

VIN- LAB05 : Calibration et correction de la distorsion des caméras

Auteur : Guillaume Bangala

Version : 31.10.2018



Objectifs

Les objectifs de ce travail de laboratoire sont :

- se familiariser avec les principes de programmation de base du matériel KEYENCE,
- calibrer un système optique afin de corriger la distorsion de la caméra.

Durée :

- labo : 4 x 45 minutes

Table des matières

1	Partie théorique : Programmation sur KEYENCE - principes de base	2
1.1	Les variables	2
1.1.1	Catégories et types de variables	2
1.1.2	Les noms de variables	2
1.1.3	Déclarer une variable	3
1.1.4	Accéder à/allouer une variable	4
1.1.5	Astuce – l'outil d'auto-complétion	4
1.2	Les fonctions	5
1.2.1	Paramétrer le bloc	5
1.2.2	Accéder aux résultats du bloc	6
1.3	L'utilisation des images par le contrôleur XG	7
1.3.1	Images de simulation	8
1.3.2	Images de référence	8
2	Partie pratique – création d'un programme KEYENCE	11
2.1	Résultats observés	12

1 Partie théorique : Programmation sur KEYENCE - principes de base

Comme sur beaucoup de langage de programmation, la programmation sur KEYENCE se fait au travers :

- **de variables** qui permettent de stocker/restituer des informations,
- **de fonctions** qui permettent de faire toutes sortes d'opérations.

1.1 Les variables

1.1.1 Catégories et types de variables

Les variables sont séparées en plusieurs catégories :

1. Les variables **Locales** qui appartiennent au programme en cours : elles sont donc accessibles uniquement dans le programme en cours,
2. Les variables **Globales** qui appartiennent au contrôleur : elles sont donc accessibles à tous les programmes qui tournent sur le contrôleur ; on y stockera exclusivement les variables invariables propres au système et qui ne change pas d'un programme à l'autre, comme les résultats de la balance des blancs, par exemple,
3. Les variables **Image** qui appartiennent aussi au programme en cours : elles sont stockées à part des 'variables Locales' car de par leur taille, elles sont traitées différemment, mais elles se comportent de la même manière,
4. Les variables **Temporaires** qui appartiennent à un bloc de calcul « Calculation » qui sont instanciées au début de l'exécution du bloc et détruite à la fin de ce dernier. Elles ne peuvent donc pas être propagées d'un bloc à l'autre et ne servent donc que de variables tampons utiles dans les processus calculatoires pour stocker des informations temporaires,
5. Les variables **Système** qui appartiennent au contrôleur et qui permettent de manipuler le comportement direct de l'automate en activant le Trigger, ou en manipulant des flags BUSY/IDLE du système.

Le langage de programmation KEYENCE étant typé, les variables Locales et Globales peuvent prendre différents types :

- **Numerical** : une simple valeur numérique,
- **Position** : une double valeur numérique des coordonnées d'un point (X et Y),
- **Line** : une double valeur numérique θ (T) et ρ (RH) correspondant à la représentation selon Hough d'une ligne,
- **Circle** : une triple valeur numérique coordonnées du centre (CX et CY) et le rayon (CR),
- **3D Pos** : une triple valeur numérique des coordonnées d'un point (X, Y et Z) en 3D,
- **Plane** : une triple valeur numérique des paramètres définissant un plan en 3D (PPA, PPB et PPC).

Chacun de ces types peut être **simple**, ou sous forme de **tableau (array)**.

1.1.2 Les noms de variables

Sur les logiciels XG-X, chaque variable est désignée par un préfixe et un nom :

'préfixe"nomDeVariable'

Pour le préfixe, chaque catégorie de variable se différenciera dans le code par un préfixe spécifique :

- **'#'** désigne les variables locales,
- **'\$'** désigne les variables globales,
- **'&'** désigne les variables images,
- **'@'** désigne les variables temporaires,
- **'!'** désigne les résultats des blocs (expliqué au point [1.2.2](#)),
- **'%'** désigne les variables système.

Pour le nom de la variable, rien à signaler, si ce n'est que :

- la casse de caractère est respectée,
- l'usage des '_' dans les noms est prohibé.

Ainsi, pour créer une variable locale appelée "maVariable", on écrira simplement "#maVariable".

1.1.3 Déclarer une variable

Pour déclarer une nouvelle variable, il suffit de cliquer sur l'icône « Variables » (illustré sur l'image 1) et une fenêtre s'ouvre avec 3 onglets distincts pour chacune des catégories de variables présentées plus haut. Cette fenêtre est accessible n'importe quand même si d'autres fenêtres sont ouvertes.

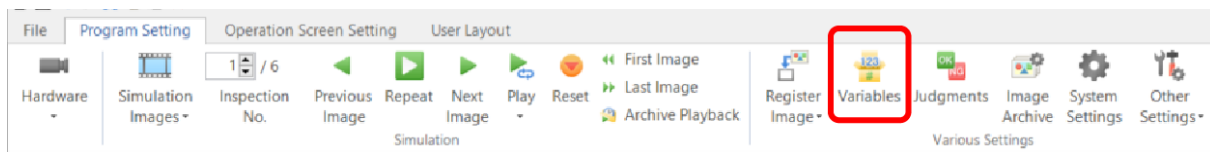


FIGURE 1 – Localisation de l'icône Variables

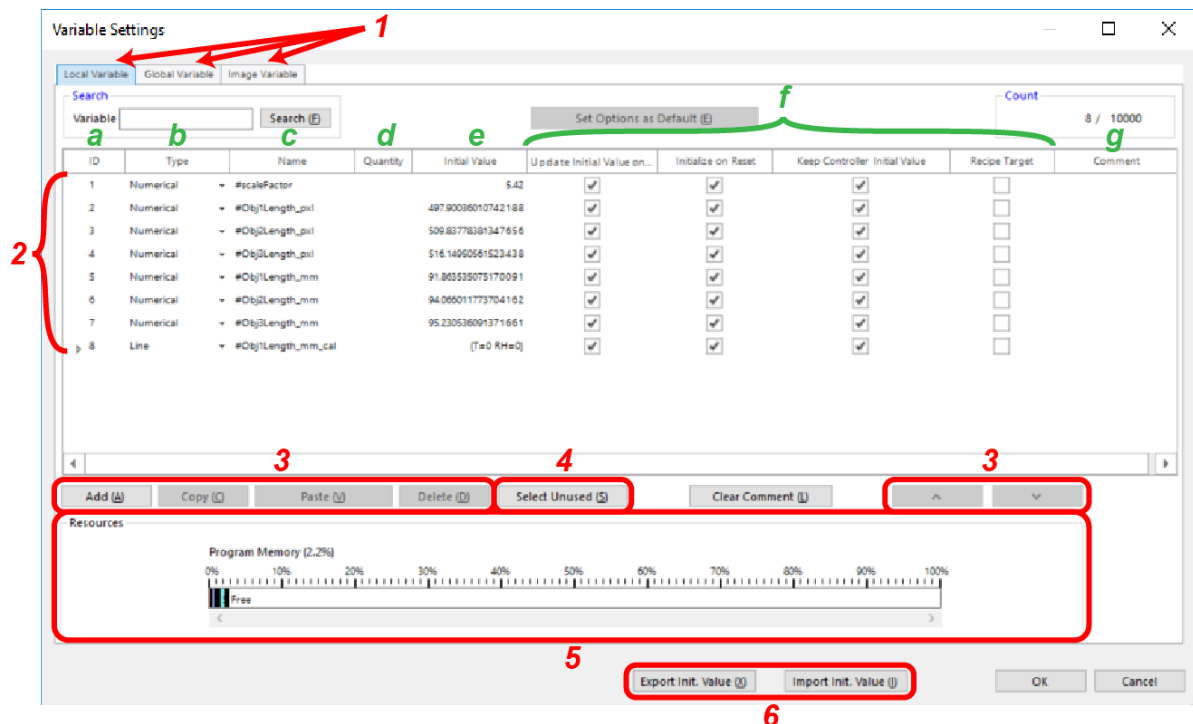


FIGURE 2 – Fenêtre de gestion des variables

Cette fenêtre présente ¹ :

1. Un onglet par type de variables,
2. Pour chaque catégorie de variable, la liste de toutes les variables de cette catégorie, avec pour chaque variable :
 - (a) son ID,
 - (b) son type,
 - (c) son nom,

1. Les numéros de la liste ci-dessous correspondent aux indices sur l'image 2

- (d) la quantité d'éléments contenus dans cette variable (quand la variable est un tableau),
 - (e) sa valeur initiale,
 - (f) divers flags,
 - (g) un commentaire pour la décrire.
3. des boutons pour ajouter/copier/coller/supprimer/réorganiser l'ordre des variables,
 4. un bouton pour trouver toutes les variables inutilisées dans le code,
 5. un affichage de l'utilisation de la mémoire que ces variables occupent,
 6. des boutons pour importer/exporter les valeurs initiales des différentes variables du programme.

Une fois que cette fenêtre est ouverte, il faut :

1. cliquer sur le bouton ajouter,
2. entrer un nom de variable (si aucun préfixe n'est spécifié, le software le prefixera automatiquement le nom en fonction de la catégorie choisie),
3. entrer une valeur initiale.

Attention 1 : veuillez à toujours déclarer vos variables en local!

Attention 2 : Toutes les variables sont créées à l'appel du programme et ne sont détruites que lorsque le programme est quitté (suite à une extinction du système ou un changement de programme, par exemple). Cela qui signifie que les variables sont conservées (avec leur valeur) à la fin de l'exécution du programme suite à un Trigger, et lancer un nouveau Trigger ne réinitialisera pas les variables. De plus, par défaut, le flag « overwrite initial value on save » des variables est activé, ce qui permet au programme d'enregistrer les valeurs actuelles des variables comme valeurs initiales futures. Pensez à désactiver cette fonction, si ce n'est pas ce que vous désirez!

1.1.4 Accéder à/allouer une variable

Une fois que la variable est déclarée dans la fenêtre « Variables », elle est créée et instanciée dans tout le programme et est donc appellable dans tout le code (comme dans des blocs de calcul "Calculation", par exemple, ou encore comme paramètre de blocs fonctionnels *illustré sur le bas de l'image 5*).

Pour appeler la variable, il faut simplement écrire son nom comme spécifié dans le paragraphe 1.1.1 et le tour est joué. Ensuite, d'une manière générale, l'accès au différents champs d'une structure ou éléments d'un tableau se fait **comme dans le langage C** :

- Pour une **structure**, l'accès à chaque champ se fait avec un '.'. Par exemple, pour accéder à la coordonnée X d'une variable globale de position appelée "maPosition", on écrira "\$maPosition.X",
- Pour un **tableau**, l'accès à chaque élément du tableau se fait avec des crochets, les indices vont de 0 à $taille_{tableau}-1$. Par exemple, pour accéder à la 3ème case du tableau local #positionsDesPoints, on écrira "#positionsDesPoints[2]".

1.1.5 Astuce – l'outil d'auto-complétion

XG-X VisionEditor possède un système d'auto-complétion efficace qui propose automatiquement toutes les variables disponibles. Par exemple, une variable globale position appelée "\$firstPosition" est déclarée, en écrivant '\$', VisionEditor proposera toutes les variables globales, dont celle-ci, puis une fois que "firstPosition" est sélectionné, si un '.' est ajouté, VisionEditor proposera automatiquement tous les champs disponibles.

1.2 Les fonctions

Les fonctions dans le contrôleurs XG sont toutes encapsulées dans des blocs fonctionnels. Le matériel KEYENCE offre des fonctions performantes de haut niveau, facilement et grandement paramétrables qui couvrent pratiquement tous les besoins liés à la vision industrielle. Pour ajouter une fonction, il suffit donc d'ajouter un bloc de plus à votre programme, de régler ses paramètres, et rien de plus.

1.2.1 Paramétrer le bloc

Lorsqu'une fonction est rajoutée au code, la fenêtre de paramètre s'ouvre automatiquement et donne accès à tous les paramètres de la fonction répartis sur plusieurs onglets. Le premier onglet est toujours l'onglet "General" qui regroupe des informations descriptives sur le bloc, telles que :

- son type de fonction qu'il exécute,
- son ID,
- son nom²,
- son flag d'exécution,
- un espace commentaire.

Certains blocs possèdent d'autres informations sur cet onglet, comme la variable image d'entrée, par exemple, mais tous ces paramètres sont ajustables sur les onglets suivants.

Comme décrit plus haut, les différents onglets permettent de paramétrer la fonction. Il existe plusieurs cas de figure pour paramétrer un élément :

- avec des valeurs **fixes** avec un seul champ à remplir,
- avec des valeurs **variables** avec 2 champs à remplir et une case à cocher.

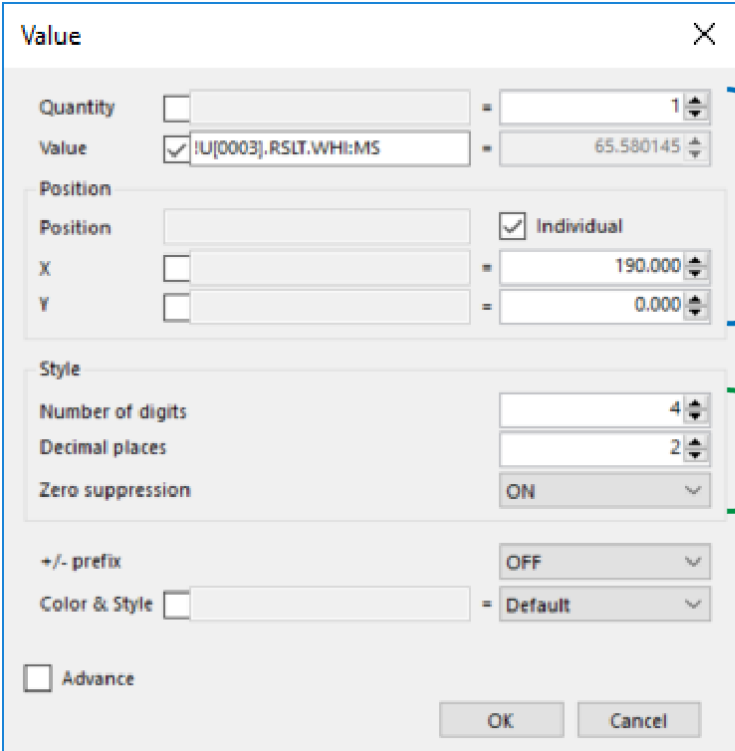


FIGURE 3 – Types de paramètres de blocs

Certains paramètres possèdent le champ « current value » : lors du paramétrage du bloc, il est possible de Trigger/Run le programme pour voir son comportement et afficher la valeur de ce paramètre, comme pour ajuster les limites admissibles, par exemple.

2. Prenez l'habitude de nommer vos blocs de manière précise et intelligente et de brièvement commenter le comportement attendu de vos blocs.



FIGURE 4 – *Champ 'Current value'*

1.2.2 Accéder aux résultats du bloc

Lors de leur exécution, chaque bloc génère une série de résultats qui décrivent les résultats intermédiaires et finaux de son exécution. Ils se rangent dans plusieurs catégories :

- les résultats de Jugement qui correspondent à un OK/NOK du bloc (ou d'une partie du bloc),
- les résultats de valeur absolue qui est ce que le bloc de fonction a mesuré dans le référentiel de sa caméra,
- les résultats de valeur mesurée qui est la valeur absolue mais rapporté dans le référentiel de l'objet à mesurer.

qui sont accessibles depuis la fenêtre Unit Result, sous l'onglet « Unit Result » et affiche les résultats résumés ou les résultats détaillés du bloc (comme illustré sur l'image 5).

Tous les résultats détaillés sont stockés dans une structure nommée « RSLT » stockée dans la structure du bloc de fonction. Comme les variables normales, cette structure et donc ses sous-résultats sont accessibles depuis n'importe où dans le code (comme illustré sur l'image 5).

Unit Result					Unit Result						
Unit Results					Unit Results						
Unit	System	Local	Global		Unit	System	Local	Global			
Name	Measured Value	Upper Limit	Lower Limit		Name	Definition	Sta...	Nu...	Current...	Initial Value	
Segments	93				IU[0003].RSLT						
Detect Segments	7	----	----		UJG	Unit Result (NG=...	-	-	0		
Edge Width Max.	65.580	-----	-----		UJGP	Unit Result (NG=...	-	-	0		
Edge Width Min.	64.245	-----	-----		UJGN	Unit Result (OK=...	-	-	1		
Primary Target					UJGOK	Unit Result (OK=...	-	-	1		
Edge Width	65.580	-----	-----		UJNG	Unit Result (NG=...	-	-	0		
Target					UERR	Unit Error	-	-	0		
					UBID	Unit Error Code	-	-	0		
					EXTM	Execution Time	-	-	2.383		
					EXE	Unit Execution	-	-	1		
					EXEN	Unit Non-Executi...	-	-	0		
					EXCT	Processing Count	-	-	1		
					NGCT	Fail (NG) Count	-	-	0		
					SCLX		-	-	0.124893		
					SCLY	Scale Factor (Y)	-	-	+0.124893		
					SCLL	Scale Factor (Len...	-	-	0.124893		
					RGN		-	-	-		
					SGN:MS	Number of Seg...	-	-	93		
					DGN:MS	Number of Dete...	-	-	7		
					DGNJG	Number of Dete...	-	-	0		
					N		-	-	0 100		
					NH:MS	Maximum Numb...	-	-	1		
					NLO:MS	Minimum Numb...	-	-	1		
					W		-	-	0 100		
					WJG	Segment Width...	-	-	0		
					WH:MS	Maximum Width...	-	-	65.580		
					WH:AB	Maximum Width...	-	-	525.090		
					WHJG	Maximum Width...	-	-	0		
					WLO:MS	Minimum Width...	-	-	64.245		
					WLO:AB	Minimum Width...	-	-	514.397		
					WLOJG	Minimum Width...	-	-	0		
					WAMS	Average Width R...	-	-	64.956		
					WA:AB	Average Width R...	-	-	520.091		

Résultats résumés

Résultats détaillés

Accès aux résultats depuis le reste du code

Value	<input checked="" type="checkbox"/> IU[0003].RSLT.WH:MS	=	65.580145
-------	---	---	-----------

FIGURE 5 – Accès aux différents résultats de l'unité

1.3 L'utilisation des images par le contrôleur XG

Les contrôleurs XG manipulent 2 types d'images :

- **les images capturées** sont celles qui sont fournies par la caméra et sont soumises à un traitement (mesures/traitement d'image/etc...). Elles sont 'prises' au début du trigger au moment où le bloc « capture » est appelé, puis manipulées traitées tout au long du processus pour être supprimée à la fin, si aucune autre instruction n'est donnée. Ces images capturées sont stockées dans des variables images,
- **Les images de références** sont des images qui servent de référence (d'où leur nom) pour des opérations/calculs tels que la calibration, le pattern matching, ou encore comme base de donnée de caractères pour la lecture OCR. Elles sont enregistrées directement dans le contrôleur dans un tableau au moment de la création/édition du programme et **existent dès le début du programme et ne sont pas supprimées à la fin de l'exécution du programme**. Elles sont inaccessibles en tant que « variable » directement, mais

il est possible d'y accéder par l'onglet « Image » de tous les blocs qui nécessitent une image de référence.

1.3.1 Images de simulation

Lorsque le contrôleur fonctionne, les images capturées sont celles envoyées par la caméra. Mais lors du développement sur un PC sur VisionEditor, aucune caméra ne vient alimenter les blocs capture. C'est possible de simuler cela – c'est pour cela qu'on parle d'images de simulation – grâce à la fenêtre 'Simulation images' (ouverte en appuyant sur l'icône éponyme illustré sur l'image 6 où on peut glisser et déposer toutes les images que l'on souhaite faire passer dans le programme.

Chacun des blocs de capture possède une liste de slots dans lequel on peut glisser des images. Pour que les images soient valides, il faut que les images soient au **format '.bmp'** et aient **exactement la même dimension que la variable image dans laquelle elles seront stockées**. Pour chaque image valide, une miniature apparaît dans la liste. Pour valider la saisie, il suffit de cliquer sur le bouton OK.

Une fois les images insérées, les boutons de la section « Simulation » permettent de simuler le Trigger du programme :

1. Indique l'indice dans la liste de l'image 'actuelle' qui sera la prochaine à être insérée dans le programme,
2. Trigge le programme avec l'image actuelle,
3. Trigge le programme avec l'image suivante ou précédente dans la liste,
4. Passe en boucle toutes les images contenues dans la liste des images de simulation.

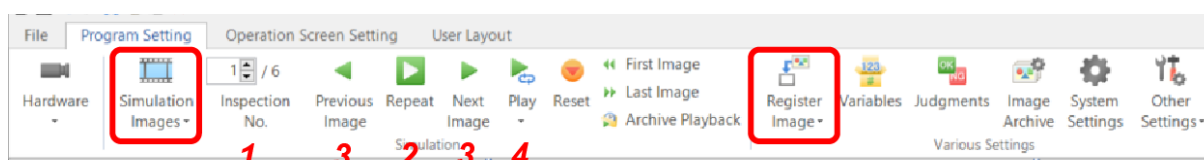


FIGURE 6 – Outil de gestion du Trigger en mode Simulation

1.3.2 Images de référence

Chacun des blocs nécessitant une image de référence possède un onglet de paramètres dans lequel il sera possible de spécifier la variable image sur laquelle la variable de référence sera calquée et le numéro de la variable de référence. Attention : ce numéro doit pointer sur une variable de taille et de type cohérents avec l'image pointée par la variable spécifiée dans « Image Variable ».

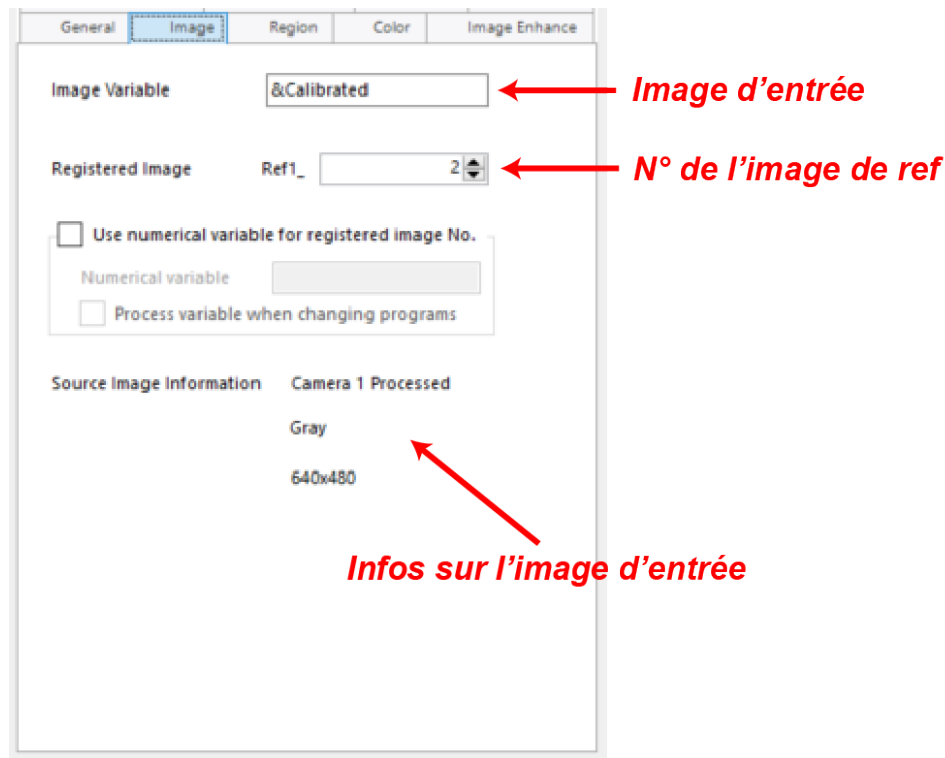


FIGURE 7 – Entrer une image de référence comme paramètre

Une fois ce paramètre introduit, le bloc calcule et travaille sur l'image de référence spécifiée. Pour faire commuter sur une image capturée, il suffit de modifier l'image traitée de "Reference Image" à "Captured Image".

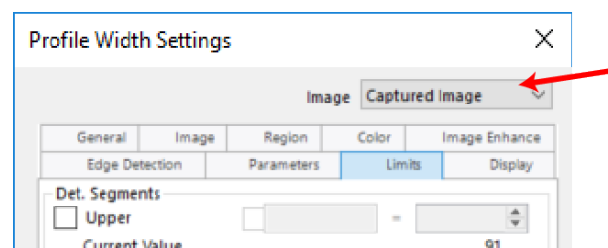


FIGURE 8 – Commuter le traitement entre l'image de référence et l'image capturée

Pour enregistrer une image de référence, il suffit simplement à tout moment de cliquer sur le bouton "Register Image" (illustré sur l'image 6) et une fenêtre s'ouvre pour nous permettre d'enregistrer une image actuellement dans le processus (traitée ou non).

Register Image

Select Image

Type: Captured Image

Image Variable: &Cam1Img

Camera No.: 1

Region: Edit(E)... Clear(L)

Options

Camera No.: 1

File Name: ref1_0

☐ JPEG View Images (P)...

Adjust Image Position

X: 0.000

Y: 0.000

Angle: 0.000

Reset (R)

Save (S) Close (C)

FIGURE 9 – Enregistrer une image de reference

Cette fenêtre permet de spécifier :

- **Type** : type d'image (référence/capture/fichier) à enregistrer,
- **Image Variable** : nom de la variable contenant l'image à enregistrer,
- **Camera No.** : le numéro de la caméra d'où vient l'image de référence,
- **Region** : la région de l'image à enregistrer,
- **File Name** : le numéro/nom que l'image enregistrée aura,
- **JPEG** : active la compression des images au format JPEG (utile lorsque la place est limitée),
- **View Images** : affiche toutes les images de référence enregistrées dans le contrôleur pour ce programme,
- **Adjust Image Position** : ajuste la position/orientation de l'image à enregistrer.

2 Partie pratique – création d'un programme KEYENCE

Dans ce labo, vous devrez créer un programme qui permette de :

- calculer et corriger la distorsion de la caméra,
- mesurer la longueur et la hauteur de différents objets,
- afficher les résultats à l'écran.

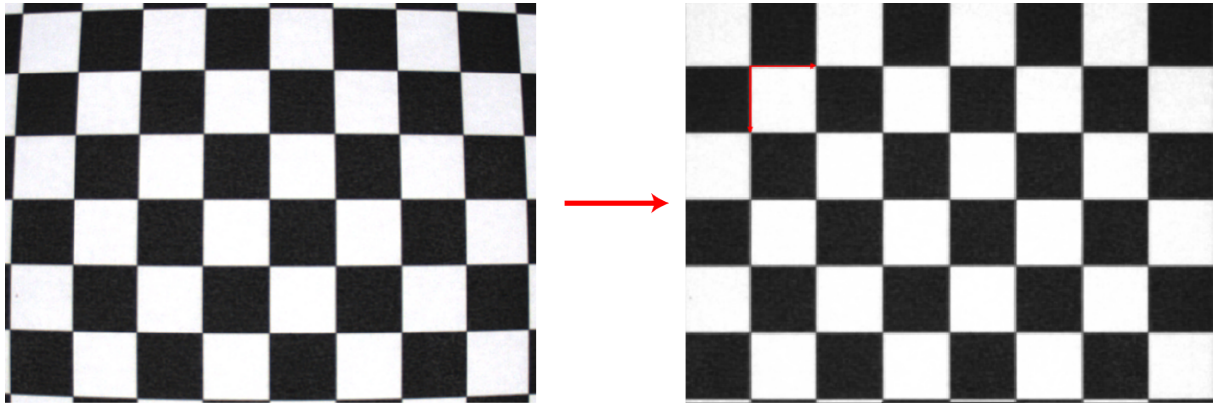


FIGURE 10 – Résultats de la calibration sur le damier de calibration

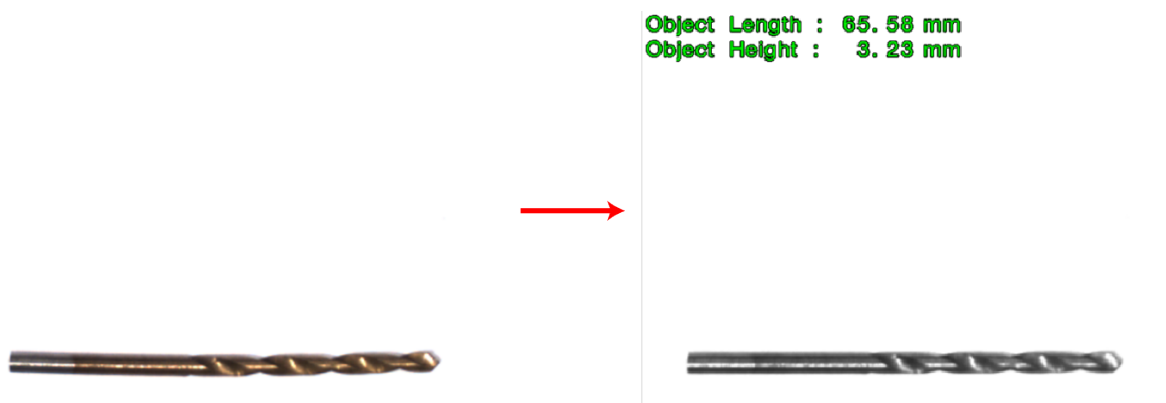


FIGURE 11 – Entrée et Sortie+affichage du programme demandé

Une partie des blocs de fonctions seront présentés durant le laboratoire, mais en cas de doutes/questions plus approfondis, référez-vous à la documentation fournie par KEYENCE

KEYENCE XG-X VisionEditor Reference Manual

qui contient toutes les fonctions et leur paramétrage.

2.1 Résultats observés

Reportez ci-dessous les dimensions des objets que vous avez mesurées sur les différentes images grâce à votre programme :

Objets		Hauteur [mm]	Largeur [mm]
Objet 1	<i>Théorique</i>	3	65.5
	image 1		
	image 2		
	image 3		
Objet 2	<i>Théorique</i>	35.5	8.5
	image1		
	image2		

TABLE 1 – Résultats mesurés avec votre programme