Rapport de travail

**NoteCounterCHF** - Reconnaissance de billet Suisse

**Traitement d’image**

# Introduction

Nous utilisons presque tous les jours des billets de banque afin de faire nos courses ou d’acheter divers choses au long de la journée. Certaines personnes à la vue atténuée n’ont pas autant d’aisance pour la lecture de la valeur des billets. Cela peut être dû à une mauvaise vision des couleurs (daltonisme, …), à une vision trouble (peut venir de divers maladies ou déformations de partie de l’œil) ou encore à la cécité.

## Contexte et problématique

Si une personne est mal voyante, Il reste le toucher qui permet de distinguer les valeurs des billets grâce à leur conception. En effet, les anciens billets suisses (8ème série) déjà disposaient d’un système de forme pour distinguer les valeurs. Le billet de 10 CHF, par exemple est reconnu par un cercle avec un point au milieu.



Les anciens billets (8ème série) commencent à disparaître aujourd’hui pour laisser place à la 9ème série des billets suisses. Les billets de 10, 20, 50 sont déjà presque tous remplacés par des nouveaux modèles mais la transition est encore en cours. Néanmoins, vu la tendance des banques à déprécier les anciens billets et à leur disparition certaine, nous avons décidé de négliger cette version pour privilégier les nouveaux.

Plus d’information sur la transition à l’adresse : <https://www.snb.ch/fr/iabout/cash/history/id/cash_history_serie9>

Sur les billets récents, les formes ont disparu pour laisser place à une écriture de la valeur du billet en relief.



Si la personne mal voyante ne dispose pas de capacité sensorielle adéquate ou n’a pas d’autres moyens d’identifier la valeur de ses billets, elle est coincée. Sa dernière option est de recourir à une aide d’une personne externe pouvant l’aider lors du payement ou tout simplement à faire ses achats.

## Buts

Le programme NoteCounterCHF a donc pour but de fournir une alternative autonome pour les personnes mal voyantes en détectant la valeur des billets affiché sur une image ou une vidéo.

Le projet est essentiellement un projet de recherche des différentes méthodes d’analyse d’image qui permettrait un développement plus poussé. Les essais des technologies se sont graduellement

## Environnement

Ce projet est développé dans le cadre du cours de Traitement d’image avec M.Tièche lors du 2ème semestre de la 3ème année au sein de la HE-Arc.

Le développement se fait en Python (version 3.6) à l’aide de la librairie de traitement et d’analyse d’image OpenCV (version 3.4.1).

# Recherche

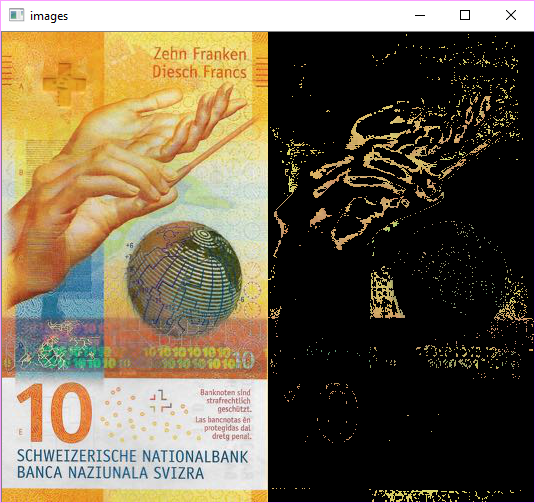
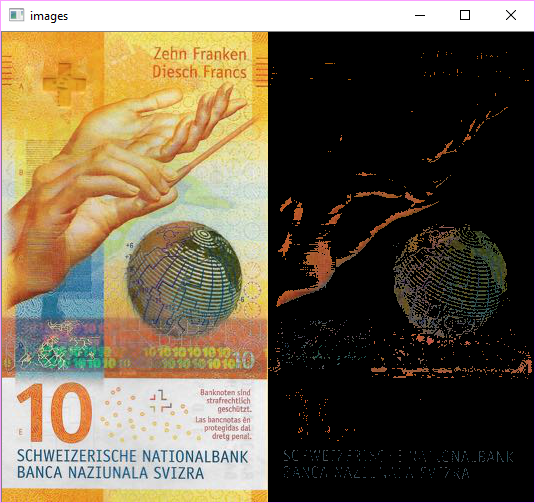
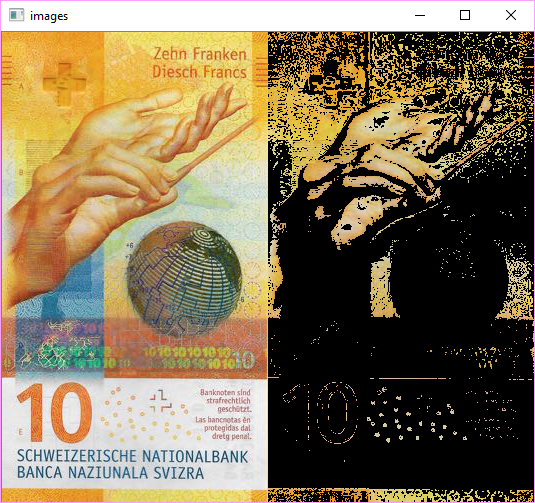
La recherche de méthodologie s’est voulue progressive. Nous avons d’abord analysé une image simple d’un billet afin de réussir à distinguer sa valeur par un moyen quelconque.

## Reconnaissance par couleur

La 1ère approche et la plus simple se base sur cet exemple :

<https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/>

L’idée était de trier des images de billet simple en fonction du taux de couleur présent dans un certain champ défini en RVB. Par exemple avec le billet de 10 CHF pour chaque composante on obtient :



Si on compte le nombre de pixels relatifs on obtient les résultats suivants :

Simple color detection :

Non black point count for Yellow : 27232

Non black point count for Red : 9541

Non black point count for Green : 7644

On voit bien que la couleur de billet qui correspond le mieux est la couleur jaune. C’est donc un billet de 10 CHF ce qui est bien le cas à l’œil.

## Reconnaissance de contour / Classement

L’algorithme de reconnaissance pour couleur n’étant valable que pour une image de billet, on prévoyait de définir une zone en trouvant des formes approximatives de billet dans l’image, ce qui nous aurait permis de détourer, et d’utiliser la reconnaissance par correspondance de couleur à cette zone pour compter chaque billet sur l’image.

Malheureusement (ou heureusement), en commençant la recherche à ce sujet, une solution plus appropriée s’offrait à nous, l’utilisation du pattern matching.

## Pattern Matching

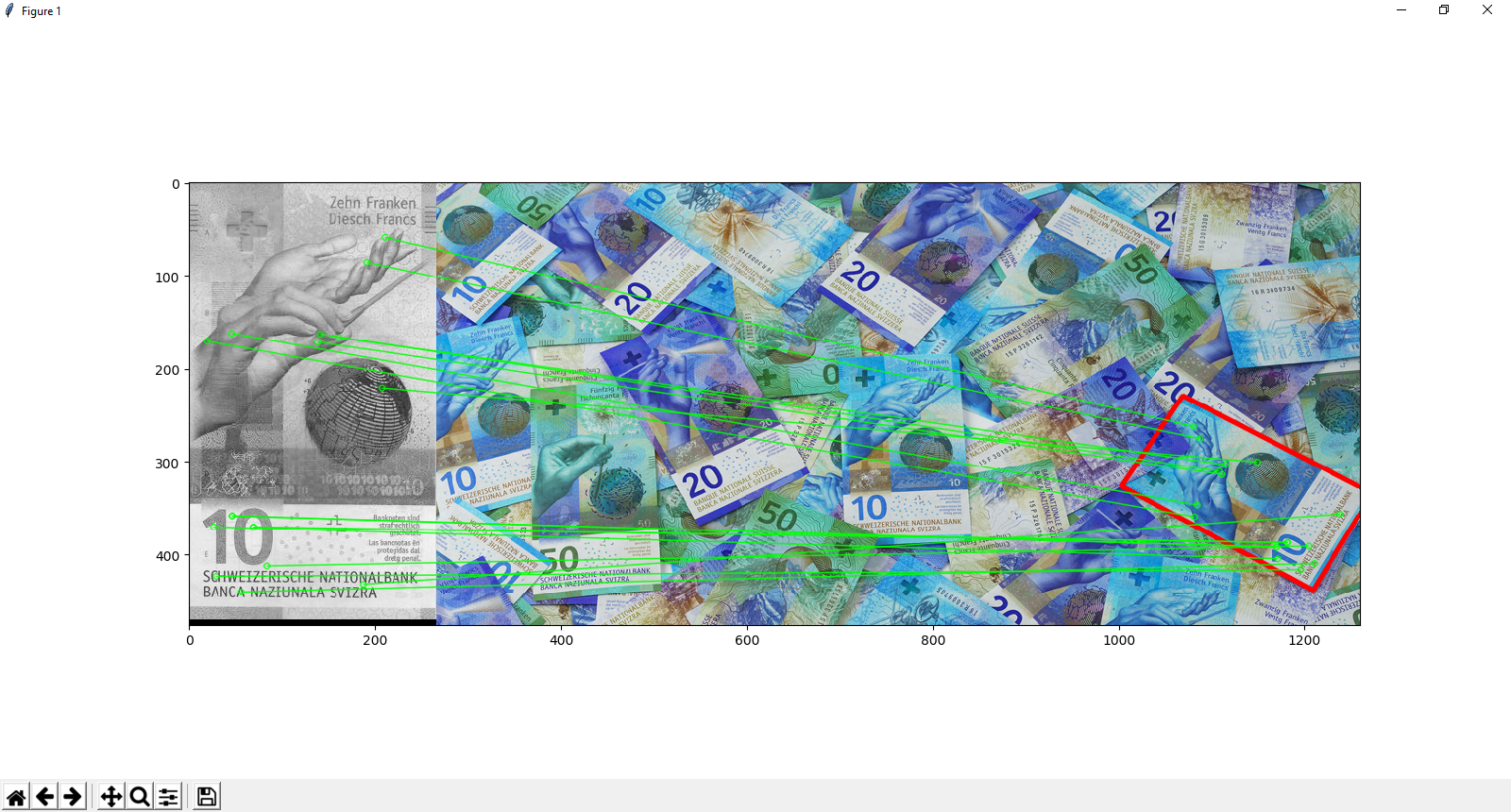
L’idée du pattern matching était bien sûr toujours de détourer une image de billet afin d’appliquer une détection de couleur. Malheureusement cet algorithme est relativement lent.

Nous avons commencé par effectuer une reconnaissance d’un billet dans une position convenable (soit très proche de la taille et de l’angle des modèles) ce qui fonctionnait bien. Avec des variations d’angle et de taille du billet à retrouver dans l’image, l’algorithme ne fonctionnait plus et il fallait faire varier l’angle pour chaque degré et pour chaque taille afin de reconnaitre le billet dans l’image (sachant qu’il faut appliquer une granularité relative à un temps de réponse acceptable d’un point de vue utilisateur).

Entre la boucle de tout les modèles de billets (6 pour le moment mais 14 prévus), la boucle de variation de taille et finalement celle de l’angle, nous avons décidé de changer de méthode pour quelque chose de plus performant : l’homographie.

## Homographie

Avec l’homographie et les points de ressemblances, on obtient des zones approximativement très correctes.



On voit dans cette 1ère image que la forme, bien que plus petite et d’orientation différente, a très bien été détouré.

L’algorithme s’occupe de trouver des points d’intérêts dans toute l’image ce qui a pour effet de trouver la portion d’image possédant le plus de ressemblance avec un modèle mais ainsi on ne peut trouver qu’un seul élément correspondant. Les points de correspondance sont affichés en mode graphique mais uniquement énuméré en mode normal. L’indication utilisateur est la suivante :

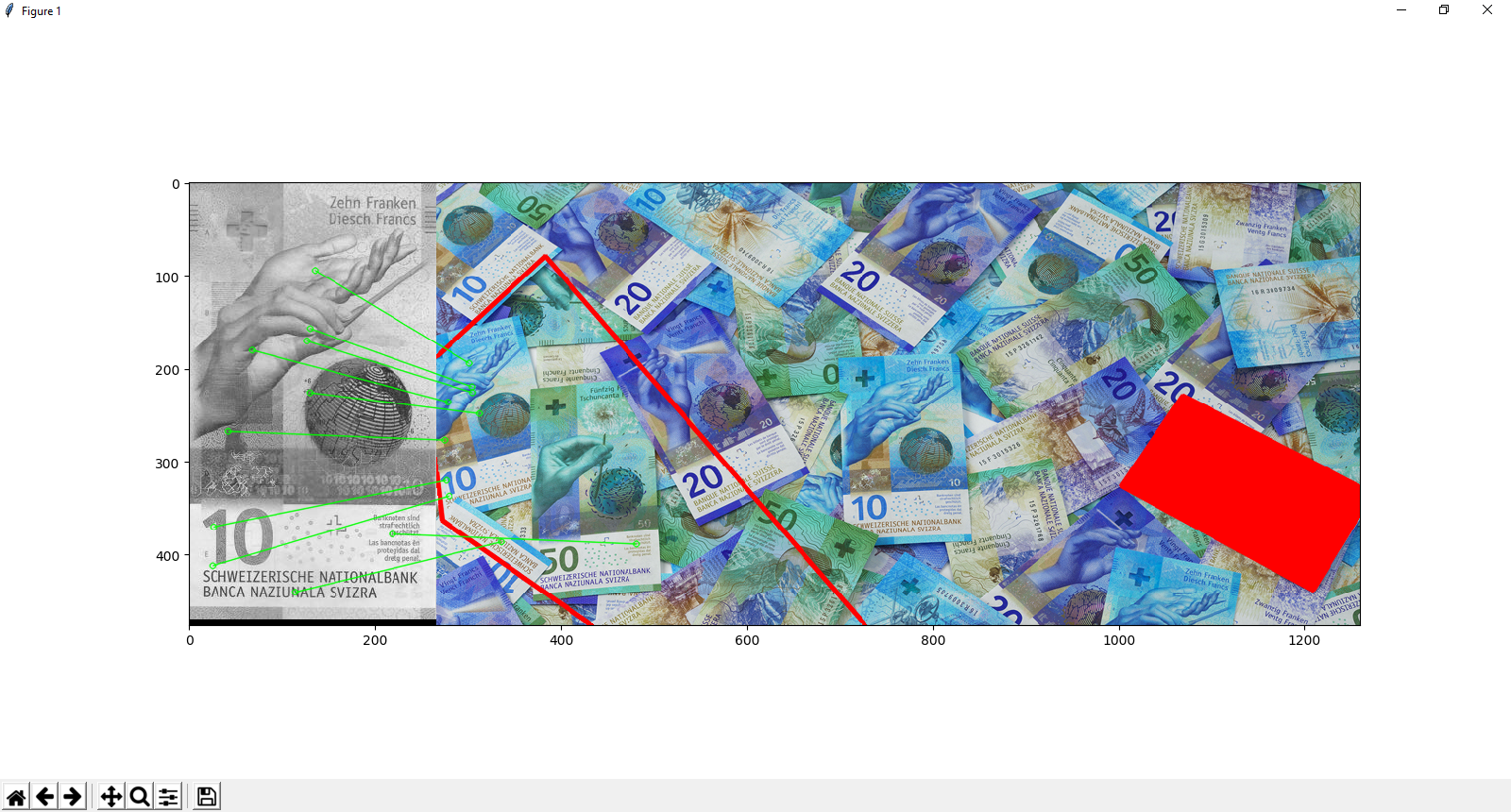
Founded match - 45/38 similarities

->Founded a 10 note.

On voit ici que l’indice de similarité est de minimum 38 points d’intérêts.

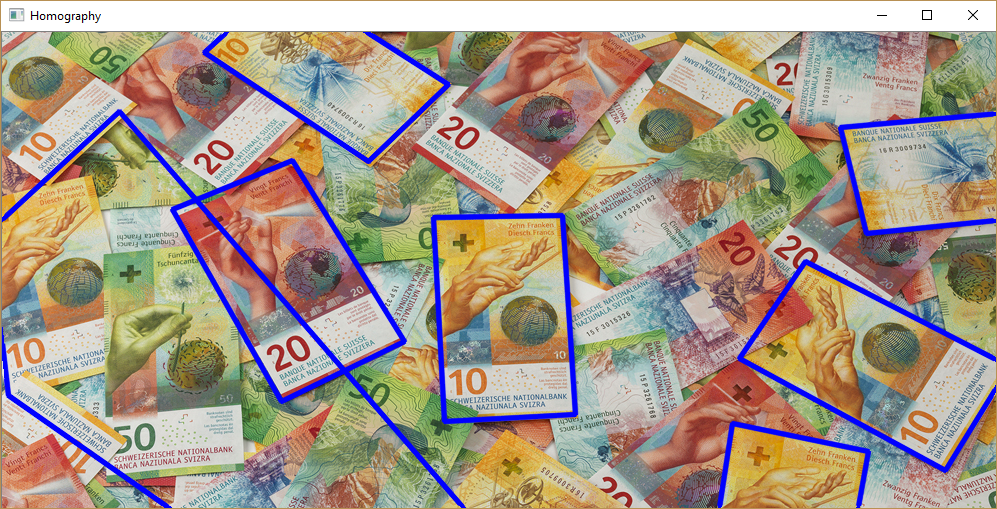
Pour pouvoir en trouver plusieurs il a fallu masquer la zone dans l’image servant au processus et extraire les contours dans une autre image à destination de l’utilisateur. L’affichage ci-dessus étant disponible uniquement en mode debug.

La capture du mode debug ci-dessous montre bien le déroulement de la reconnaissance de plusieurs billets :



Pour distinguer plusieurs billets (donc des modèles différents) on réinitialise les zones rouges (masquage) de l’image pouvant être masquée (la bleue ci-dessus) et on change simplement le modèle utilisé.

Finalement le rendu utilisateur se présente comme suit :



# Conclusion

Toutes les méthodes n’ont pas donné de résultat mais l’homographie a donné des résultats satisfaisants mais pourrait encore être optimisé. On remarque dans les 2 dernières image (homographie) qu’une forme ne correspond pas avec les autres et n’est pas approprié ni en taille et ni en forme.

Pour l’améliorer, on pourrait placer d’autres forme de contrôle (forme rectangulaire, taille par rapport aux autres billets trouvés) mais il ne faut pas que cela empiète sur la possible variation dû à des perspectives ou a des torsions du billet. On pourrait également mixer plusieurs méthodes comme ajouter une reconnaissance par couleur pour confirmer la valeur du billet détouré.

En outre, l’utilisation des différentes méthodes a permis de comprendre l’utilisation et la technique la plus approprié pour la reconnaissance d’image par un modèle précis.

## Sources

Le lien suivant propose la méthodologie suivante : <http://answers.opencv.org/question/118850/best-method-to-recognize-currency-notes/>

* find contours, check for rectangles in appropriate size
* crop and align to xy axes
* use machine learning on cropped images to classify

Currency euro recognition: <https://github.com/carlosmccosta/Currency-Recognition>

Good tutoriel in how to separate by color with OpenCV: <https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/>

Histogramme to collect color range: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html>

Homography: <https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html>

<https://docs.opencv.org/2.4/doc/tutorials/features2d/feature_homography/feature_homography.html>