Traitement de Signal 3T

Louis Delbrouck Shayann Gonzalez
Medhi Godefroid Thomas Jacob

1 Introduction

Il s'agit du rapport de notre groupe sur notre projet Cue-Locks sobrement intitulé É-plaqué. S'agissant du projet présenté comme Contrôle Dimensionnel, l'objectif est de vérifier et de comparer différents aspects concernant des plaques de Cue-Locks.

2 Cahier de charge

2.1 Objectif

On commence par établir une plaque comme référence et par la suite les autres plaques y seront comparées.

2.2 MVP attendu

Un outil terminé et capable de déterminer la compatibilité d'une plaque passée après la référence tout en conservant l'appareil à la même position.

2.3 Limites et contraintes pour l'utilisateur

Le Projet ayant une durée limitée, il ne nous était pas possible d'implémenter un outils parfaitement abouti et 100% efficient. Il est donc bien évident que le projet comporte des limites fonctionnelles que nous allons détailler ici :

- La position de la caméra doit obligatoirement être fixe sinon le *ratio pixel/cm* est changé et nos données n'ont plus de sens si deux images comparées ont des *ratios* différents.
- Il est important de conserver une solution qui reste *User-friendly*
- L'utilisateur souhaite une solution qui ne soit pas basée sur des paramètres de conformités mais sur une plaque passée en référence. Les critères de conformités doivent donc être susceptibles de changer d'une image de plaque à l'autre

Pour ce qui est du choix de librairie Python, le client n'ayant rien imposé, nous avons fait nos propres choix que nous allons maintenant expliquer.

3 État de l'art

Les solutions existantes pour le contrôle dimensionnel utilisent fréquemment des bibliothèques telles qu'*OpenCV* et des *réseaux de neurones convolutifs*[1]. Les systèmes de vision industrielle intègrent des caméras spécialisées pour l'inspection automatisée, tandis que des solutions commerciales englobent diverses industries avec des capteurs avancés. En comparaison, notre solution se distingue en se concentrant sur la comparaison de plaques à la chaîne incluant des critères tels que : les dimensions, les couleurs et la détection de défauts, ce qui peut la rendre particulièrement adaptée à des applications spécifiques où un contrôle dimensionnel strict s'impose.

4 Aspect Technique

Nous avons choisi assez tôt ce que nous allions prendre en compte pour l'analyse et la comparaison des plaques.

- Nous avons commencé par essayer de déterminer les contours d'une plaque et puis du reste des plaques (voir lien vers annexes screen contours).
- Ensuite, une fois ce début de traitement d'image fait, nous avons réparti les différents filtres :
  - Thomas prendra le filtre défaut qui deviendra le code *holes*.
  - Mehdi prendra le filtre couleur qui deviendra le code *plate_color*.
  - Shayann prendra le filtre dimension qui deviendra le code *plate_size*.
  - Louis prendra le filtre orientation qui deviendra le code *plate_orientation*.
- Lors d'une première présentation orale avec nos professeurs où nous leur avons fait part d'une petite démonstration de l'état de l'application (qui a bien sûr évolué depuis), nous avons recueilli des remarques visant à corriger ou améliorer notre solution.
  - ✔ Inverser la binarisation de l'image afin d'avoir les défauts en blanc + labelisation pour les trous (voir dans le cours théorique, annexe aussi).
  - ✔ Dilatation et érosion.
  - ✔ Augmenter le contraste pour améliorer la qualité.
  - ✘ Songer à boucher les petits trous pour les plaques brunes afin de déterminer si la plaque a des défauts ou pas. [2]

1. Les neurones convolutifs sont des unités spécialisées dans les réseaux de neurones artificiels qui traitent des motifs locaux dans une image, facilitant ainsi la détection de caractéristiques complexes.

2. Nous avons pas tenu compte de ce point car beaucoup de problèmes avec les plaques brunes. Ce n'était donc pas une priorité et nous avons abouti à un résultat concluant sans ça.

- ✓ Plutôt que de faire un filtre gaussien pour avoir une image plus nette, on doit pouvoir augmenter le contraste.
- ✓ Finaliser la plate_size.
- ✓ Finaliser la plate_orientation.
- ✓ Finaliser le plate_color.
- ✓ Finaliser la plate_holes.
- ✓ Finaliser la GUI.
- ✓ La GUI doit effectuer les différents tests et afficher les résultats.

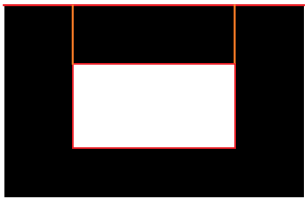
- Une fois arrivé à la fin du projet et suite à des discussions avec nos professeurs, nous avons admis que nous aurions pu réaliser une fonction de pré-traitement de l'image qui aurait été commune aux fichiers plutôt que de faire des choses sensiblement similaires, mais presque à chaque fois uniques.

4.1 Analyse de la logique générale

L'ensemble de notre application a été réalisée en python à l'aide de librairies comme OpenCV [3]. (Open Source Computer Vision Library) mais aussi numpy [4]

Voici comment la plupart des codes fonctionnent :

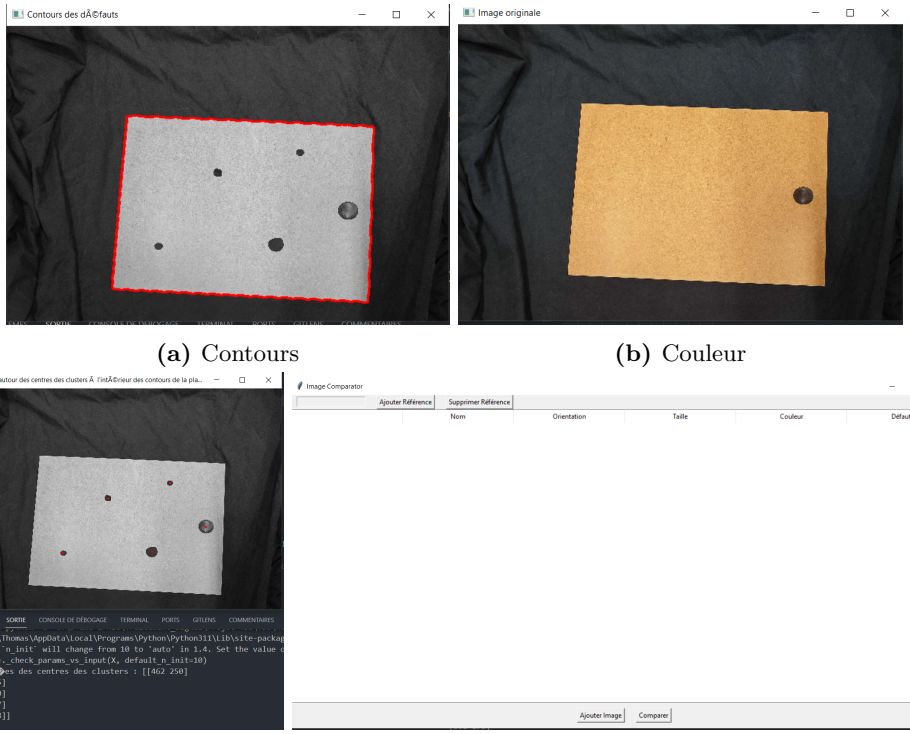
① On **charge une image** placée en paramètre .
② On applique maintenant un **pré-traitement** (varie un peu d'un fichier à l'autre : augmentation de contraste, flou Gaussien).
③ On change l'image en **nuance de gris**.
④ On applique un seuil afin de **binariser l'image**.
⑤ En général, on **détermine les contours** de la plaque [5]. (*N'ayant pas la donnée px/cm, nous travaillons avec des tailles relatives qui servent surtout à des fins de comparaisons entre les plaques.*)
⑥ On applique ensuite la logique propre au code (mesure d'un Δ longueur dont arctan donne un angle pour orientation par exemple).



$$\text{angle} = \frac{\text{Distance entre le point supérieur et le sommet gauche du contour}}{\text{Distance entre le point supérieur droit et le sommet droit}}$$

3. Lien vers la documentation officielle d'OpenCV : `https://docs.opencv.org/4.x/d7/da8/tutorial_table_of_content_imgproc.html`
4. Lien vers la documentation officielle de Numpy : `https://numpy.org/doc/1.26/user/index.html#user`
5. Plusieurs algorithmes essayés (Canny, Sobel, gradiants de couleurs à la section 6.1 dans le fichier `annexes.tex`)

⑦ **Afficher l'image traitée** (si c'est demandé dans l'appel à la fonction)



(a) Contours **(b)** Couleur

(c) Défauts **(d)** visuel du prototype de l'application

Figure 1 – Affichage Post-Traitement

⑧ **Version Finale**, ses avantages et limites.

Notre application servant jute de consultance, nous avons jugé qu'il n'était pas nécessaire de stocker les données mesurées avec de la persistance [6] . Au contraire, pour ne pas nous encombrer de données inutiles une fois que l'on sait si la plaque est conforme ou non, on écrase le *json* à chaque nouveau chargement d'une référence.

Nous sommes toujours limité par l'angle et la position de la caméra, car ce n'était pas un critère à prendre en compte ici pour la réalisation de ce projet. De plus, n'ayant pas la valeur exacte *px/cm* (nous sommes cependant sûrs que cette valeur est la même pour toutes nos photos), il est donc difficile d'estimer avec précision la taille de la plaque et nous devons donc nous contenter de comparer sur base d'estimations.

Notre solution, É-plaqué, comporte tout de même des avantages notables comme une super gui conçue pour être *User-friendly*, vous permettant de pouvoir charger le dossier d'images intégralement avec la garantie que chaque image ne sera chargée qu'une seule fois. La référence sera toujours bien mise en évidence en haut d'un tableau de type *tk.treeview* [7](module de la librairie graphique ~~mal foutue~~ bien documentée en ligne).

6. Fait de conserver ses données pour la postérité dans une base de données par exemple.
7. Module de Tkinter servant à mettre en place des tableaux type *excel*, lien vers la documentation treeview

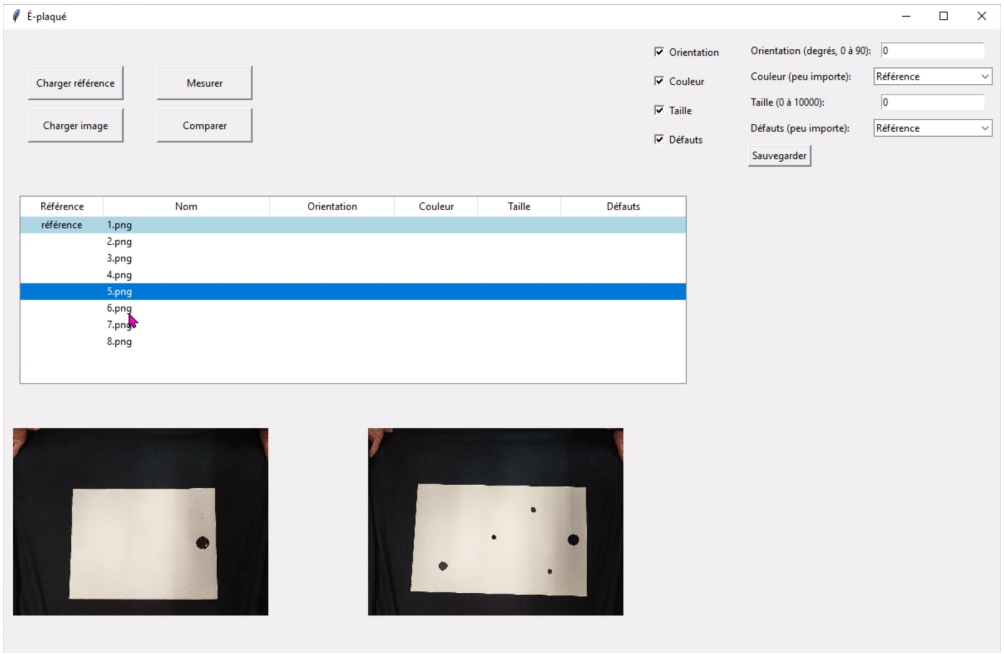


Figure 2 – Visuel Final de l'application

Mais aussi et surtout la possibilité d'effectuer les tests indépendamment les uns des autres ainsi que d'appliquer une tolérance sur les différents champs comparés. Enfin l'affichage tel que montré dans le point 7 1a est bien évidemment disponible depuis l'application. Nous estimions qu'une version en ligne de commande ne serait pas adaptée à une utilisation active de l'application.

5 Conclusion et version Mk II

Nous sommes arrivés au terme de ce projet assez rapidement. Les filtres n'étant pas tous compliqués à mettre en place, c'est surtout la mise en place de la *gui* qui nous a pris du temps, car la documentation est immense et généralement très axée concept théorique plus que mise en application dans un cadre pratique (bien qu'il existe quand même des vidéos *Youtube* qui traitent le cas).

5.1 Pour des éventuelles améliorations

On pourrait envisager d'améliorer le support sur lequel on place les plaques par un support gradué sur lequel via un système d'OCR [8].

8. Optical Character Recognition, selon wikipedia : La reconnaissance optique de caractères (ROC, ou OCR pour l'anglais optical character recognition), ou océrisation, désigne les procédés informatiques pour la traduction d'images de textes imprimés ou dactylographiés en fichiers de texte.

5.2 Conclusion Thomas

Dans l'ensemble le projet ma beaucoup plus car il était assez libre, ne comprenait pas un workflow agile que je trouve trop contraignant à respecter et à suivre dans la durée. Ce que j'ai beaucoup apprécié ça a été (au début) le processus de mise en place du pré traitement puis du traitement en tant que tel des images passée en paramètres, ainsi que la conception de la gui vu qu'on était libre de lui donner la forme qu'on voulait.Si je devais continuer le projet je réfléchirais à trouver un moyen de déterminer le ratio px/cm

5.3 Conclusion Mehdi

Dans l'ensemble, je ne suis pas fan de ce genre de projets en informatique même si j'ai appris des choses. Ce n'est pas ma branche préférée. Cependant, le groupe a bien travaillé sur le projet et a pu être terminé dans les temps. Aussi, l'organisation et la pénibilité du projet ont été amoindries étant donné la liberté donnée dans le cadre de ce projet, ce qui est appréciable.

5.4 Conclusion Louis

Comme on le dit, toutes les bonnes choses ont une fin. C'est pourquoi je rédige cette brève conclusion personnelle pour partager mes impressions sur notre projet de traitement de signaux. Dans l'ensemble, j'ai grandement apprécié cette expérience, en particulier parce que nous avons exploré un domaine nouveau pour nous, à savoir le traitement d'images et les divers filtres associés. De plus, la cohésion au sein du groupe a été très chouette, ce qui a renforcé notre motivation à concrétiser le projet et à travailler de manière organisée, d'autant plus que nous collaborons déjà sur plusieurs autres projets.

5.5 Conclusion Shayann

J'ai globalement bien apprécié participer à ce projet de groupe. C'est la première fois que je travaillais sur du traitement d'image et j'ai trouvé ça très intéressant et amusant à faire. Cela nous permet d'un peu mieux comprendre le fonctionnement des applications ou logiciels de traitement. J'ai bien aimé le fait de devoir réfléchir à des manière de filtrer les images en fonction de ce que nous voulions. Le groupe que nous avions a aussi aidé à l'appréciation du projet car nous nous entendons tous bien et nous sommes déjà ensemble dans d'autres groupe de projet. Grâce à cela, nous sommes déjà habitué à travailler ensemble et nous pouvons continuer à utiliser notre méthode de travail qui fonctionne pour les autres cours.

6 Annexes

6.1 Choix des algorithmes de détection de contours

Pour la réalisation du code de *holes.py*, code réalisé pour la détection et le référencement de défauts dans une plaque. J'ai essayé de déterminer les contours de la plaque de plusieurs façons.

1. Détection de défaut sur l'image de la plaques
2. **Problème** : Détection de défauts hors de la plaque, il faut donc limiter la recherche de défaut à l'intérieur de la plaque
3. Recherche d'un algorithme de détection de contours dans le cours théorique
4. Détection de contours par **gradient de couleur** [9].
5. **Problème** : D'une photo à l'autres voire même d'un défauts à l'autre sur la même plaque les valeurs de couleur changent et il est donc compliqué de définir quelle est la couleur du défaut. **Solution Abandonnée**.
6. Détection de contours par **l'algorithme de Sobel** [10].
7. **Problème** : Peu importe le seuil défini, je constate un effet de neige [11]sur l'image **Solution Abandonnée**.
8. Finalement je me tourne vers **l'agorithme de Canny** afin de délimiter les contours de la plaque

9. La détection de contours par gradient de couleur est une technique qui identifie les contours d'objets dans une image en analysant les variations locales de couleur, exploitant les changements de teinte pour détecter les transitions entre régions.

10. La détection de contours par l'algorithme de Sobel est une méthode qui utilise des opérateurs de convolution pour accentuer les variations d'intensité dans une image, permettant ainsi l'identification des contours par l'analyse des gradients de luminosité.

11. Une image bruitée est une image contenant des variations aléatoires de luminosité ou de couleur, souvent indésirables, introduites par des facteurs tels que la sensibilité du capteur, générant un aspect granuleux ou irrégulier.

```
1          def detect_defauts(image_path, screen=False):
2              image = cv2.imread(image_path,
    cv2.IMREAD_GRAYSCALE)
3              image = cv2.resize(image, (600, 400))
4              image_blur = cv2.GaussianBlur(image, (5, 5), 0)
5              edges = cv2.Canny(image_blur, 50, 150)
6              contours, _ = cv2.findContours(edges,
    cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
7              min_contour_size = 20
8
9              if contours:
10                 filtered_contours = [cnt for cnt in contours
    if cv2.contourArea(cnt) > min_contour_size and 50 <
    cnt[0][0][0] < 550 and 50 < cnt[0][0][1] < 350]
11                 plaque_mask = np.zeros_like(image,
    dtype=np.uint8)
12                 cv2.drawContours(plaque_mask,
    filtered_contours, -1, 255, thickness=cv2.FILLED)
13                 points_noirs = []
14
15                 for y in range(image.shape[0]):
16                     for x in range(image.shape[1]):
17                         if
    cv2.pointPolygonTest(np.concatenate(filtered_contours),
    (x, y), False) > 0:
18                             if image[y, x] < 100:
19                                 points_noirs.append((x, y))
20
21                 points_noirs_array = np.array(points_noirs)
22                 num_clusters = 5
23
```

Listing 1 – Utilisation de l'algorithme de Canny pour détercter les défauts à l'intérieurs des contours de la plaque