20200916

Le programme est inspiré du programme d’acquisition de la PCO.

Pour ne pas être gêné par les messages d’erreur du aux librairies absentes j’ai zippé les classes de la PCO et de caméra IDS.

Actuellement la seule caméra qui est opérationnelle c’est la caméra simuCalc. Les paramètres de cette caméra proviennent du fichier xml, paramSimu.xml. Ce fichier est dans le répertoire de la classe simuCalc Class. Dans la classe simuCalc on a deux fonctions SetParameter et GetImage.

## Acquisition d’un fond

### Ajout d’une fonction Acquisition Fond.

Basé sur le bouton Sav Ajout de Background

Dans la boucle ‘Logging’ on a trois états supplémentaires :

1. StartBackground
2. acqBackground
3. StopBackground

### Soustraction du fond à l’affichage

Dans la boucle ‘Data Display’ on a besoin de savoir si on soustrait le fond et quelle est la valeur de ce fond.

On commence par modifier Message Queues en ajoutant une FIFO Display. Il faut modifier : Create All Message Queues.vi

Mais je me rends compte que cette façon de procéder est lourde. Il faut ajouter une FIFO, il faut disposer du tableau de fond. Je pense qu’il faut traiter la soustraction du fond dans la partie Acquire.vi.

Première étape : on transforme Data Notifier en SGL pour tenir compte des valeurs négatives après soustraction du fond.

L’événement est envoyé à la boucle UI Message Loop

Cette boucle envoie la commande de soustraire un fond ou pas. L’état du bouton poussoir est transmis via le variant ‘Massage Data’. La boucle UI Message Loop retransmet le variant à la boucle d’acquisition.

20200918

Ajout d’un répertoire de Configuration config

On peut imaginer qu’on démarre avec "Launch Settings Dialog". L’utilisateur choisit la configuration. Ensuite on charge les données nécessaires ou au moins le chemin des données, exemple le fond, la matrice de commande, etc.

Dans la boucle Acquisition Message il faut définir une structure qui contient ces paramètres. Cette structure sera une ‘Type Def’.

* ROI string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\roi.fits
* NSSPUP int 2
* IMAGE string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\image.fits
* FOND string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\fond.fits
* SEUIL float 0.85
* REF\_RAZ string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\ref\_raz.fits
* REFERENCE string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\reference.fits
* NACTIONNEUR int 4
* TENS\_PERT string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\tens\_pert.fits
* MATCOM string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\matcom.fits
* TENS\_OFFSET string D:\USERS\MONTRI\BABYDOL\LABVIEW\EXEMPLE\_ARCHITECTURE\CONFIG\tens\_offset.fits
* GAIN\_BOUCLE float 0.00

Commande essentielles

Bouton arrêt ;

Recharger la configuration ;

Acquisition images ;

Fond ;

Seuillage ;

Acquisition de pentes

Acquisition de pentes de références, chargement de références, application, RAZ

Calcul de la Matrice de commande (Acquisition, calcul, application)

Tension d’offset (Acquisition, calcul, application)

Mode Step 1 acquisition et 1 application de la commande calculée

Boucle ouverte/fermée

Gain Boucle

Raz DM

Envoie manuel de tensions

Perturbation (choix fichier, gain, ON/Off)

20200923

### Lecture du fichier LUT

Si l’utilisateur ne lit pas de fichier LUT on conserve l’ancienne LUT.

### Lecture du fichier CONFIG

Développement d’une fonction simple qui lit le fichier config et cherche un mot clé, exemple fond. A partir du chemin du fichier de configuration on lit le contenu sous forme de chaine de caractères. Puis on recherche un mot clé. Si on trouve les caractères ‘:\’, on a un chemin. On lit la chaine de caractères après le mot clé jusqu’à la fin de la ligne  (\n). Puis on reconstruit le chemin.

Exemple la lecture du fond est utilisé dans la phase configure Hardware. Le fond est mis à zéro et le tableau est utilisé dans la boucle d’acquisition où on soustrait systématiquement un fond même s’il est nul.

### Seuillage

thresholdValue (85) en pourcentage du Max de l’image

Application du seuil

Le seuil, la commande d’application de seuil et la valeur du seuil sont utilisés dans la boucle d’acquisition. Elles forment un cluster, ce cluster doit aussi comporter la matrice de commande, les références, les tensions d’offset, etc.

La fonction qui calcule la max de l’image et applique le seuil en fonction du pourcentage du max de l’image est : applyThreshold.vi. Cette fonction reçoit la valeur du seuil et un booléen qui commande l’application du seuil.

Le seuil est appliqué dans "Acquire".

## Cluster avec les paramètres de la boucle de pointage fin

Le cluster comprend :

Le fond, le seuil, la commande d’application du seuil, la référence, la matrice de commande, les tensions d’offset

## Affichage des Centre de gravité

Premier test en modifiant juste l’affichage et pas la sauvegarde

La notification est transformée en notification de cluster.

Actuellement le cluster contient l’image et un tableau de SGL. Pour le premier test j’utilise le timeStamp donné par la caméra et le Frame Index.

## Modification de Data Queue

La file d’attente est une structure, cluster. Le cluster contient :

tabIma, tableau bidimensionnel d’entiers non signés ;

tabCdG, vecteur de flottants,SGL.

C’est aussi une définition de type. Il suffit de modifier la définition de type pour répercuter la nouvelle structure de la file d’attente dans tous les sous VI qui utilisent cette file d’attente.

20200930

## Ajout du calcul de Centre de Gravité

On se base sur la fonction issue de PERSEE. Cette fonction est ajoutée au sous VI, Acquire.vi. Grâce à la boucle qui gère les événements utilisateurs on peut calculer le centre de gravité ou le dé valider. On transmet la valeur du booléen grâce aux variants données du message.

TBD : Dans la partie affichage on a un affichage continu. On veut arrêter l’affichage du CdG si le calcul de ce dernier est dévalidé.

A faire :

Modifier Acquire.vi pour avoir l’enchainement suivant :

1. Application de la tension de commande ;
2. Acquisition d’image ;
3. Calcul du CdG ;
4. Calcul de la commande ;
5. Mis à jour de la file circulaire.

Pour le calcul des commandes on pourra utiliser la file circulaire pour accéder aux données des itérations précédentes.

La fonction Acqire.vi reçoit tous les paramètres nécessaires via le cluster pointagFin :

Cette structure contient :

1. Le fond ;
2. La valeur de seuil ;
3. Les pentes de référence ;
4. La matrice de commande ;
5. Le gain de boucle ;
6. Les tensions d’offset ;
7. Le vecteur de tension de perturbation ;
8. Le vecteur de perturbation manuelle ;
9. la dernière image ;
10. le centre de gravité calculé ;
11. la commande calculée.

On ajoutera aussi les booléens du type :

1. Application du seuil ;
2. Calcul du CdG
3. Boucle ouverte ou fermée

20201006

## Intégration de la partie commande de miroir dans le logiciel

Comment intégrer le commande ?

Est-ce qu'on ajoute le module commande dans la partie acquisition ? Est-ce qu'on crée une boucle dédiée ? Cette boucle pourrait être une boucle de traitement de des données. Elle prend la dernière image de la FIFO et fait le traitement.

On choisit une approche machine d'état dans la boucle d'acquisition. La boucle d'acquisition va avoir trois états :

1. Acquire ;
2. Center of gravity calculation ;
3. Calculation of control voltages ;
4. data FIFO update.

Ces trois états sont exécutés de manière séquentielle tant qu'on est en phase de boucle fermée. Si on ouvre la boucle on peut avoir juste une acquisition de commande seule ou la séquence acquisition calcul de centre de gravité.

A chaque étape on met à jour le cluster pointagFin. Ce cluster contient donc la dernière image acquise, le centre de gravité, la commande calculée.

La FIFO de données est mis à jour à la fin de la séquence acquisition-calcul de centre de gravité-calcul de commandes, idem pour les données destinées à l'affichage.

Ajout de closeLoop et de loop gain (0)

La boucle de message UI permet de commander la boucle d'acquisition. Cette dernière récupère la valeur de la commande closeLoop et de loopGain pour mettre à jour le cluster PointagFin. Ensuite on fait le cycle acquisition calcul de centre de gravité, calcul de commandes et mis à jour de la FIFO.

TBD : Configuration du miroir, pilotage miroir

20201008

Mise à jour de la librairie ALPAO

Revision Version: 3.03

Test avec ampli branché OK

Modification de la boucle d'acquisition pour initialiser le miroir et libérer la ressource

On commence par modifier HardwareConfiguration.ctl en ajoutant la classe mirror.lvclass.

Dans l'état Initialize on impose la classe SIMU\_MIRROR

Dans l'état Start :

Initialisation

Open.vi

Configuration

Setup\_Configuration

getNbActuator

Dans l'état Stop

Modification de Stop Acquisition

Close

20201009

## Acquisition des données

On acquiert

1. les images brutes;
2. Le vecteur du centre de gravité;
3. Le vecteur de commande.

On appelle trois fonctions d'écriture

Une fonction pour l'acquisition des images, il s'agit d'acquérir un cube d'images au format U16.

Pour le centre de gravité et le vecteur de commandes il s'agit de 2 valeurs en SGL. A l'avenir on aura 4 valeurs.

## Premiers tests

Relecture avec IDL :

Problème dans la manière de sauvegarder confusion entre lignes et colonnes.

Ensuite il y a des états qui se répètent correction en ajoutant des "flush" au start de l'acquisition et à la mise à jour des queues de données. Vérification en se basant sur l'acquisition du numéro d'image.

Par contre le calcul de centre de gravité n'est plus bon.

20201022

Dépôt sut GitHub. Ce n’est pas entièrement clair pour moi.

Clone sur le PC PICOLO, WDOTA970Z.

On utilise tortoise GIT, on peut configurer le proxy, proxy.onera port 80 dans la partie réseau

Installation de la librairie LabVIEW pour Hamamatsu et DCAMPAPI, c’est juste pour la compatibilité.

Commentaires de la classe AMPLI ALPAO

On utilise la classe simu\_mirroir