Automatisation de l'analyse d'objets déformables

Jonathan Guillotte-Blouin, Dr Pierre Payeur – École de science informatique et de génie électrique

Introduction

Dans le cadre d'un projet de recherche ayant pour but de localiser des objets à l'aide d'une Kinect et de les classifier dans différentes catégories allant d'objets rigides à élastiques, il était auparavant nécessaire de manuellement prendre ces différents objets et de les comprimer devant ladite caméra.

Pour aider dans la démarche, mon but consiste à remplacer le processus manuel par une routine programmée et exécutée par une main robotique. La main est consitutée de trois doigts qui peuvent s'ouvrir et se refermer, ainsi que d'un moteur permettant d'éloigner ou de rapprocher les doigts 2 et 3.

Pour y arriver, il a fallu remettre en marche la main, la configurer, lire la documentation fournie, tester son fonctionnement par l'entremise de son application graphique, puis finir par l'écriture de code C++ utilisant les librairies fournies par le fabricant. Le tout a mené à une routine automatisant le processus pour des objets de différentes formes et tailles.

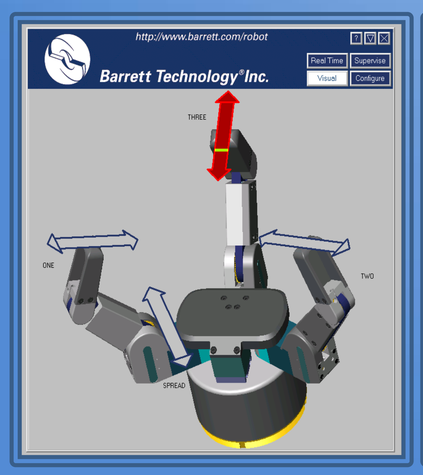
Méthodologie

Initialement, il a fallu brancher la main à l'ordinateur ainsi qu'à la source de courant. Ensuite, l'application graphique, les librairies et la documentation ont été installées sur l'ordinateur.

Puis, je me suis familiarisé avec l'application graphique offerte pour commander facilement la main, et j'ai lu la documentation. Suite à cela, j'ai lié la librairie dynamique (*.dll*) à mon projet, et j'ai écrit des programmes simples en C++ pour tester les fonctionnalités de la librairie, et voir si la main fonctionnait bien via un exécutable compilé.

Finalement, j'ai écrit des programmes pour résoudre le problème, et je me suis rabattu sur une solution fonctionnant pour tout type d'objet à la place de plusieurs routines spécifiques à chaque objet.

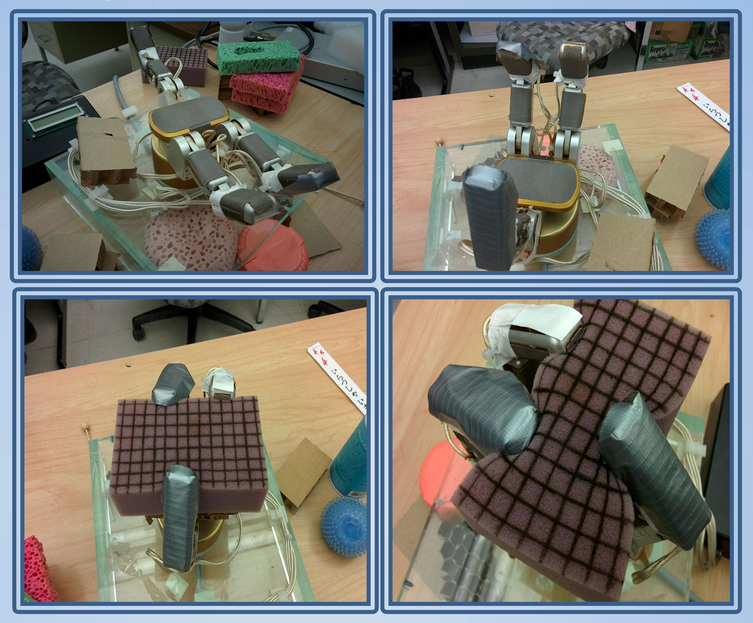
```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <limits>
#include "BHand.h"

#define NUM_OF_REPETITIONS 3 // <--- you can change the number of squeeze repetitions here
#define DELAY_TIME 2000 // <--- you can change the delay between repetitions (in ms)
#define PORT_NUMBER 1 // <---- set the port number of the hand

BHand  bh;          // Handles all hand communication
int    value;       // Hand parameter obtained with Get
int    result;      // Return value (error) of all BHand calls

// Error Handler - called whenever result!=0
void Error(void)
{
   printf("ERROR: %d\n%s\n", result, bh.ErrorMessage(result));
   exit(0);
}

//  Initialize hand, set timeouts and baud rate
void Initialize(void)
{
   if (result = bh.InitSoftware(PORT_NUMBER, THREAD_PRIORITY_TIME_CRITICAL))
      Error();

   if (result = bh.ComSetTimeouts(0, 100, 15000, 100, 5000))
      Error();

   if (result = bh.Baud(9600))
      Error();

   if (result = bh.InitHand(""))
      Error();
```



Résultats

La routine demandée en est une où la main s'ouvre et se resserre sur l'objet pour un nombre défini de répétitions. Le résultat escompté a été atteint, avec la plus grande automatisation possible.

La stratégie initiale était fonctionnelle: pour un certain objet, la main pouvait se positionner correctement, se reserrer, puis se rouvrir le nombre désiré de fois. Cependant, dès que l'on voulait essayer pour un objet d'une différente forme ou taille, il fallait changer manuellement les valeurs de positionnement des doigts.



Devant ce constat, j'ai rendu la routine plus flexible: la main commencerait ouverte, se refermerait lentement, et lorsqu'une touche sur le clavier serait pesée, la main retiendrait cette position précise comme la position de départ de serrage.

```
// move hand to default position manually (first key hit to slow down, second hit to stop)
int goToDefaultManual() {
    while (!_kbhit()) {
        if (result = bh.StepClose("G", 150))
            Error();
    }

    _getch();
    printf("slow stepping");

    while (!_kbhit()) {
        if (result = bh.StepClose("G", 50))
            Error();
    }
    _getch();

    printf("out of default method");
    return 0;
}
```

Il était maintenant possible de rouler un seul exécutable pour tout objet. Cependant, le processus demandait l'interaction avec l'utilisateur, il y avait donc place à une plus grande automatisation. Pour s'y faire, j'ai utilisé la mesure du *strain gauge* (jauge de déformation), qui permet de mesurer la déformation d'une pièce, dans ce cas-ci des moteurs des doigts. Au fur et à mesure que les doigts se referment sur l'objet, une plus grande déformation est détectée. Grâce à cela, j'ai complètement automatisé la routine: la main détecte lorsqu'elle est en contact avec l'objet, se rappelle de sa position puis se referme et se rouvre comme auparavant.

```
// move hand to default position automatically
int goToDefaultAutomatic() {
    int sgLast = INT_MAX, sgCurr, lastDelta = -1, currDelta;
    const int THRESHOLD_STRAIN_GAUGE = 4; // modify if strain values get different - use "testingStrainGauge" to see if good or not

    while (!_kbhit()) {
        if (result = bh.StepClose("G", 150)) {
            std::cout << result << "humm" << std::endl;
            return 1;
        }

        bh.Get("3", "SG", &sgCurr);

        currDelta = sgCurr - sgLast;

        // object detected
        if (currDelta >= THRESHOLD_STRAIN_GAUGE && lastDelta >= THRESHOLD_STRAIN_GAUGE)
            return 0;

        // put current values to last values
        lastDelta = currDelta;
        sgLast = sgCurr;
    }

    return 0;
}
```

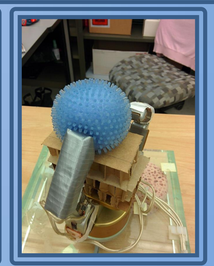
Conclusion

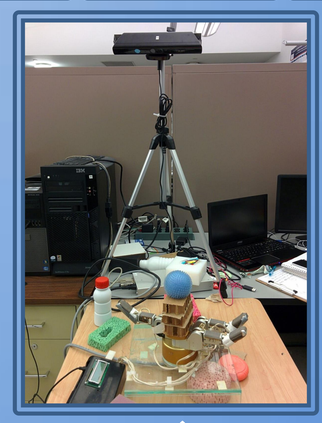
En conclusion, le but de la recherche a été accompli : initialement, il fallait manuellement serrer et relâcher l'objet, et maintenant une solution complètement automatisée est fonctionnelle.

Cette avancée a permis à la recherche de ma collègue d'avoir des prises de données plus rapides, constantes et précises vis-à-vis la prise d'objets manuellement. De plus, la recherche a permis de mettre au jour que la solution initiale d'une routine écrite par type d'objet pouvait grandement être optimisée à l'aide d'entrées sensorielles de la jauge de déformation.

À partir d'ici, une optimisation est envisageable: étudier davantage les valeurs de la jauge, et voir s'il existe une manière plus complexe de détecter le contact avec un objet, en prenant en compte l'usure de la main au fur et à mesure de son utilisation.

```
Initialization... Done
Executing -

Press Any Key to Abort...
M1: 8964
M2: 9838
M3: 8984
M4: 11
Do again? y/n: y

Press Any Key to Abort...
M1: 8966
M2: 9824
M3: 8972
M4: 11
Press any key to continue . . .
Do again? y/n: y

Press Any Key to Abort...
M1: 11362
M2: 11443
M3: 11382
M4: 14
Do again? y/n: y

Press Any Key to Abort...
M1: 9681
M2: 9792
M3: 9686
M4: 21
TOO RIGID....
Press any key to continue . . .
Do again? y/n: y
```





Remerciements

J'aimerais remercier mon directeur de recherche, Dr Pierre Payeur, pour la confiance et l'aide qu'il m'a donnée, ainsi que ma collègue Fei Hui, candidate à la maîtrise, avec qui j'ai travaillé pour son projet de recherche. De plus, j'aimerais remercier le *Programme d'initiation à la recherche au premier cycle* (PIRPC) pour cette unique opportunité et le soutien qu'il m'a apporté.

Jonathan Guillotte-Blouin
jguil098@uottawa.ca