Diplôme de Master Professionnel en

Electronique – Informatique Industrielle – Instrumentation « E3I »

Dpt Génie Electrique et des Procédés

2012

Développement d'un procédé de vision industrielle pour l'installation automatique de montage des miroiteries

RENAULT

Lyon 1

Ozan UNLU

Rapport de stage de fin d'études

Période de stage: 09/04/2012 – 07/09/2012

Tuteur professionnel: Mr. Sadan DONMEZ

Tuteur pédagogique: Mme. Sophie CAVASSILA

REMERCIEMENTS

Je tiens à remercier dans un premier temps toute l'équipe pédagogique du département GEP et les intervenants responsables de ma formation, pour avoir assuré la partie théorique de celle-ci.

Je tiens à remercier mon tuteur pédagogique Madame Sophie Cavassila, Responsable du Master Electronique – Informatique Industrielle – Instrumentation, d'avoir accepté de suivre mon dossier et également pour ses remarques et ses conseils motivants lors de ma décision de l'endroit et du sujet de stage.

Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance aux personnes suivantes, pour l'expérience enrichissante et pleine d'intérêt qu'elles m'ont fait vivre durant ces cinq mois au sein du département R&D chez OYAK - Renault:

Monsieur Ismail Kuzey, Chef de Section Systèmes Automatisés, de m'avoir accueilli comme stagiaire au sein de son équipe.

Monsieur Sadan Donmez, mon maitre de stage, Ingénieur d'application automatisme, pour ses conseils et toute la confiance qu'il m'a accordée tout au long de mon stage.

Messieurs Sinan Tarkin, Hakan Uslu ainsi que l'ensemble du personnel de la section Systèmes Automatisés pour leur accueil sympathique et leur coopération professionnelle pendant ces cinq mois.

Table des matières:

Table des figures:

I. Introduction

L'objectif de mon stage réalisé au sein de l'usine d'automobiles OYAK – Renault, dans le Département R & D, la section « Conception et Développement des Systèmes Automatisés », était le développement d'un procédé de vision industrielle capable de mesurer en temps réel la hauteur et l'épaisseur du mastic d'étanchéité juste après son encollage sur la vitre d'automobile dans l' « installation automatique de montage des miroiteries ».

La première partie du stage était la lecture de documentations techniques afin d'avoir une idée générale sur l'application à concevoir et la méthode à adopter pour satisfaire au cahier des charges.

Après avoir déterminé la technique, les exemples fournis avec le logiciel HDevelop, qui ne concernent pas forcément notre application, ont été faits pour s'habituer à l'utilisation de ce logiciel.

J'ai tout d'abord développé cette application sous HDevelop utilisant des opérateurs de la bibliothèque de vision industrielle HALCON et puis testé pour plusieurs cas indésirables. Par la suite, il était demandé d'intégrer cette application à un environnement d'orienté objet afin de créer une interface graphique.

La programmation de l'interface graphique utilisateur a été réalisée en C#. Lors de l'intégration du code programmé en langage HDevelop, j'ai utilisé la bibliothèque de liens dynamiques (.dll) fournie avec le logiciel. En effet, cette « .dll » permet d'utiliser les opérateurs de la bibliothèque HALCON sous divers environnements de développement.

En outre, cette interface graphique devrait aussi être capable de communiquer avec le monde extérieur (l'Automate). Pour cela, j'ai intégré également le protocole « OPC Server » à mon interface graphique, qui permet d'établir la communication entre le PC (interface graphique) et l'automate (Schneider Premium).

Je présente dans ce rapport les activités réalisées au sein de l'usine d'automobiles OYAK – Renault, ses départements, ses produits, ainsi que mon application. Je parle ensuite de mon gain d'expérience pendant mon stage et les divers problèmes que j'ai rencontré lors du développement de ce procédé.

II. Présentation de l'entreprise

II. A. Renault en Turquie[1] :

En **1969**, OYAK (Ordu Yardimlasma Kurumu : Fonds de Pension des Forces Armées) et Yapı Kredi Bankasi (Banque Yapi Kredi) sont devenus partenaires pour créer Renault en Turquie. L'usine de Bursa, entrée en activité en **1971**, est le premier site industriel de Renault en dehors de l'Europe occidentale.

Aujourd'hui, Renault est opérationnel en Turquie avec sa filiale industrielle et commerciale. OYAK-Renault est la filiale de production et d'exportation de véhicules particuliers et organes mécaniques Renault fabriqués en Turquie. Au total, il y a **6 200** personnes réparties entre le siège social d'Istanbul et le site de Bursa.

L'usine de Bursa est l'un des **38** centres de production, en associant à un ILN (International Logistic Network), une unité de « Carrosserie-Montage », de « Mécanique » et de « Châssis ». Avec une capacité de production de **360 000** véhicules et de **450 000** moteurs par an, l'usine de Bursa, est aujourd'hui la quatrième plus grande entreprise privée de Turquie selon ISO (Istanbul Sanayi Odasi : Chambre d'Industrie d'Istanbul). Sur l'année 2011, elle a produit **330 994** véhicules, **291 797** moteurs, **261 830** boite de vitesses et **327 994/362 819** trains avant/arrière.

Edifiée sur un terrain de **534 530 m²**, dont **301 385 m²** de surface construite, le site industriel dispose d'une chaîne complète de fabrication carrosserie-montage. Bursa assure ainsi l'intégralité du processus de fabrication.

Depuis sa création, l'usine d'OYAK-Renault occupe une position stratégique au sein du groupe Renault. Elle est placée dans une position leader en production et en exportation depuis ces 13 dernières années dans l'industrie automobile turque grâce à ses investissements dans les ressources humaines et dans les installations de production.

OYAK-Renault et leurs fournisseurs ont collaboré en mettant en avant les normes de qualité. En 1996, OYAK-Renault a obtenu la certification ISO 9001 et est devenue le premier constructeur automobile turc disposant ce certificat.

Comme toutes les usines du Groupe Renault, OYAK-Renault opère dans toutes ses activités avec une conscience d'attitude écologiste. Avec les efforts de tous les membres de l'usine, la certification ISO 14 001 a été obtenue avec « zéro défaut » en Septembre 1999.

[1] Voir ANNEXE 1: OYAK – Renault en bref

II. B. Produits:

OYAK-Renault produit des véhicules ainsi que des organes mécaniques.

Les véhicules fabriqués au sein d'OYAK-Renault sont :

Figure 1: Clio Symbol

❖ Clio Symbol: Symbol aussi appelée Thalia sur certains marchés, est une berline tricorps qui est produite en Turquie depuis 2008 et en Argentine depuis 2009. Elle s'impose sur les marchés d'Europe Centrale et orientale, de Russie, de Turquie, du Maghreb et d'Amérique du Sud.

❖ Fluence: L'usine de Bursa a capitalisé sur son expérience acquise dans la fabrication des deux précédentes générations de tricorps Mégane. La production de la nouvelle berline tricorps « Fluence » a démarré en Septembre 2009. Elle possède quatre niveaux d'équipements avec deux harmonies intérieures et treize teintes de caisse.

Figure 2: Fluence

❖ Mégane III: Après la production de Mégane berline et Clio Symbol, la production de Mégane III a commencé en 2009 sur le site de Bursa. Elle offre un niveau de confort et d'ergonomie inédit et dispose de motorisations qui associent performance, économie et respect de l'environnement.

Figure 3: Mégane III

❖ Clio III: La production de la Clio III chez OYAK-Renault a démarrée en janvier 2006 et sa gamme est complétée avec le « GRAND TOUR / ESTATE » en Septembre 2007. Les Séries Limitées comme « JEUNE/ EXTREME/ RIPGIRL2/ EXCEPTION/ RUGBY » complètent l'offre de OYAK-Renault. En Février 2009, la Clio III est passée en Phase-2 avec la nouvelle face avant, les nouvelles roues alu, les nouveaux systèmes radios et nouvelles selleries.

Figure 4: Clio III

Figure 5: Fluence Z.E.

❖ Fluence Z.E. : La version électrique de Fluence est plus longue de 13 cm que sa version thermique. La partie arrière a été entièrement redessinée pour permettre l'intégration des batteries. Elle est produite seulement sur le site de Bursa et commercialisée depuis Décembre 2011.

❖ Clio IV: La production en série de la nouvelle génération de la Clio, l'un des modèles phares de la marque au losange, vient de commencer à Bursa et à Flins. Les prévisions de capacité de production pour ce modèle sont de 280 000 véhicules/an. Aujourd'hui les organes mécaniques fabriqués au sein du département «Mécanique » chez OYAK – Renault sont des moteurs, des boites de vitesses et des châssis.

Figure 6: Clio IV

II. C. Processus :

Le site industriel de Bursa dispose d'une chaîne complète de fabrication en carrosserie montage. OYAK-Renault assure ainsi l'intégralité du processus de fabrication : emboutissage, tôlerie, peinture, montage, usinage et assemblage mécanique organe et châssis. Une seule ligne de montage accueille indifféremment les Clio Symbol, Clio III, Clio IV, Mégane III, Fluence et Fluence Z.E.

II. C. 1. Carrosserie – Montage :

L'atelier de la carrosserie montage est composé de quatre départements représentant les étapes de production d'un véhicule.

Après l'arrivée des tôles en usine, l'**« Atelier d'Emboutissage »** les transforme en panneaux de porte, toit, capot avant et arrière, etc. L'**« Atelier de Tôlerie »** les assemble par soudage. Ainsi le châssis de la voiture se forme.

Pour éviter les dommages naturels dus à la pluie, la boue, la poussière, l'humidité, etc., et pour donner une belle vue, le châssis passe dans l' **« Atelier de Peinture »** pour les opérations de protection contre la corrosion, d'apprêt, de peinture et de vernis.

Dans l' **« Atelier de Montage »** ; les sièges, le volant, les pneus, les phares, le tableau de bord, les installations électriques, le moteur, la boite de vitesses et les autres pièces mécaniques sont montés à la carrosserie peinte. Après l'achèvement de toutes ces opérations sur le véhicule, l' **« Atelier de Finition »** réalise les dernières retouches comme le paramétrage des phares, de moteurs, etc. Finalement, il les délivre à sa filiale Renault-MAIS, qui est responsable des ventes et de distribution en Turquie et à CAT pour l'international.

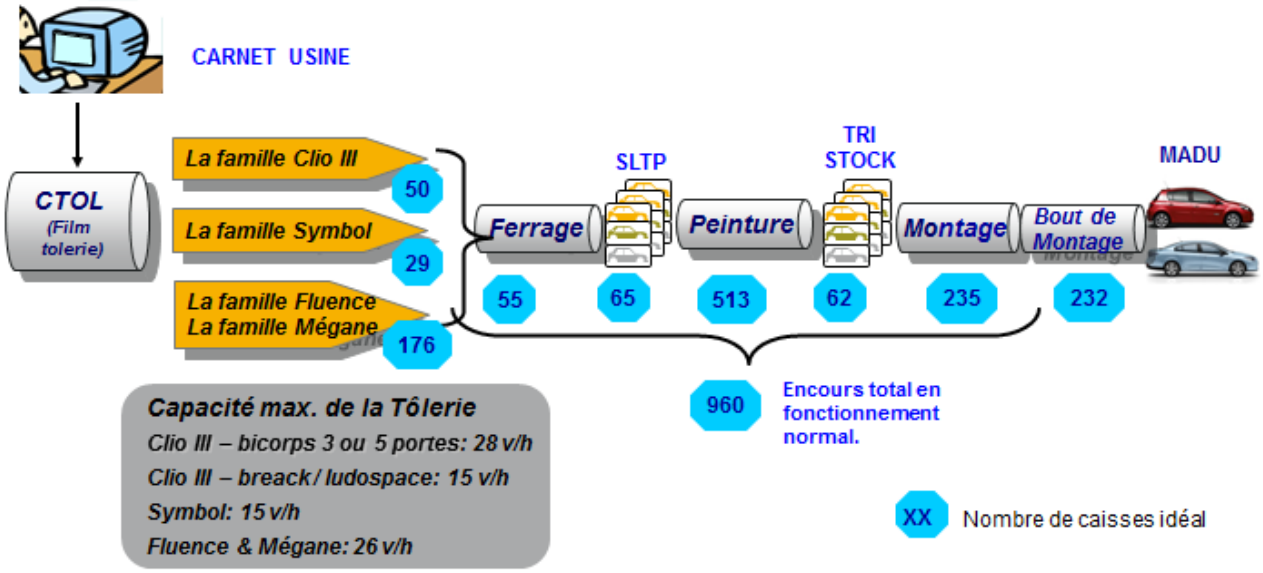


Figure 7: Processus de Fabrication selon les types de véhicules

En dehors de ces activités de fabrication, trois autres départements assument les fonctions de supports pour le site industriel d'OYAK-RENAULT :

1. DEPARTEMENT LOGISTIQUE INDUSTRIELLE:

Ce département assure les activités de l'usine en exerçant, dans une intégralité totale, toutes les fonctions de l'organisation logistique sous les sections suivantes :

- Section Documentation & Arrêt-Départ & Fin de Série
- Section Programmation & Gestion Véhicule
- Section Performance Logistique et Ingénierie Logistique Flux Pièces
- Section Approvisionnement
- Section Flux Interne

2. DEPARTEMENT DE LA MAINTENANCE ET DES SERVICES TECHNIQUES :

Ce département est constitué de 4 ateliers principaux en fonction des métiers. Il existe un Laboratoire Electronique, un secteur de Méthodes Maintenance et un atelier Maintenance Centrale comprenant également l'unité de Gestion de l'Environnement.

3. DEPARTEMENT DE PERFORMANCE INDUSTRIEL:

Ce département assure ses activités dans le cadre du Système de Production Renault (SPR) 10 de l'Usine qui a pour objectif de porter le système industriel du Groupe Renault au meilleur niveau de performance mondial. Il permet de standardiser les meilleures pratiques connues du moment dans les domaines de l'organisation et du management en vue d'une amélioration permanente au poste de travail pour optimiser le temps et les conditions de travail.

II. C. 2. Mécanique et Châssis :

L'atelier mécanique produit des moteurs, des boîtes de vitesse et des autres pièces mécaniques.

1. DEPARTEMENT MOTEUR:

Depuis sa création, OYAK-Renault a produit des moteurs à essence. À partir de 1999, l'usine a intégré une nouvelle ligne de montage pour produire les premiers moteurs « écologistes ». Pendant les 10 premières années, les moteurs sortant de ces lignes ont équipé plusieurs modèles de Renault, de Dacia et de Nissan en France, en Espagne, en Slovénie, en Colombie, en Argentine, en Malaisie, en Roumanie, en Iran ainsi qu'en Turquie.

2. DEPARTEMENT BOITES DE VITESSES:

Le Département Boites de vitesses réalise l'usinage et l'assemblage de 3 différents types de boites de vitesses équipant les différents modèles de Renault comme Symbole, Clio, Mégane HB et Fluence.

Le procédé de fabrication de boite de vitesses est constitué de tournage, de fraisage et de meulage ainsi que du traitement thermique et d'usinage pignonnerie. Le Département « Boites de vitesses » assure plus de 80% du besoin de l'Usine Carrosserie-Montage.

3. DEPARTEMENT DE CHASSIS:

Le Département de Châssis réalise l'usinage de disques de frein avant et arrière, de moyeu tambour, de moyeu disque et de porte fusée ; le soudage de berceau, d'essieu arrière et l'assemblage des trains avant et arrière pour les véhicules produits.

Les fonctions supports (le Département Logistique Industrielle, le Département de la Maintenance et de Services Techniques ainsi que le Département de Performance Industriel) fonctionnent de même façon pour l'usine Mécanique et Châssis.

III. Problématique

III. A. Objectif :

Selon la politique de l'Usine d'automobiles Oyak-Renault, la direction technique voudrait standardiser toutes ses installations et ses chaînes de production afin d'optimiser le contrôle qualité sur ses produits. A l'heure actuelle, il y a toujours de nombreuses installations qui présentent des points non-standards au sein de l'usine.

Dans le cadre du projet, la section « Conception et Développement des systèmes automatisés » a envisagé de concevoir un système de vision industrielle qui servira à mesurer en temps réel la hauteur et l'épaisseur du mastic d'étanchéité encollé sur les bords intérieurs des vitres (Pare-brises, Lunettes arrière et Custodes) avant de les monter sur les voitures.

D'un côté, ce système aura pour but d'intégrer un nouveau standard à « l'installation automatique de montage des miroiteries »[2] envisageant qu'il y aura de différentes hauteurs et épaisseurs de mastic pour chaque modèle de vitres et bien évidemment des différentes valeurs limites à respecter.

De l'autre côté, ce système contrôlera la conformité du mastic encollé en comparant la mesure effectuée aux références (valeurs limites) prédéfinies et validées par la direction technique.

III. B. Méthode :

La technique adoptée pour pouvoir déterminer la hauteur et l'épaisseur du mastic d'étanchéité s'appelle la méthode « Sheet of Light » (ou « Triangulation »). Cette méthode permet de construire un profil en 3D de l'objet à l'aide d'un ensemble composé d'une caméra et un laser.

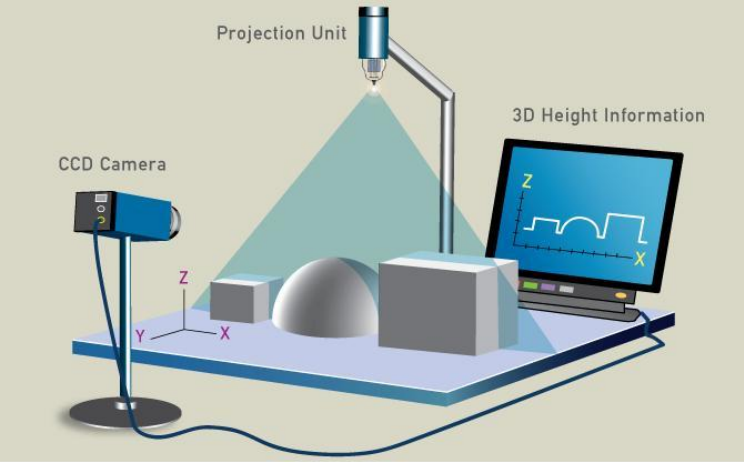


Figure 8: Configuration "Sheet of Light"

Le principe de fonctionnement de la technique « Sheet of Light » consiste à assembler les coordonnées de tous les pixels lumineux, couvrant la coupe (ou section), entre des images successivement acquises lors du mouvement du laser sur l'objet. L'assemblage des coordonnées des pixels lumineux sert à déterminer la longueur, la largeur et l'épaisseur de l'objet concerné afin de construire son profil en 3D.

[2] Voir ANNEXE 2: Photos de l'installation automatique de montage des miroiteries

Dans le cadre de notre projet, nous nous intéresserons plutôt à l'extraction des données de la hauteur et de l'épaisseur (2D) de l'objet que son profil en 3D. De ce fait, au lieu d'attendre la fin de l'acquisition de toutes les images, nous allons acquérir et traiter chaque image avant de passer à la suivante.

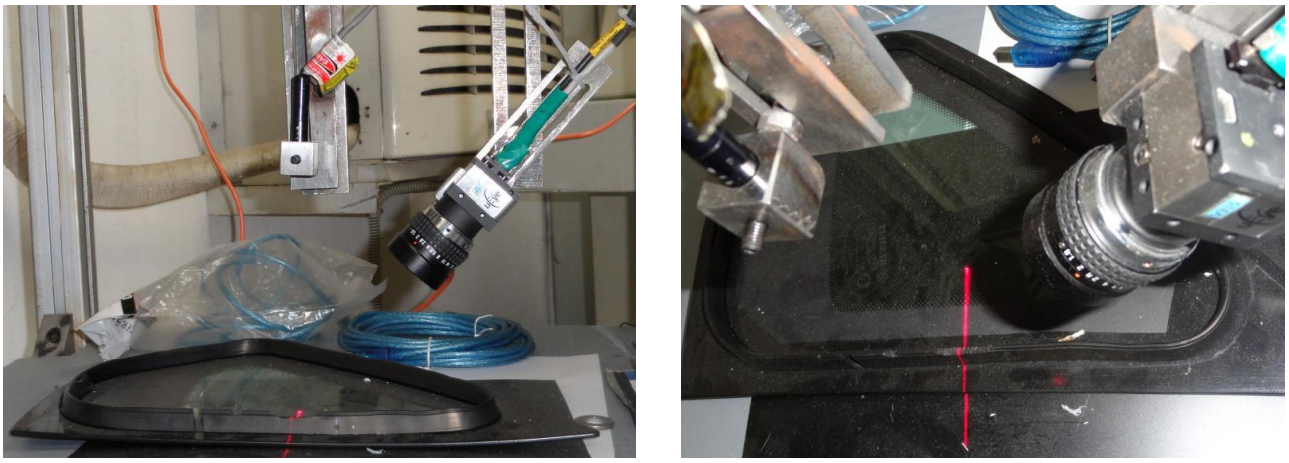


Figure 9: Configuration de l'ensemble "caméra – laser"

En raison de la forme du mastic d'étanchéité, le faisceau lumineux du laser tombant sur le mastic ressemble à un triangle sur l'image acquise. En outre, ces images consécutives seront traitées une par une par un algorithme développé en HDevelop[3].

En ce qui concerne le traitement ayant pour but d'extraire les données relatives à la hauteur et l'épaisseur du mastic, nous avons envisagé de déterminer les coordonnées du sommet et de la base du faisceau laser. A partir de ces coordonnées, chacune déterminée par rapport à la même origine, il serait possible de calculer la hauteur et l'épaisseur du mastic.

Le temps d'acquisition et de traitement d'une image devrait être le plus court possible afin que notre système puisse passer à l'image suivante. Si la vitesse de déplacement de l'ensemble « caméra – laser » sur l'objet à mesurer n'est pas synchrone au temps d'acquisition et de traitement d'une image, il se pourrait qu'il y ait des cas non-évalués ou bien des cas évalués plusieurs fois. Ces deux possibilités étant désagréables pour l'application, il serait alors très important d'assurer la synchronisation entre le temps d'acquisition et de traitement d'une image et la vitesse de l'outil servant à déplacer l'ensemble « caméra – laser » sur l'objet à mesurer.

Dans le chapitre suivant, l'algorithme du système de vision industrielle sera abordé en détail.

[3] Voir ANNEXE 3: Présentation de l'environnement Halcon / HDevelop

IV. Algorithme du système de mesure[4]

IV. A. Initialisation de la caméra et l'acquisition de l'image :

Dans un premier temps, il est nécessaire d'initialiser la caméra une fois pour toutes en appelant la fonction correspondante. Après l'avoir initialisée, nous sommes prêts à acquérir notre première image.

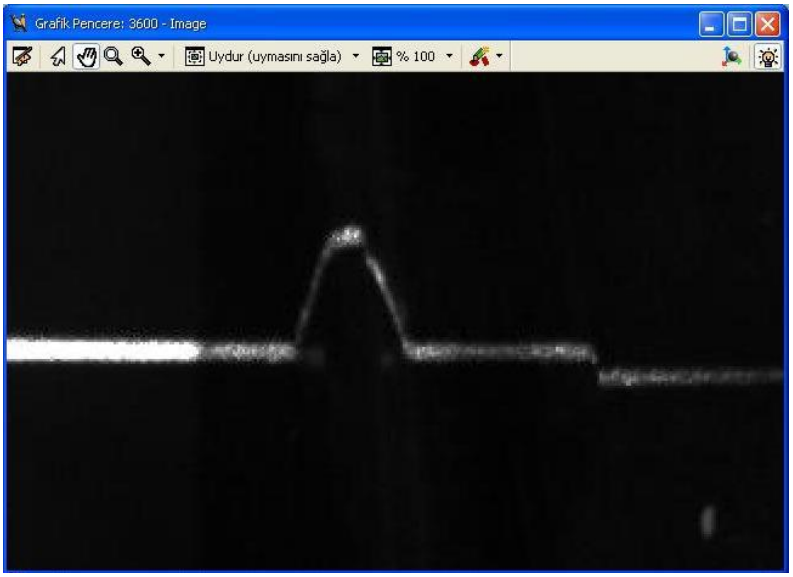


Figure 10: Image du faisceau laser

Comme nous ne désirons examiner que le faisceau lumineux du laser, il est impératif d'assurer le réglage fin du contraste dans la fonction qui initialise la caméra et la calibration de la lentille (lens) de manière à ce que notre système ne soit pas perturbé par la lumière ambiante. Pour cela, plusieurs essais ont été effectués.

IV. B. Création de la région d'intérêt (Region of Interest) :

HDevelop nous permet de tracer un rectangle interactivement en appelant sa méthode appropriée. Cette méthode prend quatre paramètres de sortie : Row1, Column1, Row2, Column2 et un paramètre d'entrée : 3600 (WindowHandle). Ces paramètres de sortie sont initialement nuls.

N .B : Si nous voudrions travailler sur la fenêtre pré-générée de HDevelop, certaines méthodes attendent le nombre 3600 comme paramètre d'entrée qui signifie WindowHandle pour cette fenêtre. Si l'utilisateur voudrait créer sa propre fenêtre ou même travailler sur une autre fenêtre en parallèle avec celle de 3600, il pourrait fermer celle-ci et en créer une autre ou même en créer une autre sans fermer celle-ci. Il est à noter que la fermeture et la génération des fenêtres s'effectuent par lignes de commande.

Nous traçons et ajustons le rectangle avec le clic gauche de la souris et pour en finir HDevelop attend un clic droit. Après le clic droit de l'utilisateur, cette méthode retourne les coordonnées dans ces paramètres de sortie.

[4] Voir ANNEXE 4: Algorithme en lignes de code « HDevelop »

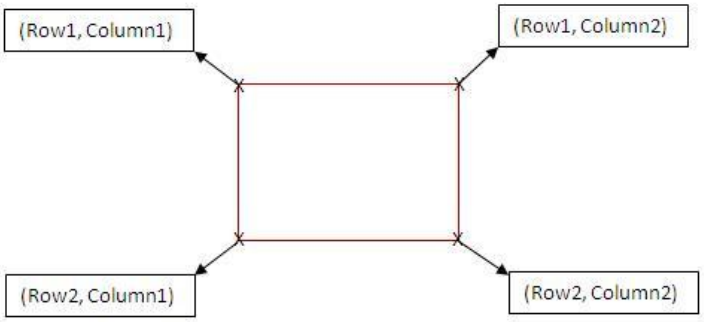


Figure 11: Région d'Intérêt (ROI)

La région d'intérêt (ROI) sera désormais l'environnement dans lequel nous allons travailler. Le but de la ROI est d'examiner juste la partie qui nous intéresse et éliminer le reste. Grâce à ce principe, nous gagnons sur le temps d'exécution lors du traitement en réduisant l'image à une région spécifique.

Après avoir tracé le rectangle de la ROI et terminé l'opération, nous obtenons les coordonnées du rectangle dans la fenêtre de contrôle.

Row1	182.0
Column1	282.5
Row2	271.0
Column2	390.5

Figure 12: Coordonnées de la Région d'Intérêt (ROI)

Une fois que nous avons tracé interactivement notre ROI et obtenu ses coordonnées, nous pourrions mettre en commentaire la méthode qui permet le traçage interactif et utiliser une autre méthode qui génère un rectangle statique. Cette méthode attend quatre paramètres d'entrée et un paramètre de sortie. Les paramètres d'entrée seront les coordonnées que nous avons pu extraire avec la méthode précédente et le paramètre de sortie sera un objet de type « Region » (type Halcon) initialement nul.

Voici la réponse de cette méthode :

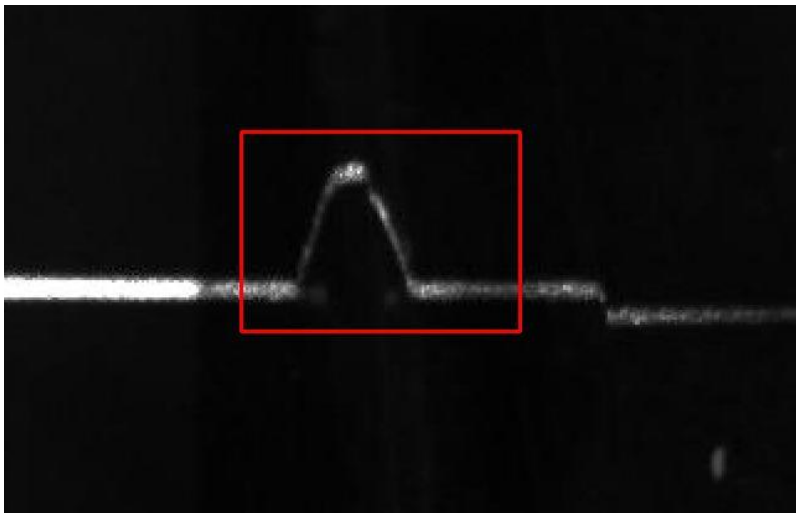


Figure 13: Tracé de la Région d'Intérêt (ROI)

IV. C. Amélioration de la visibilité du faisceau lumineux dans la ROI par filtrage :

Comme par la suite nous allons travailler dans la ROI, il nous faudra une meilleure différence de couleurs afin de détecter le plus possible le faisceau lumineux du laser. La bibliothèque HALCON possède plusieurs moyens de filtrage permettant de rendre la couleur blanche plus saillante.

Notre premier filtre (mean_image) sera celui qui lisse par moyennage. Selon sa définition, ce filtre effectue un lissage linéaire avec les valeurs de gris de tous les pixels de la ROI. La matrice du filtre a une taille MaskHeight x MaskWidth et le résultat de la convolution est par la suite divisé par MaskHeight x MaskWidth. Pour le traitement des frontières, les valeurs de gris sont reflétées dans les bords de l'image.

Voici la réponse du filtre de lissage par moyennage :

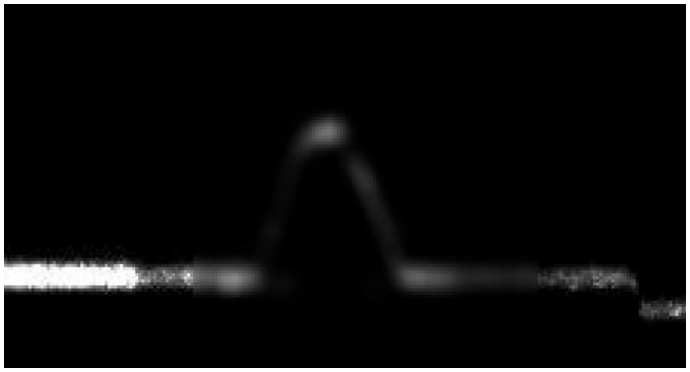


Figure 14: Réponse du filtre “mean_image”

Notre second filtre (emphasize) sera celui qui améliore le contraste de l’image se trouvant dans la ROI. Ce filtre met l'accent sur les zones à haute fréquence de l'image (les bords et les coins). L’image qui en résulte sera plus nette. Ce filtre prend comme paramètre d’entrée la sortie du premier filtre.

Voici la réponse du second filtre :

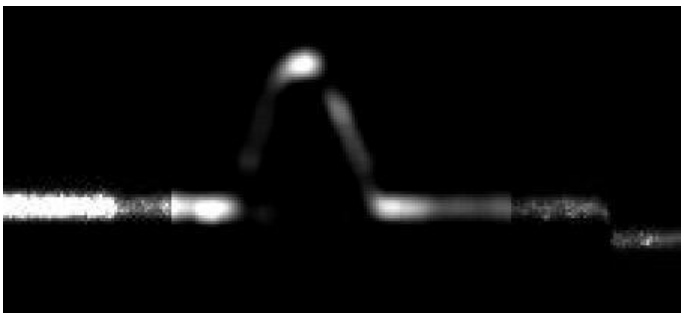


Figure 15: Réponse du filtre “emphasize”

N.B : Dans la documentation technique de la bibliothèque HALCON ces deux filtres (mean_image et emphasize) sont généralement utilisés l’un après l’autre comme nous les avons procédés ici. En outre, cette documentation technique donne également les paramètres par défaut (la taille de leurs masques) de ces filtres.

Après avoir essayé et observé plusieurs combinaisons de ces paramètres spécifiques, nous avons enfin abouti à un stade plutôt satisfaisant mais il se peut qu’il y ait d’autres moyens pour avoir un

meilleur résultat. En effet, nous avons examiné plusieurs exemples fournis avec HDevelop utilisant ces filtres afin de comprendre leurs comportements et leurs domaines d'application.

IV. D. Seuillage de l'image dans la ROI :

Selon sa définition, étant la méthode la plus simple de segmentation d'image, le seuillage (thresholding) sélectionne les pixels de l'image d'entrée dont les valeurs de gris « G» satisfont la condition suivante :

$$Min\ Gray \leq G \leq Max\ Gray$$

Tous les points d'une image remplissant cette condition seront retournés en une seule région. Quant à notre objectif, nous allons essayer de détecter le faisceau lumineux du laser au maximum dans notre ROI. Comme nous désirons détecter la couleur blanche sur le fond noir, notre seuil haut (Max Gray) sera alors 255.

En ce qui concerne le seuil bas (Min Gray), il est difficile de fixer une valeur adaptée pour tout environnement vis-à-vis de la différence au niveau de la lumière ambiante mais dans notre milieu de travail là où la lumière ambiante est éliminée par le réglage du contraste de la caméra et la calibration de sa lentille, nous avons pu détecter un nombre de pixels blancs raisonnable avec un seuil bas de 65.

N.B : Par définition des niveaux de gris, la couleur noire pure est représentée par 0 et la couleur blanche pure par 255.

Voici la réponse de la méthode de seuillage (thresholding) pour $65 \leq G \leq 255$:

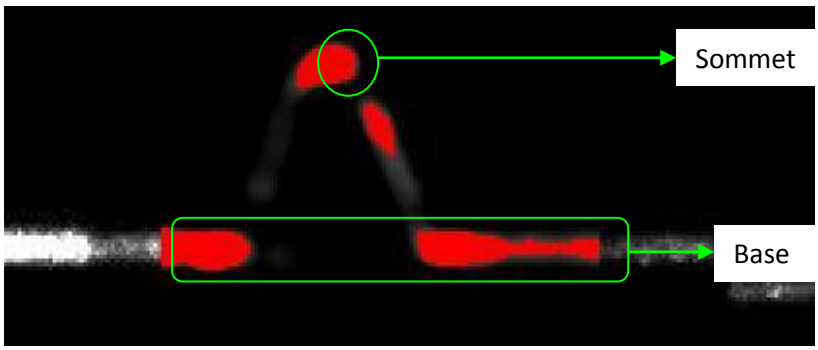


Figure 16: Réponse de la méthode "threshold"

En effet, nous cherchons à déterminer les coordonnées du sommet et de la base du faisceau laser. Donc nous avons dû faire plusieurs essais en ajustant les paramètres des filtres afin d'avoir un maximum de pixels détectés par la méthode de seuillage.

Pour avoir une idée plus claire sur l'avantage des filtres, nous pouvons comparer les deux cas.

Voici la comparaison des seuillages après les filtres et sans les filtres :

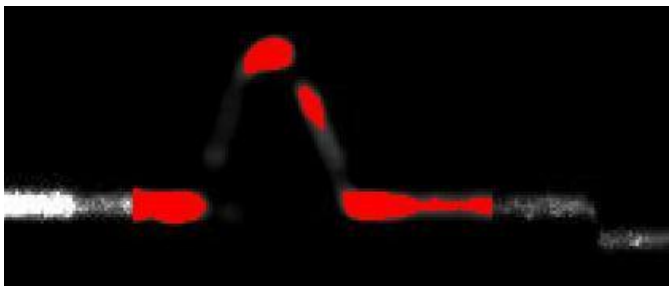


Figure 17: Seuillage après les filtres (mean_image et emphasize)

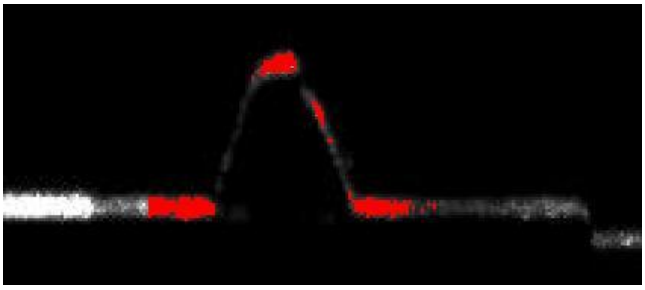


Figure 18: Seuillage sans les filtres

Nous remarquons ici qu'après avoir appliqué les filtres, il y a beaucoup plus de pixels détectés et la continuité de ces pixels sur chacun des segments nous servira par la suite pour en tirer les coordonnées du sommet et de la base du faisceau laser.

IV. E. Elimination des pixels parasites par sélection surfacique (Cas exceptionnel):

Malgré la présence des filtres, il est possible qu'il y ait des pixels blancs non-détectés dus à la mauvaise homogénéité du faisceau laser. Comme nous préférons une continuité des pixels sur chacun des segments, ces pixels parasites pourraient perturber le bon fonctionnement de notre système.

Voici un cas avec une discontinuité sur le segment droit de la base :

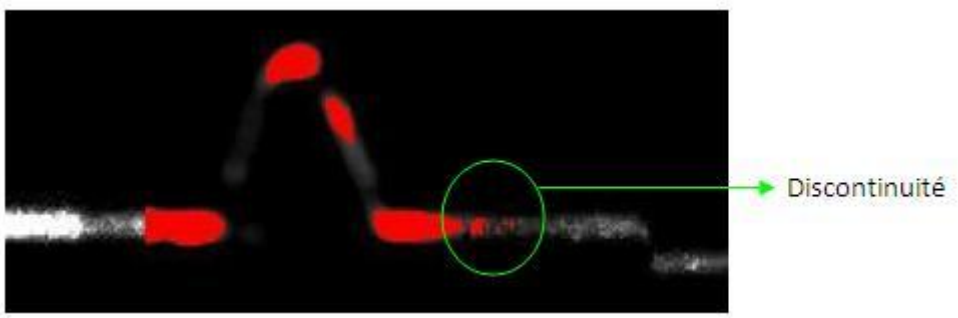


Figure 19: Présence de la discontinuité

Suite aux plusieurs essais, nous avons constaté que l'apparition de la discontinuité après le filtrage se produit rarement avec une aire maximale ne dépassant pas 20 pixels. Pour pouvoir filtrer ces pixels parasites probables, nous avons utilisé tout d'abord la méthode qui permet de fractionner l'ensemble des segments dans la ROI car il nous faudrait l'aire de chaque segment et non pas celle de l'ensemble. Par la suite nous nous sommes servis d'une autre méthode qui permet de filtrer les segments selon leurs aires dans la ROI.

Comme nous désirons éliminer les segments ayant des aires inférieures à 20 pixels, nous informons notre méthode de façon à ce qu'elle ne garde que les segments qui ont des aires entre 20 et 1000 pixels.

La valeur maximale (1000 pixels) n'est pas décisive mais sachant que chacun des segments utiles ne dépassant pas 800-900 pixels, nous avons fixé le seuil haut de notre filtre d'aire à 1000 pixels.

Voici notre image actuelle :

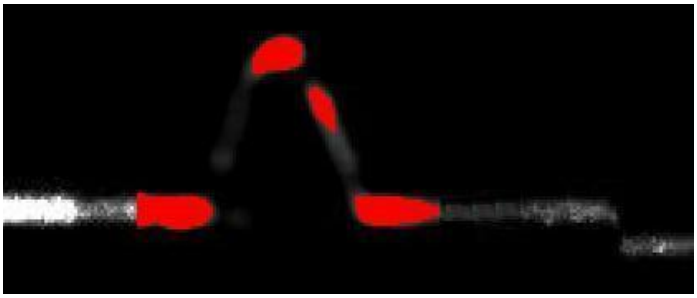


Figure 20: Discontinuité éliminée

Avant de passer à la détection des coordonnées du sommet et de la base, nous avons prévu d'effectuer une dernière amélioration.

Voici la comparaison des images de la base avant et après l'amélioration :

Figure 22: Avant l'amélioration "closing_rectangle"

Figure 21: Apres l'amélioration "closing_rectangle"

Nous avons utilisé ici une méthode qui permet de compléter un segment avec un élément structurant rectangulaire. Cette méthode prend comme paramètres d'entrée la largeur et la hauteur de cet élément rectangulaire et elle parcourt sur tous les segments afin de détecter et fermer les trous rectangulaires.

Le fait d'avoir appliqué cette amélioration nous sera très utile quand nous allons travailler sur la détection des coordonnées de la base.

Nous avions fractionné nos segments utiles pour pouvoir éliminer les pixels parasites (cas exceptionnel) et fermer les trous sur les segments avec un élément structurant prédéfini. Nous pouvons désormais passer à la détection de nos coordonnées.

IV. F. Détection des coordonnées du sommet et de la base :

La bibliothèque HALCON possède une méthode très utile pour notre application qui parcourt une région donnée ligne par ligne en ordre croissant du numéro de lignes (du haut vers le bas). Chaque ligne est examinée de gauche à droite (ordre croissant du numéro de colonnes).

Si l'opérateur rencontre un point de segment sur la ligne, le numéro de la ligne correspondante ainsi que les numéros de colonnes, le début et la fin du segment sur la même ligne, sont stockés dans trois tableaux différents. Sinon l'opérateur passe à la ligne suivante sans rien stocker dans les tableaux.

Voici une illustration du principe de fonctionnement de cette méthode :

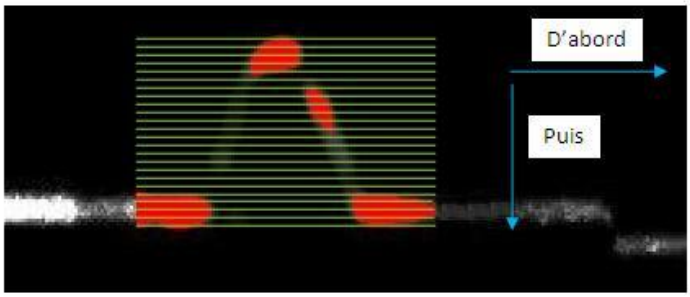


Figure 23: Principe de fonctionnement de la méthode "get_region_runs"

Zoom sur le sommet du mastic pour les voir plus en détail :

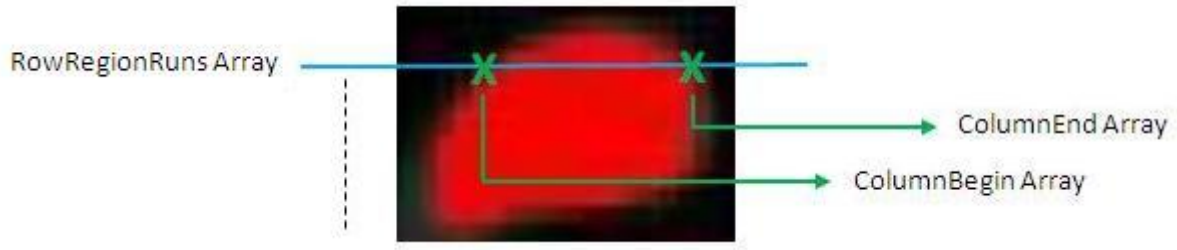


Figure 24: Zoom sur le sommet du mastic

Une fois que cette méthode a été appelée, ces trois tableaux sont remplis très rapidement. Le premier élément du tableau « RowRegionRuns » va nous donner le numéro de ligne du sommet. De même logique, le dernier élément du tableau « RowRegionRuns » nous donne le numéro de ligne de la base. Nous voyons que les numéros de ligne du sommet et de la base sont très facilement identifiables. Nous pouvons donc en déduire la hauteur en pixel du mastic étanche en soustrayant le numéro de ligne du sommet du numéro de ligne de la base.

L'aperçu des tableaux remplis par la méthode :

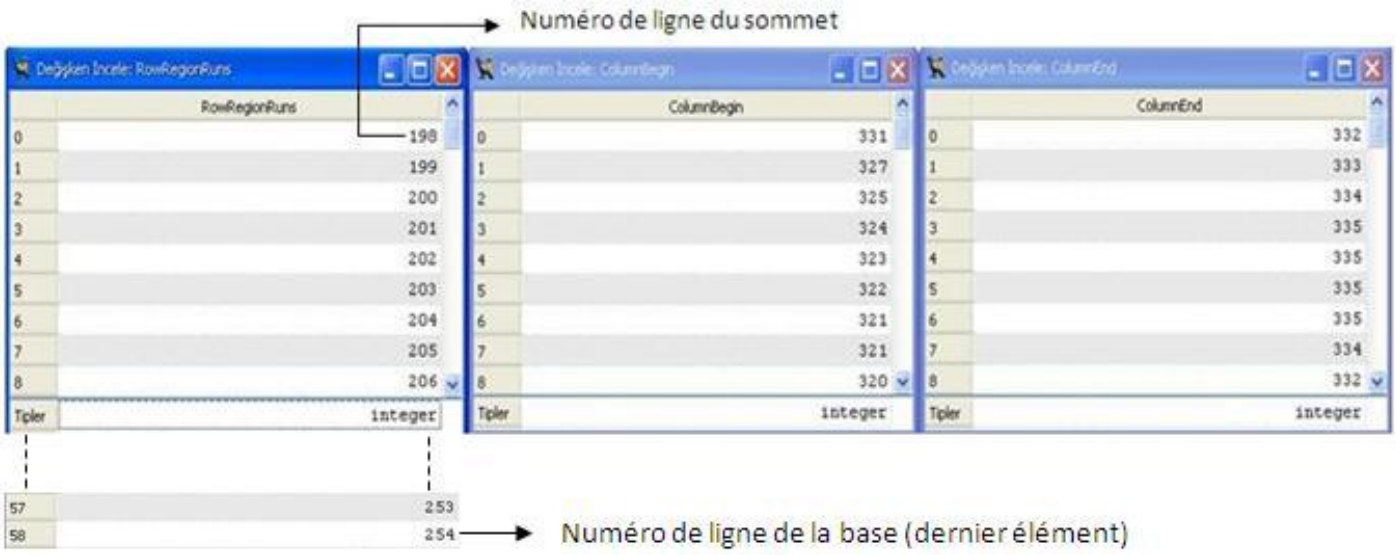


Figure 25: Aperçus des tableaux "RowRegionRuns – ColumnBegin – ColumnEnd"

Le résultat de la soustraction :

$$Hauteur\ (en\ pixel) = 254 - 198 = 56\ pixels$$

Grâce aux filtres appliqués et au seuillage, une simple opération avec les éléments du tableau « RowRegionRuns » nous suffit pour calculer la hauteur (pour l'instant en pixel) du mastic étanche.

Quant au calcul d'épaisseur de la base du mastic, il nous faudrait un algorithme robuste qui prend en compte plusieurs cas défavorables.

L'idée que nous avons prévue pour ce calcul se base sur la détection de pair de ligne dans les derniers éléments du tableau « RowRegionRuns ».

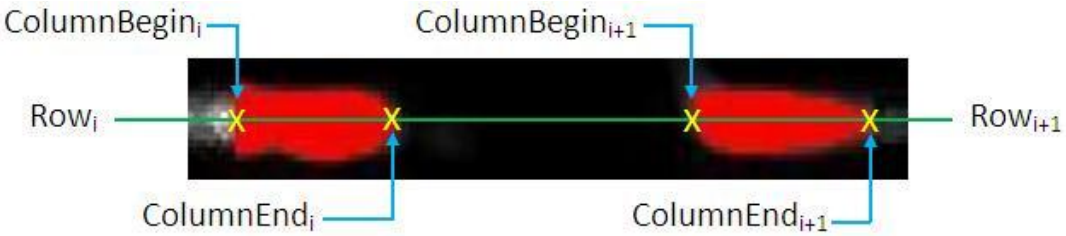


Figure 26: Coordonnées de la base du mastic

Lorsque la méthode parcourt sur la base du faisceau laser, considérant qu'il y a deux segments ne présentant pas de discontinuité dans un premier temps sur la ligne, elle détecte deux débuts et deux fins sur la même ligne. Puisqu'il s'agit de deux "ColumnBegin" et deux "ColumnEnd" différents, cette méthode va copier deux fois le même numéro de ligne l'un après l'autre dans le tableau "RowRegionRuns".

RowRegionRun Array	ColumnBegin Array	ColumnEnd Array
⋮	⋮	⋮
Paire de lignes — Row$_i$	ColumnBegin$_i$	ColumnEnd$_i$
Paire de lignes — Row$_{i+1}$	ColumnBegin$_{i+1}$	ColumnEnd$_{i+1}$
⋮	⋮	⋮

Figure 27: Tableau paire de lignes 1

Comme l'image ci-dessus l'illustre, nous avons effectué de nouveau une soustraction entre les colonnes pour en tirer l'épaisseur du mastic étanche:

$$Epaisseur\ (en\ pixel) = ColumnBegin[i+1] - ColumnEnd[i]$$

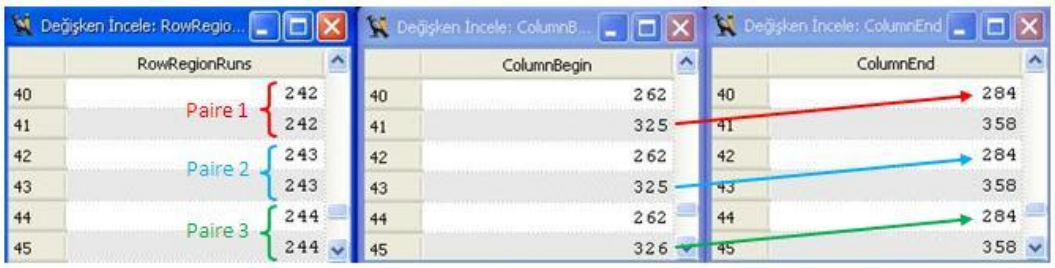


Figure 28: Tableau paire de lignes 2

Faisons les calculs d'épaisseur du mastic selon la comparaison ci-dessus :

$Paire\ 1 = Paire\ 2$: $Epaisseur\ (en\ pixel) = 325 - 284 = 41\ pixels$

$Paire\ 3$: $Epaisseur\ (en\ pixel) = 326 - 284 = 42\ pixels$

Nous remarquons qu'il y a une différence de 1 pixel entre la paire 3 et paire 1 (ou paire 2). Cette différence sera quasi nulle lorsque nous allons convertir les résultats de nos mesures en millimètre.

N.B : Il est à noter que nous ne prenons pas la dernière paire de ligne dans le tableau « RowRegionRuns » car les toutes dernières paires de ligne pourraient présenter une différence plus importante que 1 pixel entre elles due à la diffraction du faisceau laser sur le mastic.

Voici les dernières paires de ligne du tableau « RowRegionRuns » :

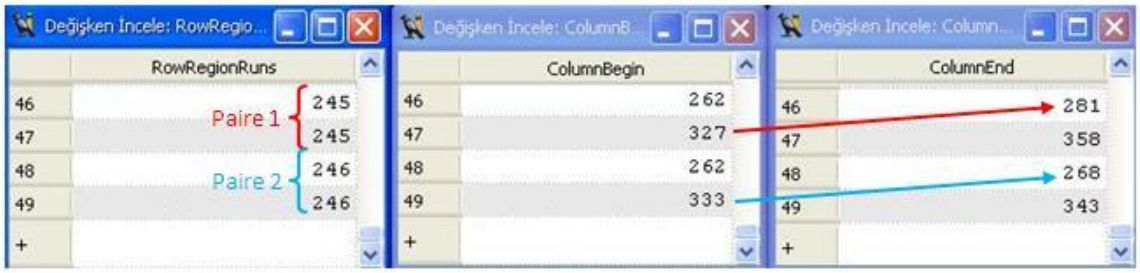


Figure 29: Dernières paires de lignes du tableau "RowRegionRuns"

$Paire\ 1$: $Epaisseur\ (en\ pixel) = 327 - 281 = 46\ pixels$

$Paire\ 2$: $Epaisseur\ (en\ pixel) = 333 - 268 = 65\ pixels$

Cette fois-ci nous remarquons une différence de 19 pixels entre les deux dernières paires de ligne. Sur plusieurs essais, nous avons toujours obtenu une différence très importante entre les toutes dernières paires de ligne. Cette différence importante apparait également sur les toutes premières paires de ligne de la base du faisceau laser.

Figure 30: Positions des premieres et dernières paires de lignes au niveau de la base

Nous voyons sur l'image ci-dessus que les paires du milieu présentent peu de différences par rapport aux premières et dernières paires. Comme la configuration de la caméra et l'épaisseur du faisceau laser sont fixes pour toutes les mesures, nous obtenons autour de 10 paires de ligne maximum au niveau de la base du faisceau laser.

Notre algorithme ne devrait donc pas considérer les derniers, par exemple, 6 éléments du tableau « RowRegionRuns » avant qu'il commence à chercher la paire. Nous nous sommes donc fixés une limite basse pour notre point de départ. Il nous faudrait aussi une limite haute pour ne pas que notre algorithme dépasse lors de sa recherche l'épaisseur Δe, par exemple 17ieme élément en partant de la fin.

IV. G. Condition générale pour procéder à la mesure :

Cette limite haute sera également notre condition globale, c'est-à-dire si le nombre d'éléments du tableau « RowRegionRuns » est supérieur à 17, l'algorithme peut procéder à la mesure. Sinon nous voulons qu'il affiche à l'utilisateur un message d'erreur signifiant qu'il y a un problème au niveau de la longueur des tableaux et il passe à l'image suivante.

N.B : La signification de cette erreur est que notre système n'a même pas pu détecter la base du faisceau laser dans la ROI puisqu'il y a moins de 20 éléments dans le tableau donc il est inutile de continuer.

Voici notre message d'erreur sur l'écran de mesure :

Figure 31: Message d'erreur

Normalement, le fait d'avoir déterminé la ROI correctement, appliqué les filtres « mean_image » et « emphasize » et paramétré le seuillage raisonnablement nous conduit à un nombre d'éléments supérieur à 17 dans le tableau « RowRegionRuns ».

Par ailleurs, il peut y avoir deux cas défavorables à cause desquels nous pourrions rencontrer ce problème lors de la mesure:

- Disparition du faisceau laser dans la ROI en raison de la présence d'un obstacle devant la caméra.
- Mauvaise calibration du laser et/ou de la lentille (lens) pour le milieu de mesure.

Après avoir testé la condition globale, nous pouvons alors procéder à la mesure. La première étape de notre algorithme consiste à affecter les sixième et septième éléments en partant de la fin du tableau « RowRegionRuns » aux deux variables, Row_1 et Row_2.

N.B : Il était dit précédemment que notre algorithme ne devrait pas considérer les tous derniers éléments. Pour cette raison, nous allons commencer à chercher notre paire à partir du sixième élément en partant de la fin.

$$Row\_1 = RowRegionRuns[nombre\ d'éléments - 7]$$

$$Row\_2 = RowRegionRuns[nombre\ d'éléments - 6]$$

Nous avons affecté deux éléments consécutifs du tableau « RowRegionRuns » aux deux variables, déclarées au moment de l'affectation, mais nous ignorons bien évidemment si ces deux éléments constituent une paire.

La prochaine étape sera de tester si les variables Row_1 et Row_2 sont égales.

IV. G. 1. Procédure pour l'égalité non-trouvée:

Dans un premier temps nous considérons le cas où Row_1 est différente de Row_2.

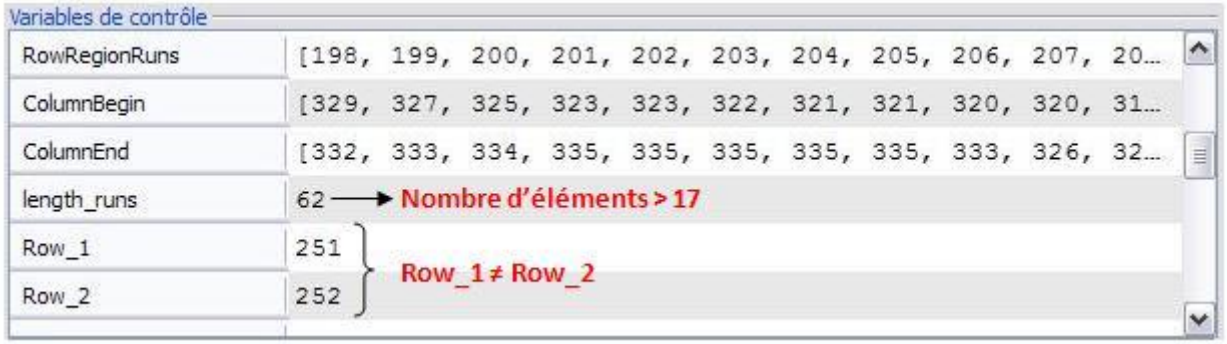


Figure 32: Aperçu de la fenêtre "variables de contrôle" 1

Comme il n'y a pas d'égalité, nous avons prévu une boucle « tant que » qui a pour but de chercher la première répétition de lignes en utilisant une variable d'index(j) initialisée par la valeur 7 avant la boucle et incrémentée à chaque fois que le pointeur d'instruction (IP) entre dans la boucle. Par la suite, cette variable d'index est utilisée pour affecter des éléments précédents du tableau aux variables Row_1 et Row_2.

Cette boucle s'exécutera tant que **(Row_1 est différente de Row_2) ET (la variable d'index est inférieure à 17)**. Nous voyons ici l'utilité de la limite haute qui est de 17 puisque nous ne voudrions pas affecter les éléments ne concernant plus la base du faisceau laser.

$$Row\_1 = RowRegionRuns[nombre\ d'éléments - j]$$

$$Row\_2 = RowRegionRuns[nombre\ d'éléments - j + 1]$$

Le pointeur d'instruction a quitté la boucle lorsque la condition d'égalité entre Row_1 et Row_2 a été validée.

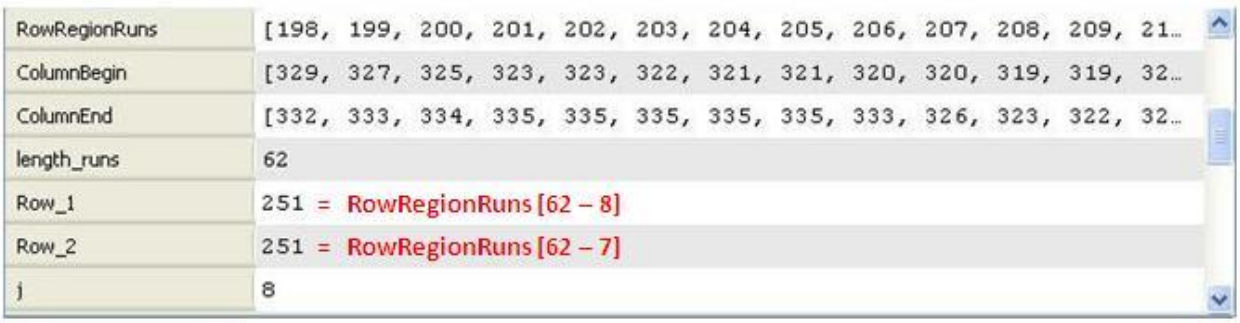


Figure 33: Aperçu de la fenêtre "variables de contrôle" 2

L'algorithme vient de détecter la première répétition de lignes mais cela ne signifie pas que c'est une paire de lignes.

IV. G. 1. 1. Problème de voisinage :

Nous avions utilisé une méthode qui permet de compléter un segment avec un élément structurant rectangulaire. Malgré l'avoir utilisée précédemment, il pourrait toujours y avoir des trous perturbant la détection de pair.

Figure 34: Problème de voisinage

L'image ci-dessus démontre le cas où l'opérateur a détecté, au niveau de la base, trois débuts de colonne et trois fins de colonne sur la même ligne. Cette situation donne donc lieu à la répétition de la même ligne 3 fois de suite dans le tableau « RowRegionRuns ».

RowRegionRuns Array	ColumnBegin Array	ColumnEnd Array
Row_i	$ColumnBegin_i$	$ColumnEnd_i$
Row_{i+1}	$ColumnBegin_{i+1}$	$ColumnEnd_{i+1}$
Row_{i+2}	$ColumnBegin_{i+2}$	$ColumnEnd_{i+2}$

Avec $Row_i = Row_{i+1} = Row_{i+2}$

Figure 35: Cas "Répétition de la même ligne 3 fois de suite"

Comme l'algorithme avait déjà trouvé une répétition de lignes, il devra par la suite examiner les voisins des variables Row_1 et Row_2 afin d'être sûr qu'il s'agit bien d'une paire. Pour faire ainsi, nous affectons les éléments voisins aux variables « one_before » et « one_after » pour les comparer avec Row_1 et Row_2.

$$one\_before = RowRegionRuns[nombre\ d'éléments - j - 1]$$

$$one\_after = RowRegionRuns[nombre\ d'éléments - j + 2]$$

IV. G. 1. 1. a. Cas du voisinage non-trouvé :

Si la condition **(Row_1 est différente de one_before) ET (Row_2 est différente de one_after)** a été validée, l'algorithme est alors sûr qu'il n'y a pas de voisinage et qu'il a trouvé la paire de lignes. Il peut désormais déterminer les coordonnées MinRow, MaxRow, Left_Column et Right_Column.

« MinRow » sera affectée par le premier élément du tableau « RowRegionRuns » car cet élément correspond au numéro de lignes du sommet.

« MaxRow » sera affectée par la variable Row_1 ou Row_2 car elles constituent une paire de lignes pour laquelle l'épaisseur du mastic d'étanchéité pourrait être calculée.

« Left_Column » sera affectée par l'élément du tableau « ColumnEnd » selon la logique expliquée précédemment :

$$Left\_Column = ColumnEnd[nombre\ d'éléments - j]$$

« Right_Column » sera affectée par l'élément du tableau « ColumnBegin » selon la logique expliquée précédemment :

$$Right\_Column = ColumnBegin[nombre\ d'éléments - j + 1]$$

IV. G. 1. 1. b. Cas du voisinage trouvé :

Dans ce cas, la condition **(Row_1 est différente de one_before) ET (Row_2 est différente de one_after)** n'a pas été validée. C'est- à -dire qu'il y a au moins un voisinage qui correspond à la répétition de la même ligne 3 fois de suite ou voire deux voisinages qui correspondent à la répétition de la même ligne 4 fois de suite dans le tableau « RowRegionRuns ».

Nous avons alors prévu une boucle « tant que » afin de trouver le cas où **Row_1 est égale à Row_2 ET one_before est différente de Row_1 et one_after est différente de Row_2**. Autrement dit, chercher la valeur de la variable d'index « j » en l'incrémentant à chaque fois tant **qu'il y a au moins 1 voisinage OU que Row_1 n'est plus égale à Row_2**.

Lorsque la valeur de la variable d'index « j » pour laquelle la condition « tant que » n'est plus valable a été trouvé, l'algorithme peut alors déterminer les coordonnées MinRow, MaxRow, Left_Column et Right_Column.

« MinRow » sera affectée par le premier élément du tableau « RowRegionRuns » car cet élément correspond au numéro de lignes du sommet.

« MaxRow » sera affectée par la variable Row_1 ou Row_2 car elles constituent une paire de lignes pour laquelle l'épaisseur du mastic d'étanchéité pourrait être calculée.

« Left_Column » sera affectée par l'élément du tableau « ColumnEnd » selon la logique expliquée précédemment :

$$Left\_Column = ColumnEnd[nombre\ d'éléments - j]$$

« Right_Column » sera affectée par l'élément du tableau « ColumnBegin » selon la logique expliquée précédemment :

$$Right\_Column = ColumnBegin[nombre\ d'éléments - j + 1]$$

IV. G. 2. Procédure pour l'égalité trouvée :

Dans un second temps nous considérons le cas où Row_1 est égale Row_2. Comme il y a une égalité, l'algorithme peut directement passer au contrôle de voisinage afin d'être sûr qu'il s'agit bien d'une paire.

Pour le faire nous affectons, comme précédemment, les éléments voisins aux variables « one_before » et « one_after » pour les comparer avec Row_1 et Row_2.

Les prochaines étapes seront exactement identiques aux méthodes adoptées pour les cas du voisinage non-trouvé (**IV. G. 1. 1. a.**) et cas du voisinage trouvé (**IV. G. 1. 1. b.**).

IV. G. 3. Préparation des outils d'affichage et calcul de la hauteur :

Après avoir déterminé les variables primordiales telles que « MinRow », « MaxRow », « Left_Column » et « Right_Column », l'algorithme peut alors passer aux calculs de la hauteur et de l'épaisseur du mastic d'étanchéité d'abord en pixels puis en millimètres.

Ces calculs seront affichés sur l'écran pour l'utilisateur. Afin d'augmenter la perception visuelle de l'utilisateur lors de la transition entre deux images successives, des lignes indicatrices qui délimitent la distance entre « MaxRow » et « MinRow », des croix pour indiquer « Left_Column » et « Right_Column » et aussi des flèches sont utilisées.

Les lignes indicatrices délimitant la distance entre « MaxRow » et « MinRow » ont aussi besoin des numéros de colonnes qui se rafraîchissent pour chaque mesure.

Hdevelop possède une méthode qui permet de réorganiser les éléments d'un tableau en ordre croissant et les mettre dans un nouveau tableau. Les éléments des tableaux « ColumnBegin » et « ColumnEnd » sont alors mis en ordre croissant dans deux autres tableaux, « SortedColumnBegin » et « SortedColumnEnd ».

En termes de colonne, les lignes indicatrices vont commencer par le premier élément du tableau « SortedColumnBegin » et finir par le dernier élément du tableau « SortedColumnEnd ». Deux nouvelles variables sont alors déterminées pour être utilisées lors du traçage de ces lignes.

IV. G. 3. 1. Calcul de la hauteur et de l'épaisseur :

Comme l'algorithme avait déjà déterminé les coordonnées « MinRow », « MaxRow », « Left_Column » et Right_Column », il suffira d'effectuer deux soustractions pour en tirer la hauteur et l'épaisseur du mastic en pixels :

$$height\_pixels = MaxRow - MinRow$$

$$height\_pixels = 251 - 198 = 53\ pixels$$

$$thickness\_pixels = Right\_Column - Left\_Column$$

$$thickness\_pixels = 353 - 308 = 45\ pixels$$

L'utilisateur s'intéresserait plutôt aux mesures en millimètres qu'en pixels. Pour cette raison, il faudrait convertir les données mesurées en millimètres.

IV. G. 3. 2. Approximation de la conversion :

Il existe des outils destinés à la vision industrielle pour calibrer la caméra. L'outil le plus connu pour effectuer la calibration s'appelle une plaque de calibration (calibration plate en anglais). Cette plaque possède des points noirs équidistants sur un fond blanc. La distance entre points étant connue,

l'utilisateur la positionne sur l'objet à mesurer et puis il utilise les méthodes appropriées du logiciel sur lequel il travaille afin de déterminer le rapport millimètres/pixels.

Voici la photo de la plaque de calibration :

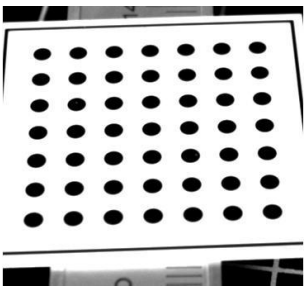
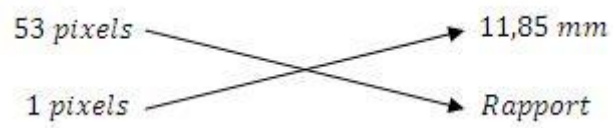


Figure 36: Calibration plate

Comme nous ne possédons pas un tel outil, nous avons dû utiliser une approximation afin de déterminer le rapport millimètres/pixels. La hauteur en pixels du mastic est connue. En utilisant un mètre, la hauteur approximative du mastic est mesurée.

$$height\_mm = 11{,}85\ mm$$

Le rapport s'en déduit par :



$$Rapport = \frac{11{,}85}{53} \sim 0{,}223\ mm/pixels$$

N.B : Ce rapport sera valable tant que la position de l'ensemble « caméra-laser » et la distance entre cet ensemble et le mastic restent inchangées.

Le rapport sera ensuite multiplié par la hauteur et l'épaisseur calculées en pixels afin d'afficher à l'utilisateur leurs valeurs approximatives en millimètres.

IV. G. 4. Conditions sur la hauteur et l'épaisseur :

Selon le type des miroiteries, la hauteur et l'épaisseur du mastic d'étanchéité seront différentes. Par exemple, pour la custode de la Mégane III, la hauteur du mastic devrait être entre 10 et 13 millimètres et l'épaisseur entre 7 et 10 millimètres.

Si les valeurs mesurées ne se trouvent pas dans ces intervalles de tolérances exemplaires, l'algorithme se chargerait de l'affichage de l'erreur à l'utilisateur et envoyer « NOT OK » vers le monde extérieur.

Pour l'instant nous avons fixé une valeur de tolérance pour la hauteur et une autre pour l'épaisseur. Cette approche sera concrétisée dans le chapitre suivant.

La dernière chose à noter dans cette partie est que la condition sur la hauteur du mastic est prioritaire par rapport à la condition sur l'épaisseur. C'est- à -dire si la hauteur mesurée est en dehors de sa tolérance, l'erreur sera affichée à l'écran et l'algorithme ne procéderait pas au traitement de l'épaisseur car selon le cahier des charges, il ne peut y avoir une épaisseur que s'il y a une hauteur.

Si la valeur de la hauteur reste dans sa tolérance, l'algorithme calculerait alors l'épaisseur en millimètres et vérifierait ensuite sa tolérance pour afficher une erreur ou sa valeur réelle à l'écran.

Voici les images des différents cas :

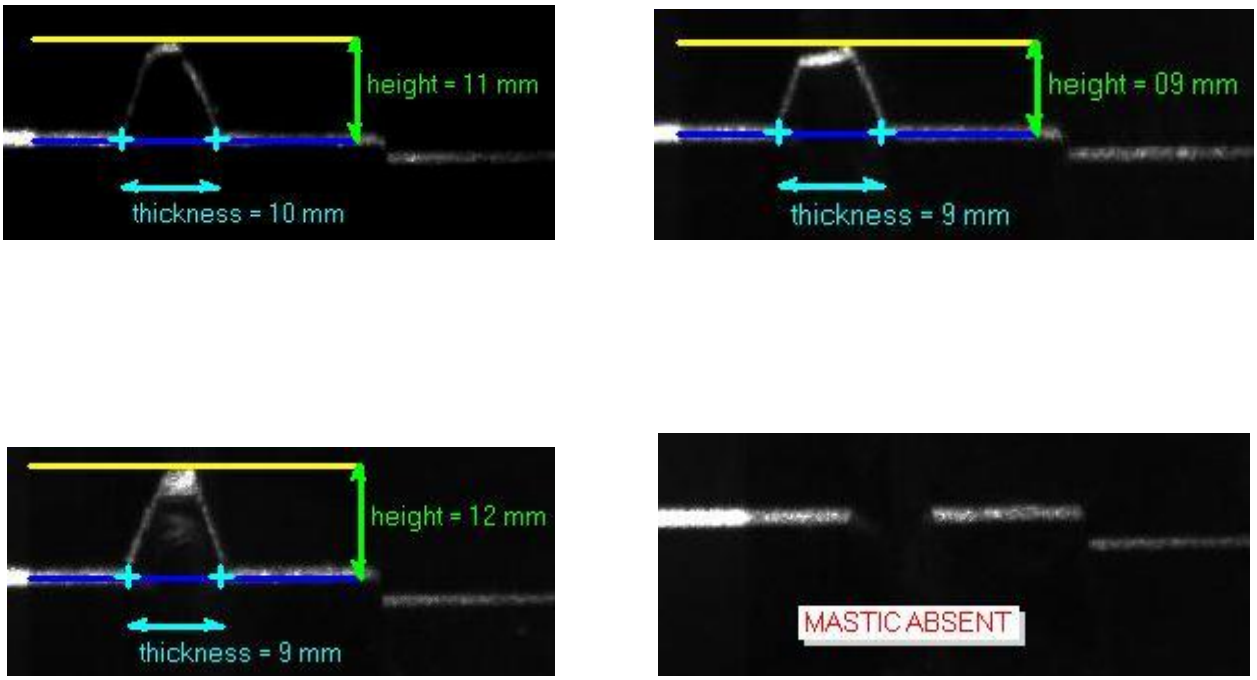


Figure 37: Mesures des différents cas probables

V. Structure du Système Complet

V. A. Principe de fonctionnement :

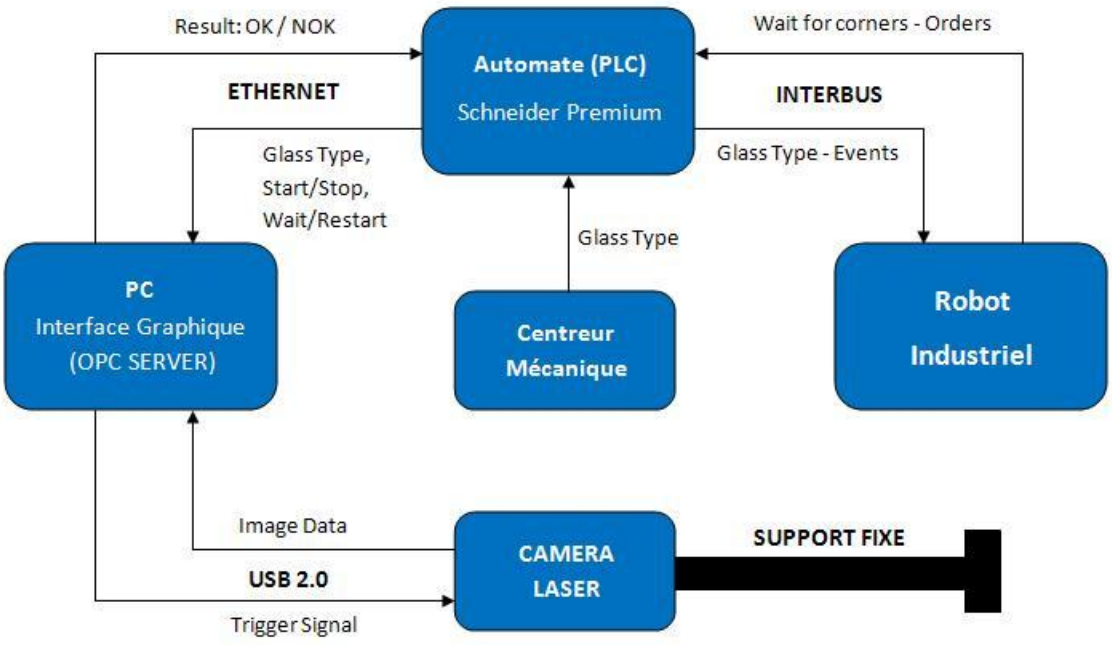


Figure 38: Principe de fonctionnement du système complet

V. A. 1. Identification du modèle de vitre :

Le centreur mécanique est un outil sur lequel les vitres d'automobile sont déposées par un petit robot à bras. Lorsqu'une vitre y est déposée, le capteur de présence sur le centreur envoie un signal numérique à l'automate afin de l'informer de sa présence.

L'automate autorise ensuite le centreur mécanique à fermer les bras de son support sur la vitre. Deux capteurs de déplacement potentiométriques, montés sur le support du centreur, envoient deux signaux analogiques à l'automate et ce dernier interprète les signaux pour identifier le modèle de la vitre déposée.

V. A. 2. Encollage du mastic d'étanchéité :

Après l'identification de son modèle, la vitre est alors prête pour l'encollage du mastic sur ses bords intérieurs. Puis, l'automate établit la communication avec un robot industriel pour que cette vitre soit prise par son préhenseur à ventouse.

Plusieurs interactions ayant lieu entre le robot et l'automate telles que les événements et les ordres, ce robot industriel se positionne avec la vitre accrochée à son préhenseur devant la buse de mastic.

N.B : La buse de mastic est commandée par un système industriel pour que le débit du mastic à encoller soit réglé pour chaque modèle de vitres. Evidemment, ces modèles sont envoyés au system industriel par l'automate.

Lorsque la procédure d'encollage du mastic d'étanchéité se termine, le robot sera alors prêt à effectuer le contrôle de qualité du mastic à l'aide de notre système de mesure « camera-laser ». Ce système de mesure sera monté sur un support fixe, juste au-dessus de la buse de mastic, pour que le

chemin à faire par le robot pour y emmener la vitre soit le plus court possible afin d'éviter le délai surabondant.

V. A. 3. Mesure de la hauteur et de l'épaisseur du mastic :

Quand le robot se positionne pour la mesure, il envoie un ordre à l'automate via la liaison interbus pour l'informer qu'il est prêt à commencer. L'automate à son tour envoie le type de la vitre au robot pour qu'il sache quelle trajectoire devrait-il exécuter.

L'automate envoie le type de la vitre également au PC via une variable de type Word (16 bits) afin que l'interface graphique puisse tirer de sa base de données les valeurs limites de la hauteur et de l'épaisseur du mastic pour la vitre concernée. Ces valeurs limitent seront utilisées comme références lors de la mesure pour les comparer aux valeurs mesurées. Par exemple, imaginons que pour la lunette arrière de la Mégane III, la hauteur devrait être entre 10 et 12 millimètres et l'épaisseur entre 8 et 9 millimètres. Si, lors de la mesure, les valeurs mesurées ne se trouvent pas dans ces intervalles, le PC enverrait alors « NOT OK » à l'automate via une variable de type booléen.

Après l'envoi du type de la vitre concernée au robot et au PC, l'automate va envoyer cette fois-ci un évènement au Robot pour l'autoriser à commencer. Il donne la même autorisation au PC via une autre variable de type booléen « Start/Stop ».

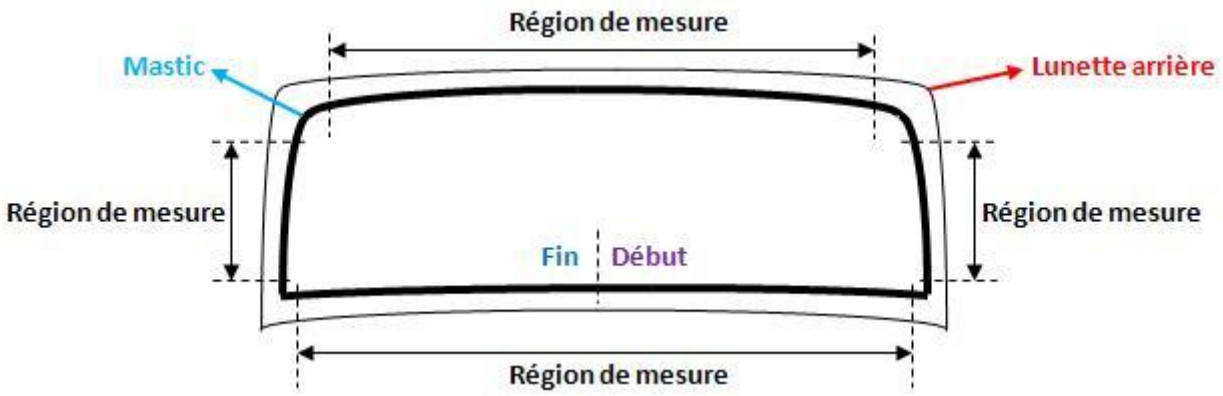


Figure 39: Illustration de la région de mesure

Le mouvement du robot et l'acquisition des données commencent alors simultanément. Notre système n'est prévu de mesurer que les bords du mastic d'étanchéité et non pas ses coins. Pour cette raison, il faudrait suspendre la mesure avant d'arriver aux coins et reprendre juste après. Les arrivées aux coins sont connues par le robot et communiquées à l'automate via des ordres. Selon ces ordres, l'automate suspend et puis reprend l'acquisition au côté du PC à l'aide d'une autre variable de type booléen « Wait/Restart ».

Lorsque la mesure se termine, l'automate envoie au PC la fin de l'acquisition via la variable « Start/Stop ». S'il n'y a pas eu de problème le PC envoie « OK » à l'automate. Si la réponse est « OK », la vitre est alors prête à être déposée sur un support à ventouse fixe. Par la suite, un autre robot industriel se charge de l'installer sur le cadre réservé de l'automobile. S'il y a eu un problème, la vitre sera déposée ailleurs sachant le mastic encollé n'est pas conforme.

V. B. Interface Graphique :

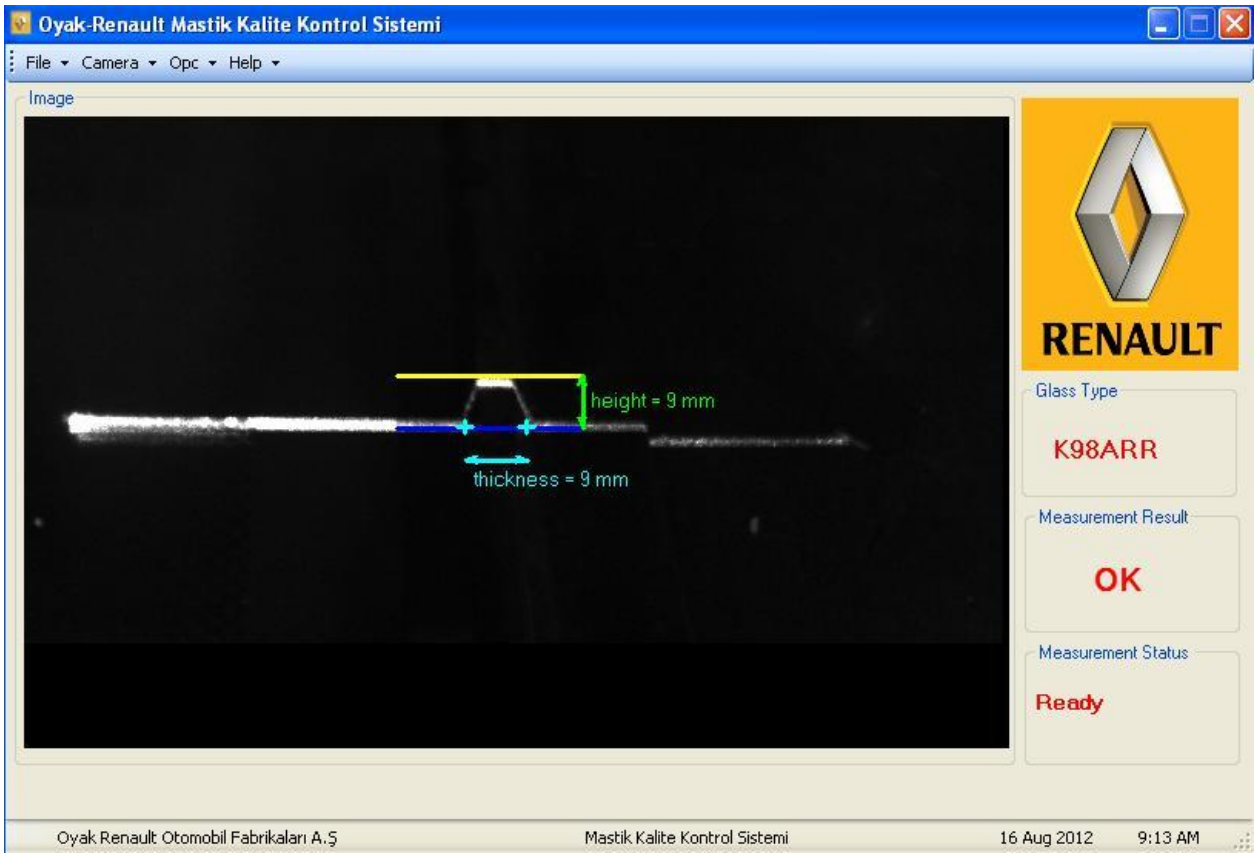


Figure 40: Interface graphique finale

L'interface graphique utilisateur a été programmée en C#. Elle a trois rôles primordiaux dans la chaîne de mesure :

- Assurer la communication bidirectionnelle avec l'automate,
- Extraire les informations concernant la hauteur et l'épaisseur du mastic d'étanchéité,
- Décider la conformité du mastic selon des références prédéfinies.

V. B. 1. Communication :

L'interface graphique communique avec l'automate en utilisant la norme « OPC Server » (Open Process Control Server). Cette norme a été conçue pour connecter un PC à des périphériques tels que des automates ou des sources de données.

Dans notre application, le PC est le serveur et l'automate est le client. Selon le type de la carte réseau disponible sur le PC, il pourrait y avoir plusieurs clients connectés à la fois communiquant avec le PC. Le transfert de données s'effectuant sur le câble Ethernet, l'interface graphique utilise une bibliothèque de liens dynamiques « OPC Automation.dll » pour gérer l'initialisation de la connexion et le transfert de données.

Notre interface graphique possède également une fenêtre de configuration des variables, accessible uniquement en mode déconnecté, afin de permettre à l'utilisateur de définir les identifiants des variables de l'automate sur lesquelles les données sont envoyées et reçues.

V. B. 2. Traitement d'image :

L'algorithme du système de mesure était expliqué dans le chapitre précédent. Comme le code était écrit en langage « HDevelop », il était impératif de le convertir en C# pour l'adapter à notre interface graphique. Pour ce faire, une autre bibliothèque de liens dynamiques (HalconDotNet.dll) a été intégrée au code de l'interface car sans celle-ci, les opérateurs de Halcon ne seraient pas utilisables dans le milieu C#.

Nous avons intégré le code concernant le traitement d'image dans une fonction sans paramètres d'entrée ni de retour qui est appelée sous un « thread ». En effet, le « thread » a pour but de parcourir le code en continue tant que la variable « Start/Stop » est égale à 1 et la variable « Wait/Restart » à 0. Lorsqu'une de ces deux variables qui contiennent les informations envoyées par l'automate subit un changement, le « thread » se détruit et les étiquettes (labels) affichent à l'utilisateur les informations correspondantes.

Start/Stop (Info Reçue)	Wait/Restart (Info Reçue)	Etat en cours	Résultat : OK/NOK (Info envoyée)
1	0	Acquisition	-
1	1	Acquisition suspendue	-
0	0	Fin de l'acquisition	1(OK) ou 0 (NOK)

Figure 41: Réponse du système de vision industrielle selon différents états des bits

V. B. 3. Décision de la conformité :

Les limites admissibles de la hauteur et de l'épaisseur du mastic sont attribuées à chaque modèle de vitres et sauvegardées dans une base de données par l'utilisateur. Dès que le modèle de la vitre est reçu par le PC, le pointeur d'instruction va chercher ces valeurs limites avant que l'acquisition ne commence. Ces valeurs sont ensuite placées dans les conditions s'occupant de la décision de conformité du mastic.

Il est à noter que ces valeurs limites peuvent être modifiées et sauvegardées à tout moment (sauf pendant l'acquisition) par l'utilisateur à l'aide d'une fenêtre auxiliaire.

VI. Expérience acquise et Difficultés rencontrées

Pendant la durée de mon stage, j'ai pu acquérir plusieurs expériences et compétences considérables. Parmi ces compétences la plus importante, à mon avis, est d'avoir amélioré mon côté autonome en ayant fait des recherches sur un tout nouvel aspect pour moi et cherché des solutions pour les points critiques de mon application.

Lors du développement de l'algorithme de « Traitement d'image », j'ai eu des difficultés au départ. D'une part, je ne possède pas les notions de base dans ce domaine et d'autre part mes collègues de travail ne pouvaient pas m'aider car ils sont plutôt spécialisés dans la programmation des automates et des robots.

Par ailleurs, lors du développement de l'interface graphique, de l'intégration du protocole de communication « OPC Server » et des tests effectués, mes collègues, surtout mon maitre de stage, m'ont aidé considérablement. En effet, grâce à leur aide, j'ai pu rattraper le temps que j'avais perdu pendant le développement de la partie « Traitement d'image ».

En outre, J'ai eu l'occasion d'assister à des formations industrielles, telles que l'Automatisme Général, Robot ABB IRC5 et Schneider Premium PLC, à l'Institut d'OYAK – Renault pour pouvoir mieux comprendre le fonctionnement de différents outils indispensables dans un environnement de production.

VII. Conclusion

Ce stage était une grande expérience qui m'a permis de faire un premier pas dans le milieu industriel. Cette expérience enrichissante au niveau humain, ainsi qu'au niveau culture industrielle m'a donc permis d'avoir un véritable aperçu du métier d'ingénieur.

Ce stage m'a aussi permis de mieux voir le fonctionnement d'une organisation complexe dans une entreprise industrielle de production.

J'ai eu donc l'opportunité de travailler sur un tout nouveau projet qui a pour but de contrôler la conformité du mastic d'étanchéité encollé dans l'installation automatique de montage des miroiteries. En effet, ce projet m'a permis d'orienter mon savoir-faire en informatique industrielle vers la vision industrielle, une nouvelle technique de contrôle qui se trouvera une bonne place dans les milieux de production en série.

J'ai pu aussi remarquer pendant la période de stage que les attentes et les objectifs de chaque division au sein de l'entreprise peuvent varier et s'orienter dans de différentes directions. Mais l'essentiel est de ne pas s'éloigner du but principal car il est impératif que chaque entité tourne dans le bon sens pour le bon avancement du projet.

ANNEXE 1: OYAK – Renault en bref:

Situation géographique	à 150 km au sud d'Istanbul et 15 km de la mer de Marmara
Superficie	534 530 m²
- dont surface construite	*301 385 m²*
Date de création	1969
Date de première sortie de chaîne	14 Mai 1971 **(Renault12)**
Secteurs d'activités	Usine carrosserie-montage Usine mécanique-châssis ILN – International Logistic Network Platform
Nombre d'employés (en Juin 2011)	6012 personnes à Bursa dont 50 personnes à Istanbul
Capacité annuelle	360,000 véhicules & 450,000 moteurs
Cadence journalière moyenne	1150 véhicules / jour
Production totale depuis la création	4.000.000 véhicules (au 28/03/2012) 3.000.000 moteurs (au 06/10/2011)
Production en 2011	330.994 véhicules 291.797 moteurs
Moteurs produits	Moteurs K : K9K, K4M, K7J, K4J
Véhicules produits	Clio Symbol, Clio III, Clio IV, Mégane III, Fluence, Fluence Z.E (Zéro Emission)

ANNEXE 2: Photos de l'installation automatique de montage des miroiteries

Figure : Photo du centreur mécanique

Figure: Photo du robot industriel KUKA, du centreur mécanique, du préhenseur à ventouse et de la lunette arrière accrochée au préhenseur

Figure: Positionnement de la vitre devant la buse de mastic pour l'encollage de celui-ci

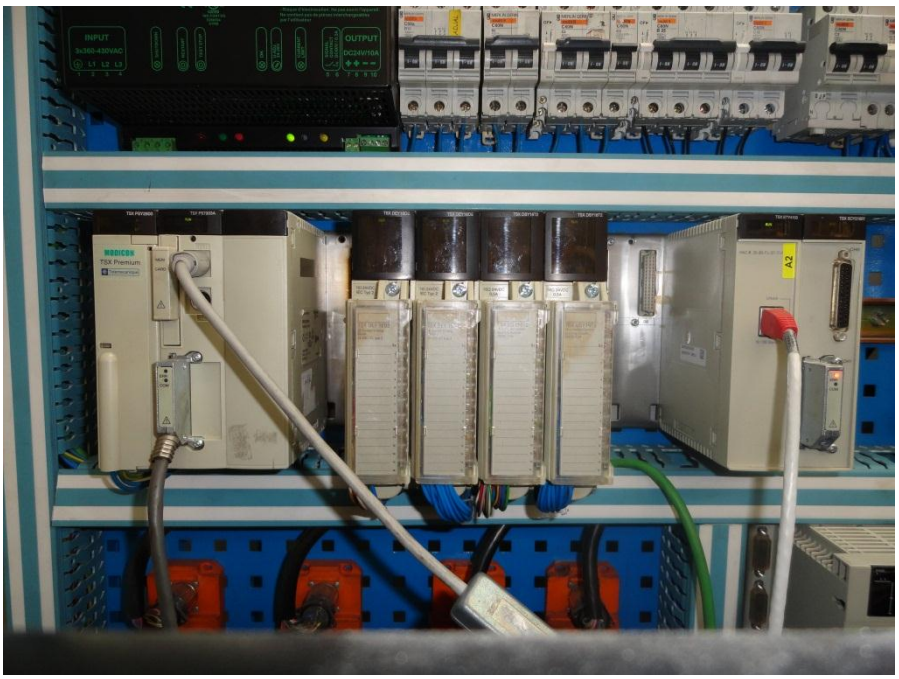


Figure: Photo de l'automate Schneider Premium qui gère l'ensemble des actions dans l'installation automatique de montage des miroiteries

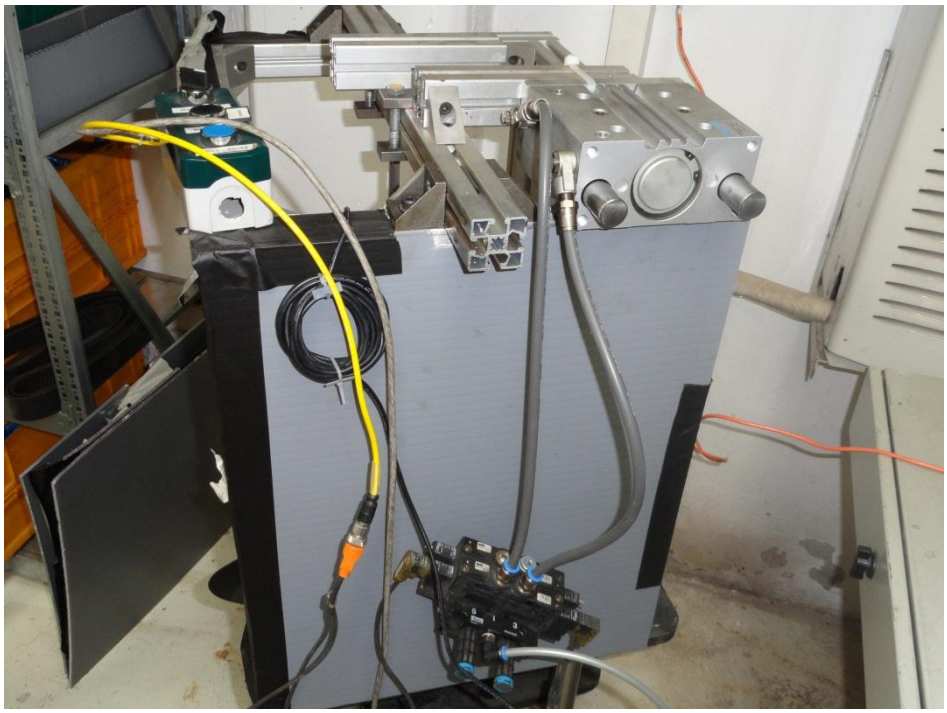


Figure: Photo du prototype en forme de cube construit pour effectuer les premieres mesures. Le mouvement de l'ensemble "caméra – laser" est assuré par un piston excité par une vanne électropneumatique

ANNEXE 3: Présentation de l'environnement Halcon / HDevelop

HDevelop est une boîte à outils pour construire des applications de vision industrielle. Il facilite le prototypage rapide en offrant un environnement de programmation hautement interactif pour développer et tester des applications de vision industrielle. Basé sur la bibliothèque HALCON, il s'agit d'un paquet de vision industrielle polyvalent et adapté pour le développement de produits, la recherche et l'éducation.

Il y a quatre façons basiques pour développer des applications d'analyse d'images en utilisant HDevelop:

- *Prototypage rapide dans l'environnement interactif HDevelop.*
  Il est possible d'utiliser HDevelop pour trouver des opérateurs ou des paramètres optimaux afin de résoudre la tâche d'analyse d'image, et ensuite construire l'application en utilisant les différents langages de programmation, par exemple, C, C + +, C #, Visual Basic. NET ou Delphi.

- *Développement d'une application qui s'exécute dans HDevelop.*
  Utilisant HDevelop, il est également possible de développer une application d'analyse d'image complète et l'exécuter dans l'environnement HDevelop. Par ailleurs, les exemples de programmes fournis avec HDevelop peuvent être utilisés comme blocs de construction pour les applications dont le programmeur souhaite développer.

- *Exécution des programmes ou des procédures HDevelop utilisant HDevEngine.*
  Si le programmeur le souhaite, il peut directement exécuter les programmes ou les procédures développés en HDevelop à partir d'une application écrite en C ++ ou tout autre langage qui peut intégrer des objets propres au .NET ou COM en utilisant HDevEngine.

- *Exportation d'une application comme C, C + +, Visual Basic, Visual Basic. NET, ou un code source C #.*
  Enfin, le programmeur peut aussi exporter une application développée en HDevelop comme un code source en C, C + +, Visual Basic, Visual Basic. NET ou C #. Ce programme peut alors être compilé et lié avec la bibliothèque HALCON de telle sorte qu'il fonctionne comme une application stand-alone (console).

HDevelop supporte activement le développement d'une application en nombreuses façons :

- Grâce à l'interface graphique utilisateur de HDevelop, les opérateurs et les objets iconiques peuvent être directement sélectionnés, analysés et échangés au sein d'un environnement unique.

- HDevelop suggère des opérateurs pour des tâches spécifiques. En outre, une liste d'opérateurs thématiquement structurée aide le programmeur à trouver rapidement un opérateur approprié.

- Une aide en ligne intégrée contient des informations sur chaque opérateur HALCON, telle qu'une description détaillée de la fonctionnalité, des opérateurs successeurs et prédécesseurs typiques, la complexité de l'opérateur, la gestion des erreurs, et des exemples d'application. Cependant, l'aide en ligne propose également un moteur de recherche qui permet de rechercher dans la documentation complète de HALCON.

- HDevelop comprend un interpréteur de programme avec les fonctions d'« edit » et de « debug ». Il prend en charge les fonctionnalités de programmation standard, telles que les procédures, les boucles ou les instructions conditionnelles. En outre, les paramètres peuvent être modifiés par le programmeur même si le programme est en cours d'exécution.

- HDevelop affiche immédiatement les résultats des opérations. Il est possible d'utiliser différents opérateurs et / ou des paramètres, et de voir immédiatement l'effet sur l'écran. Par ailleurs, l'utilisateur peut prévisualiser les résultats d'un opérateur sans changer le programme.

- Plusieurs outils graphiques permettent d'examiner les données iconiques et de contrôle en ligne. Par exemple, l'utilisateur peut extraire la forme et les caractéristiques des niveaux de gris en cliquant simplement sur les objets dans la fenêtre graphique ou alors inspecter l'histogramme d'une image interactivement et appliquer la segmentation en temps réel pour sélectionner les paramètres.

- Les assistants graphiques « Built-in » fournissent des interfaces interactives pour des tâches de vision industrielle plus complexes. Ces assistants peuvent également générer des codes en HDevelop dans le programme actuel.

ANNEXE 4: Algorithme en lignes de code « HDevelop »:

```
initCamera(AcqHandle, WindowHandle)
while(1)
grab_image (Image, AcqHandle)
get_image_size(Image, Width, Height)
*draw_rectangle1(3600, Row1, Column1, Row2, Column2)
gen_rectangle1(Rectangle, 84.5, 354.5, 174.5, 454.5)
reduce_domain(Image, Rectangle, ImageReduced)
dev_update_off()
mean_image(ImageReduced, ImageMean, 7, 7)
emphasize(ImageMean,EmphasizedImage,35,35,2.0)
threshold (EmphasizedImage, Regions, 65, 255)
connection(Regions, ConnectedRegions)
closing_rectangle1(ConnectedRegions, RegionClosing1, 200, 25)
select_shape(RegionClosing1, SelectedRegions, 'area', 'and', 20,
1000)
union1(SelectedRegions, RegionUnion)
get_region_runs (RegionClosing1, RowRegionRuns, ColumnBegin,
ColumnEnd)
dev_clear_window()

*Détection de la paire de lignes(gauche et droite)
dev_display(Image)

length_runs:=|RowRegionRuns|

if(length_runs>17)

    Row_1:=RowRegionRuns[length_runs-7]
    Row_2:=RowRegionRuns[length_runs-6]

    *Conformité des pairs de coordonnées
    if(Row_2 # Row_1)
        j:=7
        while((Row_2 # Row_1)and j&lt;17)
              j:=j+1
              Row_1:= RowRegionRuns[length_runs-j]
              Row_2:=RowRegionRuns[length_runs-j+1]
        endwhile

        *Attribution des éléments voisins
        One_before:=RowRegionRuns[length_runs-j-1]
        One_after:= RowRegionRuns[length_runs-j+2]

        *S'il n'y a pas de voisinage, tout est OK
        if((Row_1 # One_before) and (Row_2 # One_after))
             MaxRow := Row_1
             MinRow := RowRegionRuns[0]
             Left_Column:= ColumnEnd[length_runs-j]
             Right_Column:= ColumnBegin[length_runs-j+1]

        *SINON
        else
```

```
            *Chercher tant qu'il y a au moins 1 voisinage
            * OU qu'il n'y a plus de parité
            while(((Row_1 = One_before)or(Row_2 = One_after)) or
(Row_1 # Row_2))
                  j:=j+1
                  Row_1:= RowRegionRuns[length_runs-j]
                  Row_2:=RowRegionRuns[length_runs-j+1]
                  One_before:=RowRegionRuns[length_runs-j-1]
                  One_after:= RowRegionRuns[length_runs-j+2]
            endwhile

            MaxRow := Row_1
            MinRow := RowRegionRuns[0]
            Left_Column:= ColumnEnd[length_runs-j]
            Right_Column:= ColumnBegin[length_runs-j+1]

        endif
        *Parité trouvée
      else
        j:=7

        *Attribution des éléments voisins
        One_before:=RowRegionRuns[length_runs-j-1]
        One_after:= RowRegionRuns[length_runs-j+2]

        *S'il n'y a pas de voisinage, tout est OK
        if((Row_1 # One_before) and (Row_2 # One_after))
            MaxRow := Row_1
            MinRow := RowRegionRuns[0]
            Left_Column:= ColumnEnd[length_runs-j]
            Right_Column:= ColumnBegin[length_runs-j+1]
        *SINON
        else
            *Chercher tant qu'il y a au moins 1 voisinage
            * OU qu'il n'y a plus de parité
            while(((Row_1 = One_before)or(Row_2 = One_after)) or
(Row_1 # Row_2))
                  j:=j+1
                  Row_1:= RowRegionRuns[length_runs-j]
                  Row_2:=RowRegionRuns[length_runs-j+1]
                  One_before:=RowRegionRuns[length_runs-j-1]
                  One_after:= RowRegionRuns[length_runs-j+2]
            endwhile

            MaxRow := Row_1
            MinRow := RowRegionRuns[0]
            Left_Column:= ColumnEnd[length_runs-j]
            Right_Column:= ColumnBegin[length_runs-j+1]
        endif
  endif

        tuple_sort(ColumnBegin, SortedColumnBegin)
        tuple_sort(ColumnEnd, SortedColumnEnd)

        MinColumn := SortedColumnBegin[0]
        MaxColumn := SortedColumnEnd[length_runs-1]
```

```
        height_pixel:= MaxRow - MinRow
        height_mm:= (height_pixel*19/85)

        if(height_mm < 6)
            disp_message(3600, 'ABSENT BEAD OR MEASUREMENT ERROR',
'image', Height-120, Width-450, 'red', 'true')
        else
            *******Tracé de la ligne basse**********

            dev_set_line_width(3)
            dev_set_color('blue')
            disp_line(3600, MaxRow, MinColumn - 20, MaxRow,
MaxColumn + 20)

            *******Tracé de la ligne haute**********

            dev_set_line_width(3)
            dev_set_color('yellow')
            disp_line(3600, MinRow, MinColumn - 20, MinRow,
MaxColumn + 20)

            *******Tracé des flèches pour la hauteur*********

            dev_set_line_width(3)
            dev_set_color('green')
            disp_arrow(3600,MaxRow, MaxColumn + 20, MinRow,
MaxColumn + 20, 1)
            disp_arrow(3600,MinRow, MaxColumn + 20, MaxRow,
MaxColumn + 20, 1)
            disp_message(3600, 'height = '+ (height_mm)$ '.2'+' mm',
'image', (MaxRow + MinRow)/2 - 10, MaxColumn + 25, 'green', 'false')

            *******Tracés des flèches pour l'épaisseur*********

            thickness_pixel := Right_Column - Left_Column
            thickness_mm := (thickness_pixel*19/85)

            if(thickness_mm < 4)
                disp_message(3600, 'THICKNESS ERROR, CHECK ROI !!!',
'image', Height-100, Width-400, 'red', 'true')
                disp_cross(3600, Row_1, Left_Column, 10, 0)
                disp_cross(3600, Row_2, Right_Column, 10, 0)

            else
                dev_set_color('cyan')
                disp_cross(3600, Row_1, Left_Column, 10, 0)
                disp_cross(3600, Row_2, Right_Column, 10, 0)
                dev_set_line_width(3)
                dev_set_color('cyan')
                disp_arrow(3600,MaxRow + 25, Left_Column, MaxRow +
25, Right_Column, 1)
                disp_arrow(3600,MaxRow + 25, Right_Column, MaxRow +
25, Left_Column, 1)
                disp_message(3600, 'thickness = '+ thickness_mm $
'.1'+' mm', 'image', MaxRow + 30, Left_Column + 5, 'cyan', 'false')
```

```
            endif
        endif
    else
        disp_message(3600, 'TUPLE LENGTH ERROR', 'image', Height-
100, Width-400, 'red', 'true')
    endif
endwhile
```

ANNEXE 5: Références bibliographiques

- Intranet du groupe Renault (partie « Présentation de l'entreprise »)
- Machine Vision Algorithmes and Applications by Carsten Steger – Markus Ulrich – Christian Wiedemann (livre)
- Documentations techniques fournies par le logiciel HDevelop