab (imfill) pour combler les éventuels trous dans les pièces.

Pour le prétraitement, nous avons choisi d’utiliser la reconstruction géodésique par dilatation reposant sur l’utilisation de masques et de marqueurs. L’avantage de cette méthode est qu’elle permet de conserver la forme des objets et d’enlever les éléments qui sont à l’extérieur des objets choisis. Nous effectuons ensuite une ouverture et nous effaçons les objets en bordure.

Phase de prétraitement et de segmentation



## Phase de traitement et de classification

La phase de traitement et de classification permet de faire la détection des pièces à partir de l’image binarisée. Pour cela, on utilise une fonction de Matlab (imfindcircles) qui permet de créer des cercles rouges autour des objets et de récupérer leur rayon et les coordonnées du centre.

Il faut ensuite comparer ces rayons aux rayons théoriques des pièces qui sont donnés en pixels grâce au facteur d’échelle. (Ce facteur sera donné par l’utilisateur)

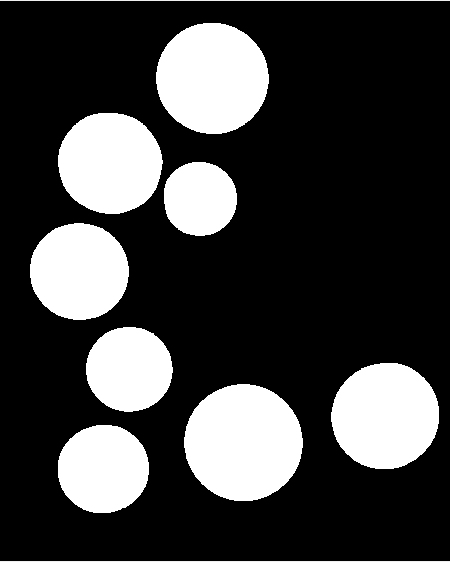
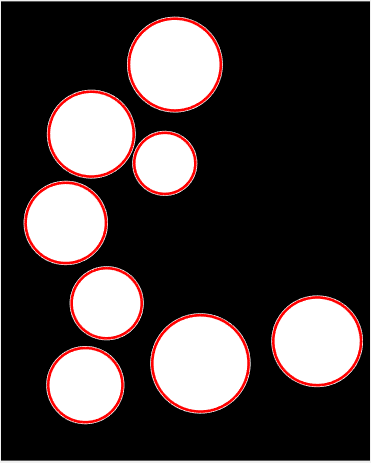
Les moyennes des couleurs seront aussi des paramètres importants à prendre en compte. Pour cela, nous calculons la distance entre les moyenne RVB qui constituent la couleur du cuivre théorique et les moyennes RVB trouvées dans la pièce choisie. Nous faisons de même pour les moyennes RVB avec l’or, l’argent et l’or 2 s’il y en a. Nous récupérons ensuite la distance minimale entre les différentes couleurs afin de savoir si la pièce est cuivrée, dorée ou autre.

En rassemblant les conditions de taille et de couleur, nous pouvons classer les différentes pièces de l’image. Ainsi, nous obtenons une classification des pièces selon leur taille et leur couleur.

# Résultats expérimentaux

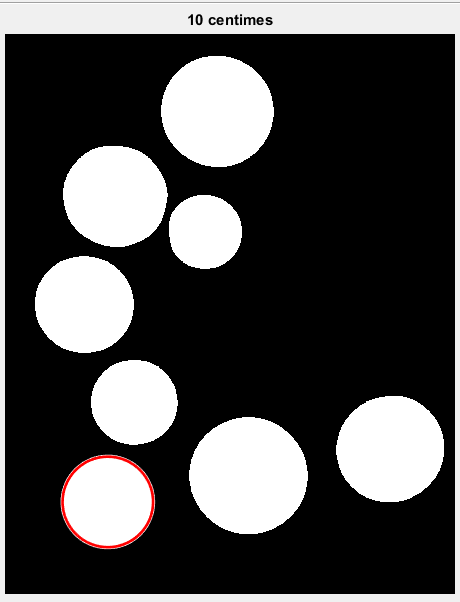
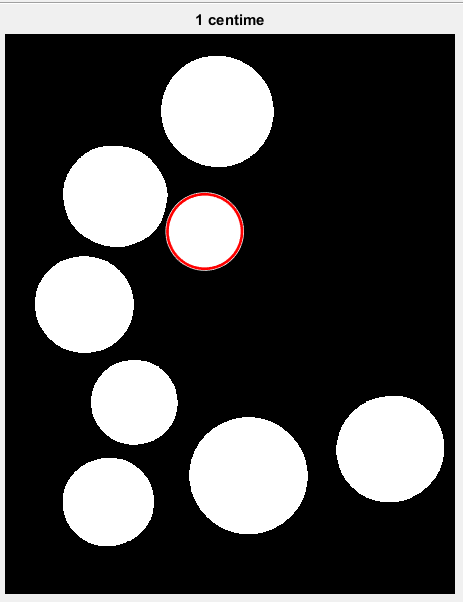
Les résultats expérimentaux montrent que la méthode fonctionne pour certains cas mais malheureusement pas pour tous. En effet, à cause de certains reflets existant sur certaines pièces, la moyenne des couleurs RVB est erronée. Ainsi, la couleur peut être identifiée comme cuivrée alors qu’elle est dorée et vice-versa.

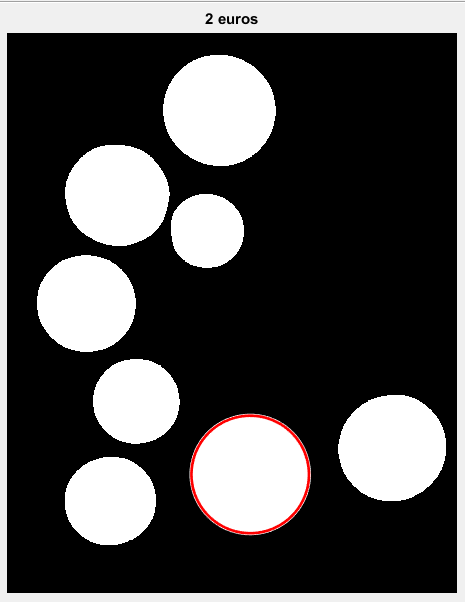
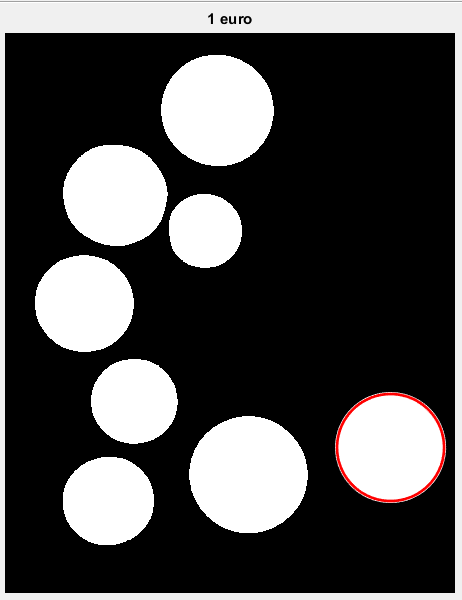
Selon la dataset, le programme est plus ou moins efficace (à cause des ombres plus ou moins marquées par exemple).

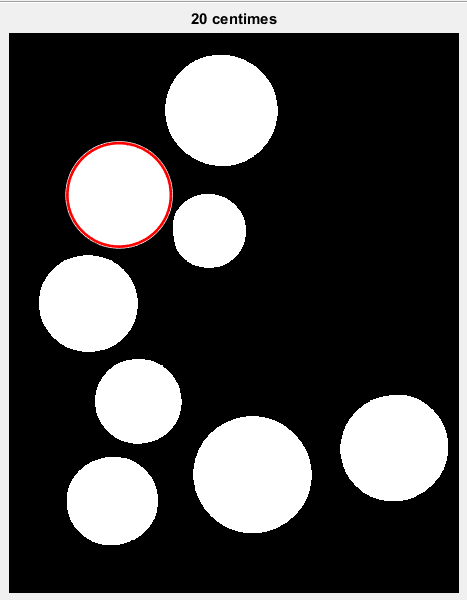
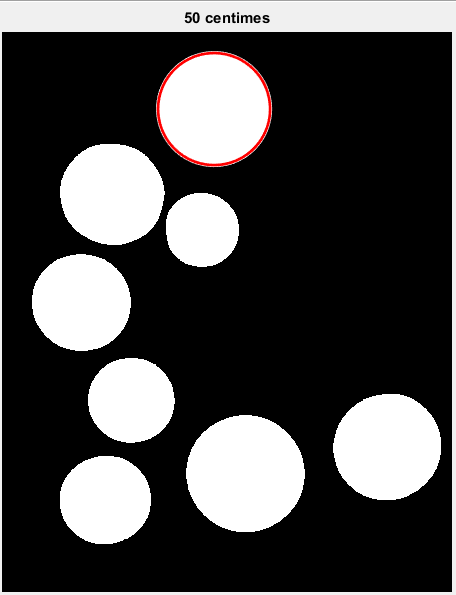


Détection des contours des pièces, récupération des rayons des pièces.

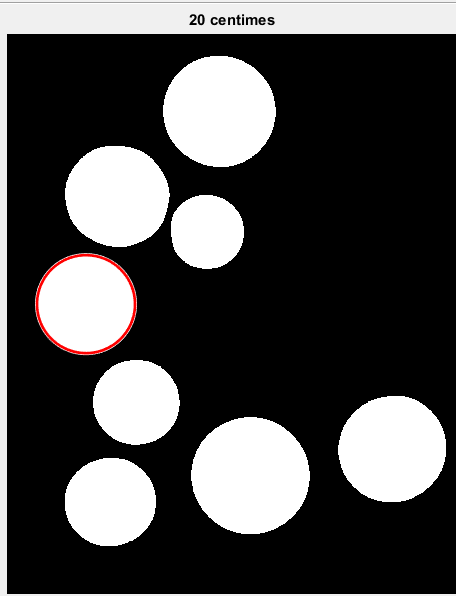
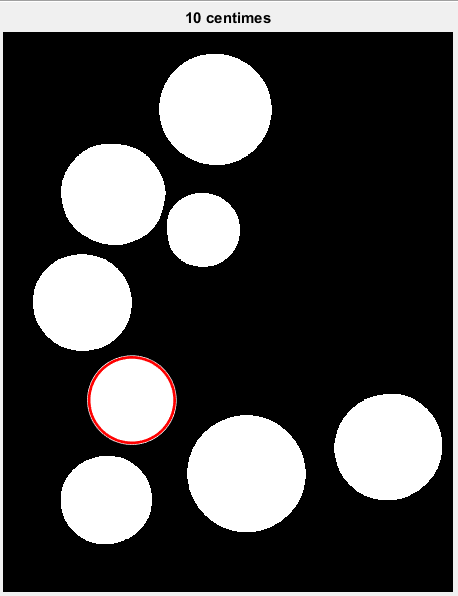
Classification des pièces :







Celles qui ne fonctionnent pas sont les pièces de 5 centimes et de 2 centimes qui sont considérées comme des 10 centimes ou des 20 centimes :

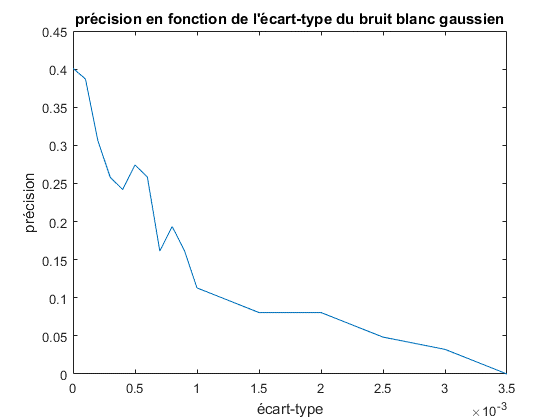


La précision de la méthode est définie comme étant le nombre d’images correctement détectées divisé par le nombre d’image en tout.

Ainsi, la précision de notre méthode de détection sur notre dataset est 40.10%.

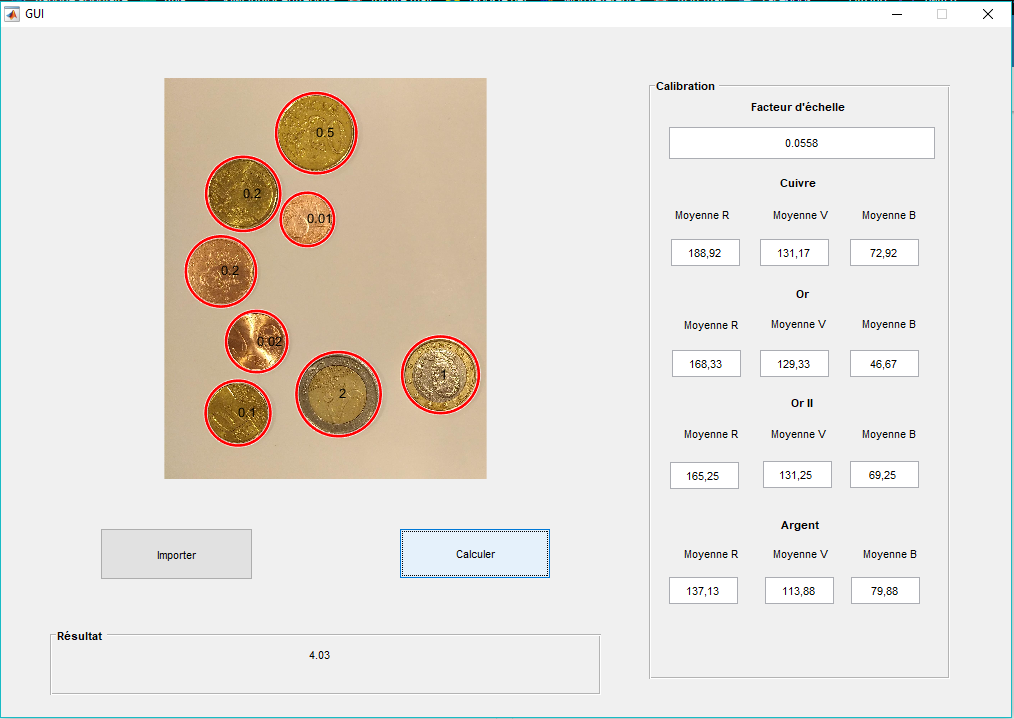
De plus, on peut mesurer la robustesse au bruit en ajoutant sur les images un bruit blanc gaussien d’écart-type variable et en mesurant la précision sur notre dataset. On peut ajouter un bruit blanc gaussien d’écart-type sigma et de moyenne 0 sur l’image I grâce à la fonction imnoise(I, ‘gaussian’, 0, sigma).

On peut ainsi dessiner un graphique représentant la précision de notre méthode sur notre dataset modifié d’un bruit d’écart-type variable.



On remarque que la précision est décroissante et son allure est plutôt exponentielle. De plus, la précision tombe à 0 à partir d’un bruit blanc gaussien d’écart-type environ 0.0035.

Finalement, on obtient sur le GUI le résultat suivant :



La pièce de 5 centimes a été confondue avec la pièce de 20 centimes, mais les autres pièces ont pu être distinguées.

# Fonctions Matlab

## Fonction de segmentation et de prétraitement

function If = negatif(I)

% Cette fonction permet de binariser et d'extraire les pièces d'une image.

% On passe d'abord l'image en négatif.

% Im est une image rgb représentant des pièces

% If est une image binaire qui correspond à la segmentation des pièces

## Fonction de classement

function compte = ciecle(mat, origin, scale, moyenne)

% Cette fonction permet de distinguer les différentes pièces d'une image en

% comparant sa taille et sa couleur par rapport aux moyennes théoriques

% calculées auparavant.

% compte : Nombre comptant le montant total des pièces d'une image (en

% euros)

%

% mat : Image prétraitée (binarisée, les pièces sont en blanc et le fond en

% noir) où l'on veut distinguer les pièces.

%

% origin : L'image couleur, celle qui correspond à mat.

%

% scale : scale factor correspondant à l'échelle de l'image en mm/pixels

function pieces = couleur(I, tab)

% sert à mesurer la moyenne des couleurs des pièces

% I : image de pièces

% tab : tableau des centres et rayons de pièces

% pieces : tableau des centres, rayons et couleurs moyennes des pièces

function [moy, sigma] = moyenne\_couleur(I, xcenter, ycenter, radii)

% Calcule la moyenne dans un cercle de l'image

% Le cercle y est forcément entièrement

% I : Image couleur

% xcenter : abcsisse du centre du cercle

% ycenter : ordonnée du centre du cercle

% radii : rayon du cercle

% moy : moyenne des pixels de I dans le cercle

% sigma : écart-type

function d3D = distance\_3D (x,y,z,x2,y2,z2)

% Fonction permettant de calculer la distance entre les moyennes théoriques

% des couleurs et celles trouvées dans l'image.

% x = moyenne couleur R théorique

% y = moyenne couleur V théorique

% z = moyenne couleur B théorique

% x2 = moyenne couleur R de la pièce

% y2 = moyenne couleur V de la pièce

% z2 = moyenne couleur B de la pièce

# Bibliographie

<https://fr.mathworks.com/help/images/ref/imfindcircles.html>

<https://fr.mathworks.com/help/images/ref/bwconvhull.html>

<https://fr.mathworks.com/help/images/examples/detect-and-measure-circular-objects-in-an-image.html>

<https://fr.mathworks.com/help/images/ref/viscircles.html>

<https://fr.mathworks.com/help/images/ref/multithresh.html>

<https://www.mathworks.com/help/matlab/creating_plots/callback-definition.html>

<https://fr.mathworks.com/help/matlab/ref/getfield.html>

<https://fr.mathworks.com/help/matlab/ref/dir.html>

<https://fr.mathworks.com/help/matlab/creating_plots/add-text-to-specific-points-on-graph.html>