Rapport
Travaux Pratiques 2

Ecrit par

Théodore Lambolez
Maximilien Petit

28 novembre 2016

Table des matières

1 Introduction

Ce deuxième travail pratique de Traitement Numérique de l'image fait suite à un travail d'initiation au logiciel Optimas. L'objectif de ce deuxième, est de réaliser dans un premier temps un contrôle qualité automatiques de connecteurs. Puis, dans un second temps de réaliser une identification de dimension de clé. Sur ce même logiciel, il faudra donc cette fois-ci explorer quelques pistes rendues possibles par sa fonctionnalité de création de "macros". On prendra soin tout au long de ce rapport, d'expliquer les choix réalisés ainsi que leurs limites.

2 Contrôle de la qualité des connecteurs

Rappel de l'énoncé du problème traité : Nous sommes dans le cas où un industriel, par exemple, nous demande de réaliser un contrôle automatisé de la qualité des composants qui sortent à la fin de sa chaîne de production à l'aide de captures réalisées par un capteur spécifique. Pour ce faire, il nous a fourni un échantillon de quelques captures de connecteurs parmi lesquelles on trouvera un témoin (ie. une capture de connecteur en bon état) ainsi que des captures de connecteurs présentant divers problèmes (broche pliée ou coupée, positionnement décentré et orientation diverse du connecteur).

Stratégie adoptée : On se propose dans un premier temps d'étudier le problème du cas basique où le connecteur serait toujours placé au même endroit sur l'image avec un même type de capteur qui serait situé à la même distance et dans les mêmes conditions environnementales (luminositée notamment).

La stratégie adoptée dans ces conditions est alors de binariser l'image en choisissant un seuil qui permettrait de ne mettre en valeur que les broches (fond en noir et objet en blanc). Ensuite, nous chercherons à caractériser un cadre (ou R.O.I.) où chaque ligne de ce cadre passerait par les broches. De cette manière, en comptant pour un nombre suffisamment grand de ligne avec un pas (sur l'axe des y) régulier entre ces lignes, on peut déterminer un pourcentage de ligne passant par le bon nombre de broche (ici donné). Dans notre étude nous nous sommes fixé un seuil pour ce poucentage qui nous permet de décider si l'état du connecteur est bon ou mauvais. Si le pourcentage est supérieur à 95estimer que le connecteur est en bon état. Dans le cas contraire, il serait en mauvais état. Cette valeur, 95définie arbitrairement qui pourrait être suceptible de changer suivant les améliorations qu'on pourrait faire de cette macro naïve.

Travail préalable d'étude de l'échantillon d'image : Puisque nous avons choisi de réaliser une binarisation à partir d'un seuil à déterminer, nous avons réalisé une étude préalable des différentes façons de déterminer ce seuil. Nous avons rapidement conclus qu'il ne serait pas forcément une bonne idée de réaliser un seuil fixe pour chaque image puisque la luminosité est une condition parfois difficile à maîtriser totalement. Nous choissons donc de déterminer un seuil calculé de manière dynamique par optimas. Pour ce faire, Optimas proposes auto-threshold. On peut configurer cet outil pour qu'il réalise son calcule suivant différents critères. Sachant que nous ne voulons mettre en évidence que les broches, après avoir essayé sur les échantillons données les différentes méthodes, nous avons convergé vers le calcul du seuil par minimisation de la variance. Néanmoins, une première limite se présente déjà. En effet, pour la capture prise dans des conditions de luminosité élevée, on remarque que d'autres parties que les broches du connecteur apparaissent en blanc. Cela est dû au fait que la différence sur cette capture entre le

niveau de gris des broches et le niveau de gris du reste est trop faible. La méthode utilisée par l'au-threshold n'est donc pas suffisamment précise ici.

Voici un échantillon des valeurs calculées pour chacune des captures de l'échantillon de base.

Nom image	seuil bas	seuil haut
Connect1.tif	110	255
Connect2.tif	116	255
Connect3.tif	116	255
Connect4.tif	128	255
Connect5.tif	126	255
Connect6.tif	119	255
Connect7.tif	124	255
Connect8.tif	130	255
Connect9.tif	129	255
Connect10.tif	124	255
Connect11.tif	126	255
Connect12.tif	129	255

TABLE 2.1 – Comparaison des seuils de binarisation automatique avec l'option de minimisation de la variance

Analyse des résultats obtenu avec cette première macro : Voici quelques résultats obtenus avec la macro dont vous pourez trouver le script en annexe.

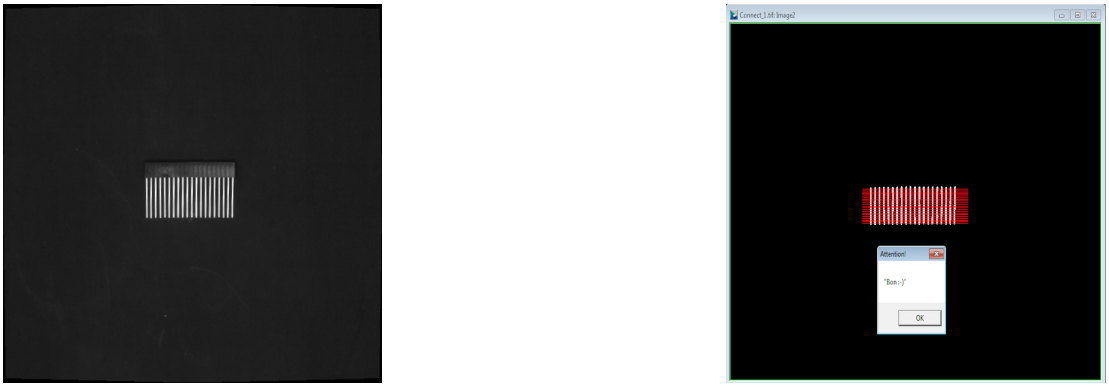

FIGURE 2.1 – Résultat de la macro pour l'image Connect_1.tiff

On constate heureusement que dans le cas de référence, la macro réalise correctement le travail souhaité.

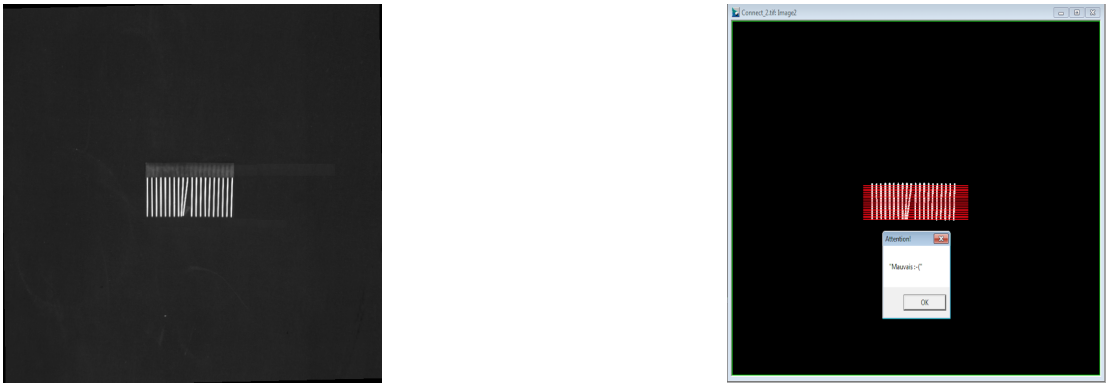

FIGURE 2.2 – Résultat de la macro pour l'image Connect_2.tiff

On remarque dans ce deuxième cas où le connecteur est placé à la même position et présentant une broche pliée que la macro considère que l'état est mauvais. En effet, à partir d'un moment, la broche pliée vient toucher une broche voisine. Puisque notre macro compte en fait les fronts montants des différents profils de lignes. A partir de la hauteur de la capture où la broche vient toucher sa voisine, la macro ne comptera plus qu'un seul front montant.

Précision à propos du calcul du front montant. Nous avons choisi de garder en mémoire dynamiquement la valeur du niveau de gris précédant lors du parcours du profil de ligne. De cette manière on peut comparer le niveau de gris précédant au niveau de gris courant. La condition de détection du front montant utilisé est en fait : si le niveau de gris précédant est 0 (correspondant au noir du fond) et si le niveau de gris courant est strictement supérieur à 0, alors on a un front montant.

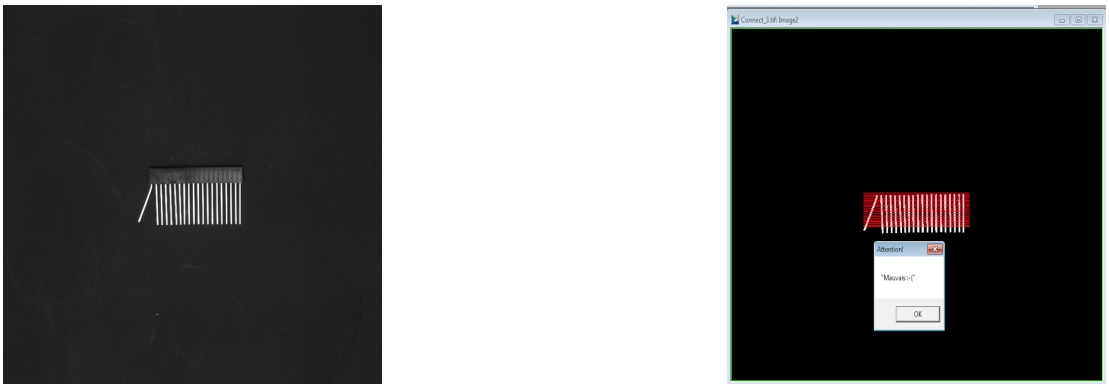

FIGURE 2.3 – Résultat de la macro pour l'image Connect_3.tiff

Nous avons ici enfin un cas intéressant à analyser pour nous donner une première piste d'amélioration de la macro. En effet, nous constatons que la macro aboutis au fait que le connecteur est mauvais. Ce qui est juste ! Néanmoins, on constate également que la première ligne n'est en fait pas prise en compte. Ce qui signifie que la macro donne

la bonne réponse pour une mauvaise raison. Ici le connecteur n'a pas exactement la même place sur la capture que le connecteur sur la capture de référence. Pour éviter ce problème, on peut réaliser une détection de la position du premier pixel appartement à une broche.

Une amélioration basique à réaliser serait de commencer par récupérer le nombre de pixel en largeur de l'image et de réaliser de cette manière des profils de ligne (avec un pas très petit) successifs. On parcours chaque profil de ligne jusqu'à trouver un pixel blanc. On trouvera de cette manière le premier pixel appartenant à une broche du connecteur étudié.

Un deuxième problème mis en évidence par ce test a été caché par le premier problème. En effet, même si le connecteur avait été à la bonne place la macro aurait dit que le connecteur est bon. Il faut réduire la taille de la R.O.I fixée au début.

Une amélioration basique faisant suite à la première serait en fait de définir différents pas de manière empiriques. En effet, une fois qu'on a la position du premier pixel faisant parti d'une broche, on pourrait définir un pas minime en x décroissant. Ensuite, il nous faut caractériser le cadre par trois autres points. Le premier point décalé d'un certains nombre de pixel vers les x croissant donne le deuxième. Le premier point décalé vers les y croissant donne le troisième point. Et finalement le troisième point décalé du même nombre de pixel que le deuxième point vers les x croissant donnera le quatrième point. On a donc ici en suivant cette méthode trois nouveaux pas à définir. En restreignant la R.O.I., la broche pliée sortirait du champs et le problème serait bien détecté !

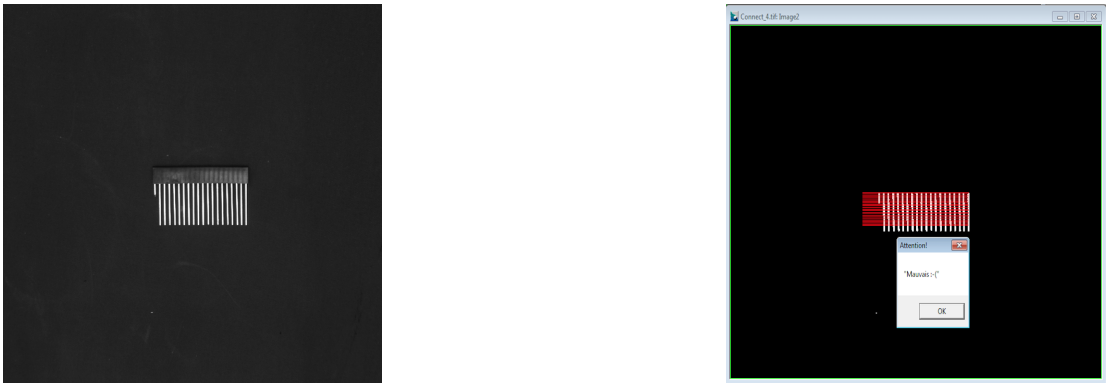

FIGURE 2.4 – Résultat de la macro pour l'image Connect_4.tiff

Ici notre macro donne le bon résultat par chance. En effet, notre R.O.I. est un peu trop décalée par rapport aux broches du connecteur. L'amélioration proposée précédemment permettrait d'éviter de jouer sur cette chance.

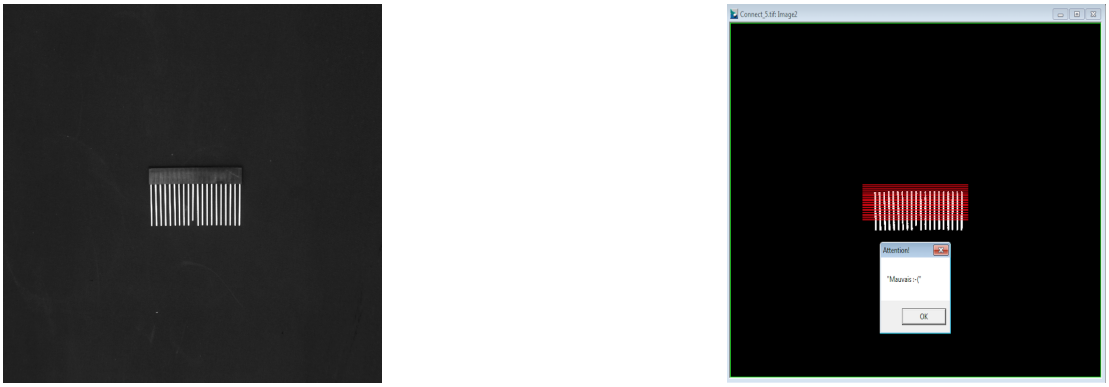

FIGURE 2.5 – Résultat de la macro pour l'image Connect_5.tiff

Nous constatons le même problème que celui rencontré précedemment, la solution à apporter est donc la même.

Autre amélioration possible : Nous avons remarqué qu'une image présente un connecteur orienté dans un autre sens que les autres. Il serait donc important de calculé l'orientation du connecteur. Pour ce faire, nous avons réfléchi à une idée à essayer. Sachant qu'on peut calculer l'orientation d'un bord en utilisant un filtrage de détection de contour comme Sobel (à la place de ne calculer que le module, on pourrait également calculer l'inclinaison du contour et donc avoir l'orientation générale du connecteur). Cependant, cette idée apporterait un autre soucis. En effet, ce genre de traitement amplifie beaucoup certains bruits. Il faudrait donc s'assurer que la capture ne soit pas bruitée avant de réaliser ce traitement avant la binarisation.

3 Identification et mesures de clés

Rappel de l'énoncé du problème traité : Dans cette seconde partie, il faut s'imaginer qu'un industriel nous demande de pouvoir identifier les différentes clés qui arriverait sur un tapis roulant pour les trier par la suite. Pour ce faire, nous avons de même un échantillon de capture de différentes clés. L'objectif qui peut d'identification d'une clé qui semble compliquée peut être ici réduit à la simple mesure de l'écartement de celle-ci. De cette manière, on pourra savoir si c'est une clé de 16 ou une clée de 11 qui défile sur le tapis roulant, par exemple. L'échantillon nous permet de constater qu'on peut travailler dans l'optique de faible variation de luminosité et également travailler avec une faible variation de positionnement de l'objet sur l'image.

Stratégie adoptée : Afin de résoudre le problème qui nous est posé, nous nous proposons de réaliser une seconde macro qui affichera la taille de l'écartement de la clé capturée en millimètre. Cette macro se déroulera en deux temps : la binarisation de l'image de manière automatique comme réalisé précédemment par la première macro de contrôle de qualité, et, la mesure de l'écartement grâce à l'analyse d'un profil de colonne bien choisi. Attention il est important de noter qu'il faut au préalable réaliser un calibrage (à partir d'une image étalon fournie) du logiciel Optimas sans lequel le rapport pixel/mm du logiciel serait éronné. Ce calibrage doit bien entendu correspondre à la même échelle que celle utilisée lors des captures. Une autre contrainte donc évidente de cette macro sera l'utilisation de la même échelle de capture pour chaque prise.

Suite au calibrage d'Optimas, nous réaliserons donc l'étape de binarisation de l'image pour mettre en évidence la clé étudiée (toujours objet en blanc sur fond en noir). De cette manière, nous pouvons beaucoup plus simplement identifier grâce à la valeur de luminance des pixels si un pixel représente une partie de l'objet ou du fond. Pour ce faire, nous réalisons toujours une binarisation avec un seuillage choisi automatiquement par l'outil auto-threshold configuré avec l'option de minimisation de la variance. C'est selon nos tests sur l'échantillon d'images fourni, la méthode la plus viable de calcul du seuil avec le critère de mise en valeur de la clé.

Suite à la binarisation, nous réaliserons donc l'analyse d'un profil de colonne qui passe perpendiculairement à l'axe de symétrie de la "machoire" de la clé. En effet, si le profil de colonne venait à ne pas être perpendiculaire, nous ne prendrions pas la mesure de l'écartement de la clé mais simplement une "corde" possible. Cette valeure ne serait pas très pertinante dans notre intérêt. Nous avons donc choisi arbitrairement deux points de l'image contenant la clé de 16 de telle manière à ce que nos deux critères soient respectés (perpendicularité et traverssement des deux parties représentant la "machoire" de la clé). Ensuite, pour chaque pixel du profil de colonne réalisé, nous avons réalisé un système de conditions afin de pouvoir compter combien de pixels appartiennant au fond réprésentent

l'écartement de la clé. En connaissant au préalable le rapport pixel/mm (ou mm/pixel) après calibrage, nous avons pu en déduire simplement une mesure en millimètre de l'écartement de la clé. En l'occurrence, il suffisait de diviser par deux le nombre de pixel pour avoir une approximation précise au millimètre près de la mesure de l'écartement.

Je rappelle enfin que de la même manière que dans la première partie, nous avons cherché à réaliser une macro simple dans un premier temps afin d'analyser quels autres problèmes se posent, de les analyser et de les résoudre pour optenir un algorithme plus robuste.

Nous allons donc voir dans la prochaine partie les résultats obtenus par la seconde macro (présente en annexes).

Analyse des résultats obtenu avec cette seconde macro : (à finir)

4 Annexes

4.1 Première Macro : Contrôle qualité des connecteurs

```
ClearScreen ();

REAL rPairs[,];
rPairs = GetAutoThreshold(ArROIHistogram,6,2,,ActiveLuminanceRange,);
if ( rPairs && 1 < GetShape(rPairs)[0] )
   {
// sucessful autothreshold

// set threshold
                Threshold(rPairs[1,0]:rPairs[1,1]);

// alternative which works even for color
        GrayToBinary(,rPairs[1,0],rPairs[1,1]);
}


LineCNVFactors[0..6] = 0.0 : 1.0 : -1.0 : 256.0 : 0.0 : 3.0;

REAL Longu = 256;
REAL xmin = 10.000;
REAL xmax = 18.000;
REAL ymin = 9.200;
REAL ymax = 11.250;

REAL y = ymin;
INTEGER nbligne = 0;

while(y <= ymax)
{
        CreateLine(xmin : y :: xmax : y);
        y = y + 0.100;
        nbligne = nbligne +1;
}

MultipleExtract (TRUE);


INTEGER nbligne_correctes = 0;
for (i=0; i<nbligne ; i++)
{
        INTEGER nbbroche = 0;
        INTEGER nv_gris = 0;
        for (k=0; k<Longu ; k++)
        {
                        if (nv_gris == 0 && mLnLuminance[i,k] > 0)
                        {
                                nbbroche++;
                        }
                        nv_gris = mLnLuminance[i,k];
        }
        if (nbbroche == 20)
        {
                nbligne_correctes++;
        }
}
                //show(nbligne_correctes);

if( (nbligne_correctes/nbligne)*100 < 95 )
{
        show("Mauvais :-(");
}
else
{
        show("Bon :-)");
}
```

FIGURE 4.1 – Première macro de contrôle de la qualité des connecteurs

4.2 Seconde Macro : identifications et mesures de clés

```
ClearScreen();

REAL rPairs[,];
Histogram();
rPairs = GetAutoThreshold(ArROIHistogram,6,2,,ActiveLuminanceRange,);
if ( rPairs && 1 < GetShape(rPairs)[0] )
   {
// sucessful autothreshold

// set threshold
                Threshold(rPairs[1,0]:rPairs[1,1]);

// alternative which works even for color
                GrayToBinary(,rPairs[1,0],rPairs[1,1]);
}
Delete(rPairs);


LineCNVFactors[0..6] = 1000.0 : 0.50562 : 1000.0 : 0.51137 : 1000.0 : 1.5169;


CreateLine (122.59 : 222.96 ::
122.59 : 163.13);
MultipleMode = TRUE;
MultipleMode = TRUE;
DataCollection (0);
SetExport (mLnLength, optExtractFlag, TRUE);
SetExport (LnLength, optExtractFlag, TRUE);
SetExport (mLnLuminance, optExtractFlag, TRUE);
SetExport (LnLuminance, optExtractFlag, TRUE);
MultipleExtract (TRUE);
CloseWindow ("Measurement Explorer");


//variables utiles
REAL longueur_colonne = mLnLength*2;
INTEGER i = 0;
INTEGER taille_cle_pixel = 0;
INTEGER taille_cle = 0;
INTEGER bool_est_dans_cle = 0;
INTEGER bool_est_entre_cle = 0;

//affichage si nÊcessaire ‡ mettre ‡ 0;
INTEGER bool_affichage = 1;
INTEGER bool_affichage2 = 1;
INTEGER bool_affichage3 = 1;
INTEGER bool_affichage_longueur_colonne =1;
if (bool_affichage_longueur_colonne == 0)
{
        show(longueur_colonne);
        bool_affichage_longueur_colonne = 1;
}


for(i=0 ; i < longueur_colonne ; i++)
{
        if ( mLnLuminance[0,i] == 255 && bool_est_dans_cle == 0)
        {
                bool_est_dans_cle = 1;
                if (bool_affichage == 0)
                {
                        show("Est dans la clÈ");
                        bool_affichage = 1;
                }
        }
        if ( mLnLuminance[0,i] == 0 && bool_est_dans_cle == 1)
        {
                bool_est_dans_cle = 0;
                bool_est_entre_cle = 1;
                if (bool_affichage2 == 0)
                {
                        show("Est entre la clÈ");
                        bool_affichage2 = 1;
                }
        }
        if ( mLnLuminance[0,i] == 0 && bool_est_entre_cle == 1)
        {
                taille_cle_pixel++;
        }
        if ( mLnLuminance[0,i] == 255 && bool_est_entre_cle == 1)
        {
                if (bool_affichage3 == 0)
                {
                        show("Ecart fini");
                        show("taille pixel" +taille_cle_pixel);
                        bool_affichage3 = 1;
                }
                break;
        }
}

taille_cle = (taille_cle_pixel+1)/2;

show(taille_cle);
```

FIGURE 4.2 – Première macro de caractérisation des clés par leur taille