Treball Final de Carrera

Processat de vídeo aplicat a la robòtica

Isaac Micó Rodríguez

Enginyeria Tècnica Industrial especialitat electrònica industrial

Directors: Pere Martí, Ramon Reig

Vic, febrer de 2012

ÍNDEX

1- INTRODUCCIÓ

Com ja va predir E. Moore (1965) en la seva llei, la tecnologia digital gairebé ha duplicat el seu nombre de transistors per unitat de superfícies cada divuit mesos. En la pràctica, aquesta llei es reflexa en que a l'any 1995 es treballava amb xips de l'ordre de 10.000 transistors per unitat de longitud, a l'any 2005 amb xips de 100.000 transistors i actualment utilitzen més d'un milió de transistors. Aquesta espectacular progressió ha revolucionat completament la tecnologia, fins al punt de trobar-nos en el camp domèstic petits aparells capaços de realitzar milers de càlculs per segons. En el camp industrial, tenim robots i màquines de càlcul numèric que són capaços de fer milions de càlculs per segon, el qual hagués sigut impensable en el passat.

Com hem pogut veure en el transcurs de l'última dècada, la potència de càlcul de les màquines modernes ha provocat el desenvolupament de les tècniques de control industrial fins al punt de poder processar models discrets d'objectes amb tres dimensions dins d'un espai de treball d'un robot amb sis graus de llibertat, càlcul de trajectòries i de moviment d'eixos per fer interpolacions o moviments en temps real, capacitat d'actuació sobre motors amb precisió, sistemes de sensors que mesuren variables físiques amb graus de precisió molt elevats, etc.

En el cas de les tecnologies de la comunicació, els algoritmes de processament d'imatge s'han desenvolupat fins al punt de detecció i seguiment de diferents objectes dins d'una imatge, transformades per extreure informació de contorns, algoritmes de processament estereoscòpics, processament de vídeo a temps real, etc.

Després de buscar, analitzar i recopilar la informació necessària per el meu projecte, vaig decidir fer un ús combinat de la tecnologia de robòtica i processament d'imatge a partir d'un robot i una webcam, ja que considero que es un camp amb moltes possibilitats d'investigació de cara a futurs projectes..

Per últim voldria avisar de que les errades d'ortografia d'aquesta memòria en els trossos que fan comentaris a algoritmes són degudes a que si possés accents, al copiar el codi, aquest no compilaria per no reconèixer els accents. Alguns exemples són: *“parametres, deteccio, direccio, etc..”.*

2- OBJECTIUS

L'objectiu principal d'aquest treball és desenvolupar competències bàsiques de programació de l'alumne. Tot i això, en el desenvolupament d'un projecte d'investigació no només s'aprèn la part tècnica, també s'aprèn a ser imaginatiu resolent els problemes que sorgeixen per el camí, s'aprèn la importància de l'organització i la metodologia a l'hora d'investigar sobre algun tema, a més, es desenvolupa el sentit de superació personal.

Els objectius tècnics d'aquests projectes són els següents:

1. *Instal·lació i funcionament de llibreries amb Ubuntu:* es desenvoluparà la capacitat a l'hora de resoldre errors en instal·lacions de llibreries.

2. *Programació amb llenguatge python:* estructures de dades, programació condicional, dispositius externs, programació orientada a objectes, etc.

3. *Disseny de circuits electrònics per adaptar nivells de voltatge:* circuits bàsics d'aïllament entre fonts d'alimentació.

4. *Posició, trajectòries i sistemes de coordenades amb espais 3D:* funcionament del programa *robotstudio*

5. *Programació amb llenguatge Rapid:* tipus de sistemes de coordenades, estructures condicionals, lectura d'estat de senyals digitals d'entrada, instruccions de moviment, etc.

6. *Tècniques de reducció de ressonàncies:* temps d'espera, zones neutres, marges d'error i altres coeficients per reduir les ressonàncies d'un sistema a temps real.

3- VIVISIÓ GLOBAL

Per tal de poder definir més específicament el contingut i en què consisteix aquest projecte he decidit explicar-ho per parts per fer-ho més intel·ligible. S'ha de tenir en compte que l'objectiu d'aquesta memòria està enfocat de cara a que futures investigacions puguin partir de la base creada per aquest projecte. És per això que en aquesta memòria s'ha fet un esforç de síntesi per tal que aquesta pugui ser utilitzada com una guia de consulta per futures aplicacions relacionades amb robòtica, processament i utilització de la càmera uEye de la que disposa la Universitat de Vic.

Aquest projecte consisteix en acoblar una càmera web *uEye* en una de les extremitats d'un robot industrial (ABB Irc5). Aquest robot, es mourà en cas què es detecti una cara a través de la *webcam*. L'objectiu és intentar que la cara es situï sempre al mig de la imatge captada per la càmera. En la *figura 1* podem veure el funcionament del moviment de rotació del robot vist des de dalt.

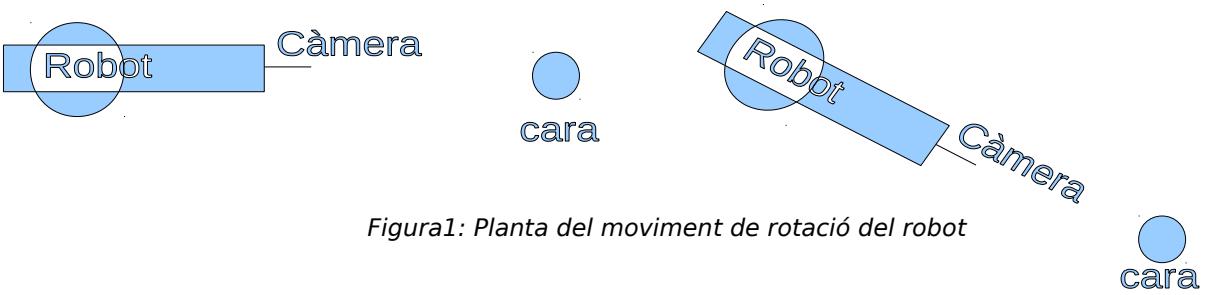


Figura1: Planta del moviment de rotació del robot

L'altre moviment que realitza el nostre robot és la translació per interpolació lineal. En la *figura 2* podem veure el funcionament del moviment de translació del robot vist des del costat.

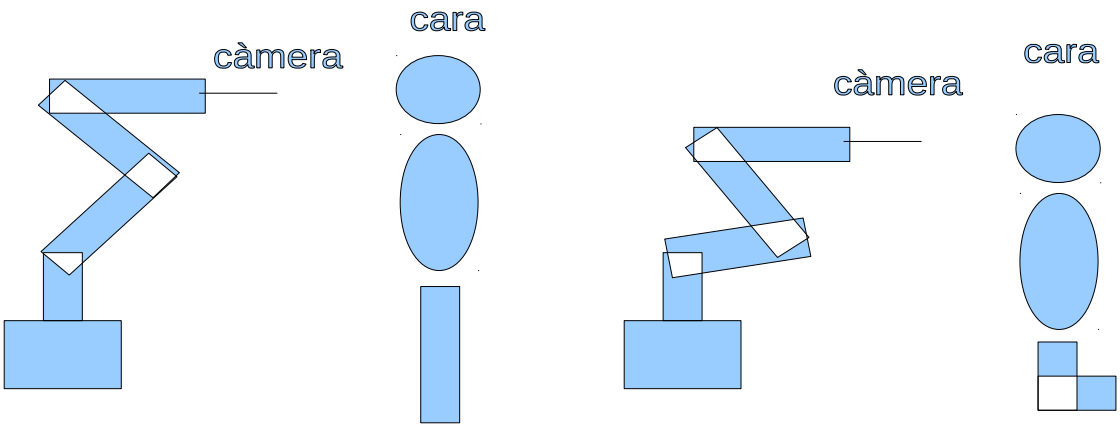
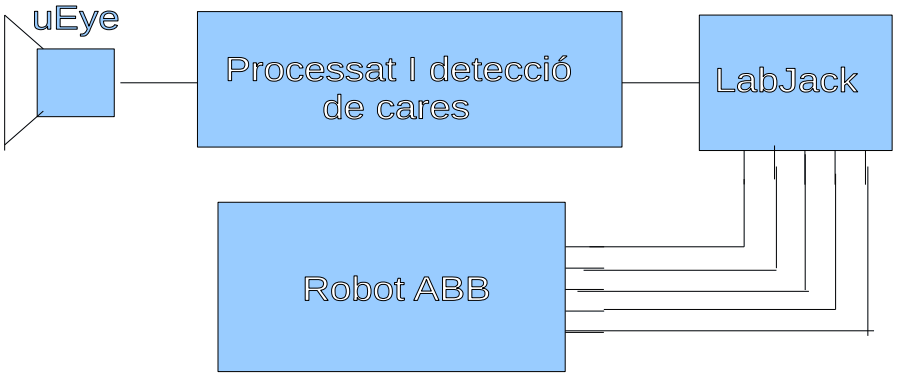


Figura2: Perfil del moviment de translació del robot

Per facilitar la programació del conjunt, podem dividir les tasques amb 4 fases diferents. A continuació expliquem les diferents etapes (les paraules amb negreta són els apartats generals d'aquesta memòria).



La **Càmera uEye** s'encarrega de capturar imatges i passar-les a un PC utilitzant Python. La funció de **Detecció de Cares** ens indica la posició i dimensions de la cara. Tot seguit, la funció **LabJack** s'encarrega de canviar els valors digitals de les 5 senyals mitjançant programació amb Python. L'algoritme que conté tot aquest procés és l'**Algoritme Python.** Per últim el **Robot ABB** interpreta el valor d'aquestes 5 senyals i es desplaça mitjançant increments amb l'objectiu de situar la cara al mig de la imatge capturada.

Aquestes 5 senyals de control són les següents:

1. Detecció de cara
2. Rotació amb sentit horari
3. Rotació sentit antihorari
4. Increment d'altura
5. Decrement d'altura

4- CÀMERA UEYE

4.1- Instal·lació drivers

Primer que tot, és important remarcar la dificultat a l'hora de treballar amb una webcam que no suporti les llibreries v4l (vídeo for linux) com és el cas de l'ueye. Tot i això, instal·lant els drivers i llibreries indicats, és pot aconseguir un funcionament adequat.

El primer que s'ha de fer és instal·lar els drivers de la càmera ueye proporcionats per el fabricant. Els podem trobar a la pàgina web del fabricant: *www.ueyesetup.com*. En el cas d'aquest projecte, estem funcionant amb la versió de drivers 3.50 amb ubuntu 10.04.

Cal remarcar que s'han de tenir uns mínims coneixements informàtics per tal d'assolir una correcta instal·lació. En qualsevol cas, s'intentarà donar una guia d'instal·lació ben detallada.

Per instal·lar els drivers cal baixar-se'ls del link comentat anteriorment. En cas de no trobar la versió 3.50 funcionant a 64 bits a la web del fabricant, poden ser baixats des de el següent enllaç:
http://code.google.com/p/pyuEye/downloads/list

Un cop baixats, procedim a la instal·lació mitjançant la comanda *sh ./* En cas de no saber utilitzar la consola d'aplicació, seguiu els passos mostrats a continuació.

Ens situem amb la consola d'aplicacions al directori on hem descarregat els drivers amb la comanda *cd*. Per tal d'assegurar-nos d'estar situats al directori indicat, podem usar la comanda *ls* per que ens mostre els arxius del directori on estem situats. Per exemple:

```
maik@maik-laptop:~$ cd Desktop/uEye_Linux_350_64Bit/     (ens situem al directori desitjat)
maik@maik-laptop:~/Desktop/uEye_Linux_350_64Bit$ ls      (mostrem en pantalla tot els arxius)
uEye_Handbuch_Dateien      ueyesdk-setup-3.50-eth-amd64.gz.run
```

uEye_Manual_EN_Start.html ueyesdk-setup-3.50-usb-amd64.gz.run
uEye_Manual_Files

A continuació instal·lem els drivers amb la comanda *sh* amb nivell d'administrador. És a dir, situats en el directori on hem descarregat els drivers:

sudo sh ./ueyesdk-setup-3.50-usb-amd64.gz.run

(la comanda sudo serveix per executar la instrucció amb permisos d'administrador)

En cas d'una instal·lació correcta, passeu al següent apartat. En cas que la instal·lació no s'hagi efectuat de manera adient, a continuació mostrem els típics erros a l'hora d'instal·lar i com solucionar-los.

Els errors més típics a l'hora d'instal·lacions de drivers per a programari són la falta de llibreries. En el cas del uEye, són necessàries les següents:

libc6 (llibreries de C/C++ necessàries per a la instal·lació)
libqtgui4
libqt4-network (necessàries per l'aplicació camera manager)
libqt4-qt3support
libqt4-opengl (necessàries per l'aplicació ueyedemo)

Per instal·lar-les, utilitzarem la següent comanda:

*sudo apt-get install *nom_de_la_llibreria*

Per exemple:

sudo apt-get install libqtgui4

Un cop instal·lades totes les llibreries, podem provar un altre cop d'instal·lar els drivers. *(sudo sh ./ueyesdk-setup-3.50-usb-amd64.gz.run).* En cas d'instal·lació correcta, saltar al següent apartat. En cas que la instal·lació segueix sense funcionar, a continuació mostrem com trobar la resta de llibreries necessàries per a la instal·lació.

Normalment, quant tenim un error d'instal·lació, en apareix a la pantalla de la consola un missatge del tipus */../libc.so.6 :: not found.* Per tal de saber a quina llibreria pertany aquest arxiu, podem consultar a la següent pagina web:

http://packages.ubuntu.com/

Busquem a l'apartat '*search content of packages*' el nom de l'arxiu que necessitem per a la instal·lació. Premem buscar, i a continuació ens mostra el nom de la llibreria que conté l'esmentat arxiu.

Per exemple:

Ens apareix un error d'instal·lació: */lib/i386-linux-gnu/libc.so.6* :: packages not found

Busquem a quina llibreria pertany aquest arxiu com hem mostrat anteriorment.

libc.so.6 pertany a la llibreria libc6. Per tant, instal·lem aquesta llibreria:

sudo apt-get install libc6

Veurem que ja no ens apareix aquest error a l'hora de la instal·lació. Instal·lem totes les llibreries necessàries fins efectuar una correcta instal·lació dels drivers.
Per a més informació o suport, podeu contactar a *isak.mico@gmail.com*.

4.2- Funcionament i directoris

Un cop instal·lats els drivers, podem provar les eines que ens ofereix el fabricant com el cameramanager o ueyedemo en el següent directori:

/usr/local/share/ueye/bin/ueyedemo

Per activar/desactivar els drivers, utilitzarem la següent comanda:

sudo /etc/init.d/ueyeusbdrc {start,stop,status}

En cas de voler treballar amb C/C++ podem trobar els codis de les aplicacions ofertes per el fabricant al directori:

/usr/src/ids/ueyedemo

L'arxiu uEye.h, necessàri per compilar amb entorns C/C++ està situat en el directori:

/usr/include

Per des-instal·lar els drivers, podem teclejar la següent comanda:

sudo /usr/local/share/ueye/bin/ueyed_install-usb uninstall

4.3- Instalació Python-uEye

Un cop comprovat el correcte funcionament dels drivers procedim amb la instal·lació de Python i el programari necessari per utilitzar l'ueye.

És important remarcar que els drivers que ens proporciona el fabricant no estàn preparats per treballar amb certs llenguatges d'alt nivell com és el cas del python. Per tant, serà necessari instal·lar unes llibreries nombrades pyuEye. Aquestes llibreries han estat desenvolupades per Ricardo Amezquita Orozco i poden ser trobades al següent link:

http://pyueye.googlecode.com/svn/trunk/

Per instal·lar aquestes llibreries ens descarreguem tots els arxius del link i els situem en el mateix directori. I procedim a la instal·lació de llibreries necessàries per tal de que ens funcioni pyuEye.

El primer que necessitem instal·lar és Python. En aquest projecte s'ha utilitzat Python2.6, a més és necessari instal·lar les llibreries de desenvolupador python-dev, les llibreries python2.6-wxgtk2.6, les llibreries de treball vectorial python-numpy i el programari Cython. Aquest últim és degut a que el pyuEye funciona utilitzant algunes llibreries de C/C++ com ara els wx.*. Per tant, el Cython s'encarrega de compilar les esmentades llibreries en C/C++ per a Python.

Per tant, comencem instal·lant el programa Cython. El podem trobar a la web: www.cython.org. Descarreguem la última versió i extraiem els arxius en un directori. Ens situem amb la consola d'aplicacions en el directori on tenim tots els arxius d'instal·lació de Cython i teclejem la següent comanda:

sudo python setup.py install

Un cop instal·lat Cython correctament, procedim amb la instal·lació de python i les seves llibreries necessàries per a pyuEye. Per tant, teclejarem les següents comandes:

sudo apt-get install python2.6
sudo apt-get install python-dev
sudo apt-get install python-numpy
sudo apt-get install python2.6-wxgtk2.6
sudo apt-get install python-imaging

A continuació, ens situem amb la consola d'aplicacions en el directori on hem descarregat tots els arxius d'instal·lació de pyuEye i teclegem la següent comanda:

sudo python setup.py install

En cas d'una instal·lació incorrecta, serà necessari comprovar quines llibreries ens falten per una correcta instal·lació com s'ha explicat en l'apartat “Instal·lació drivers”

Un cop ho tenim tot ben instal·lat, podem procedir a provar el funcionament d'aquest drivers i llibreries per utilitzar la camera ueye amb python.

4.4- Captura imatge/video amb Py-uEye

PyUeye és un conjunt d'arxius i llibreries desenvolupats per poder utilitzar les càmeres uEye amb un llenguatge d'alt nivell com el python capturant les imatges com a vectors del tipus Numpy. Com hem comentat anteriorment, aquestes llibreries estan desenvolupades mitjançant Cython, és a dir, aprofita certes llibreries de C/C++ com ara WxVidCap i uEye.c entre d'altres. També és important remarcar que aquests plugins no han estat desenvolupats per IDS i es troben en estat alpha, per tant, els utilitzarem completament baix la nostra responsabilitat. Tot i això, funcionen perfectament.

Podríem comentar totes les funcions per a la càmera uEye que ens proporciona el pyuEye. Tot i això, l'objectiu d'aquesta memòria és documentar de forma abreviada el desenvolupament de l'aplicació. Per tant, només es comentaran les funcions necessàries

per a l'aplicació.

Per tal de fer-ho més fàcil d'entendre, podem mirar-nos l'algoritme *"captura.py"* en aquest algoritme veiem com poder fer una captura amb l'uEye i mostrar-la per pantalla. Utilitzant les llibreries Image.

Primer que tot serà necessari importar totes les llibreries necessàries per aquest programa: ueye, numpy i Image.

Les llibreries *numpy* són necessàries degut a que els drivers de l'ueye ens passen la informació de la captura amb format numpy-array. Les llibreries Image són necessàries per fer la conversió de vector (array) a format d'imatge PIL (mòdul Image) i per poder-ho mostrar per pantalla.

A continuació comentem el codi per tal de mostrar el que fa cada instrucció.

```
#Importem les llibreries necessàries
import ueye
import numpy
import Image
# Creem la variable 'a' de la classe ueye
a=ueye.Cam()
 # Capturem una imatge (la imatge ens be amb format vector numpy)
b=a.GrabImage()
#Convertim el vector a objecte del tipus Image.image
im = Image.fromarray(b)
# Mostrem la captura per pantalla aprofitant les llibreries Image
im.show()
#Imprimim per pantalla el tamany de la imatge per tal de saber amb quin
#format estem treballant
print im.size
```

En el cas de l'aplicació dissenyada per a aquest projecte, no seran necessàries cap altres instruccions referenciades amb l'ueye. Tot i això, a continuació es mostren altres instruccions importants de cara a futurs projectes o algoritmes que incloguin una càmera uEye.

```
ncam=GetNumberOfCameras()  # Retorna el nombre de càmeres connectades.
dblFPS = cam.GetFramesPerSecond() # Retorna els nombre de frames per segon
newFPS=cam.SetFrameRate(FPS)  # Configurar el frame-rate de la càmera
rv=cam.SetBlCompensation(nEnable, offset) #Adjust de nivell de grisos
rv=cam.SetGamma(nGamma)  # Aplicar una correcció gamma
rv=cam.ResetToDefault()  # Resetejar a valors predeterminats
rv=cam.SetColorMode (Mode)  # Canvi de mode de color
```

Les llibreries pyueye, inclouen moltes altres funcions que poden ser molt útils de cara a treure el màxim partit possible a la nostra càmera. Per poder veure totes les instruccions que ens ofereixen aquestes llibreries, així com la sintaxis d'aquestes, podem consultar l'arxiu *"ueye.pyx"* situat a la carpeta "ueye" del directori on em descarregat tots els arxius d'instal·lació del pyueye.

Un cop sabem com capturar un fotograma podem realitzar el mateix 25 cops per segon per tal d'obtenir una successió de 25 imatges per segon aprofitant les llibreries opencv. Un bon exemple de com treballar amb vídeo el tenim en l'arxiu "video_continu.py" de la carpeta "Algoritmes" adjunt a l'apèndix. A continuació comentem el codi:

```
import ueye
import numpy
import Image
from opencv.cv import *
from opencv.highgui import *

if __name__ == '__main__':
    # Petita benvinguda
    print "OpenCV Python capture video for uEye"

    # Capturem una imatge de l'uEye i la transformem a format opencv
    a=ueye.Cam()
    b=a.GrabImage()
    im = Image.fromarray(b)
```

```
# Creem una imatge amb format opencv
cv_im = cvCreateImage(im.size, 8, 3)
# Copiem el contingut de la imatge capturada per l'ueye
cv_im.imageData = im.tostring()


cvNamedWindow("Example1", CV_WINDOW_AUTOSIZE)   #Creem una finestra per
                                                #mostrar la imatge
cvShowImage( "Example1" , cv_im )    #Mostrem el primer fotograma per pantalla

# creem un bucle on s'anira actualitzant i mostrant en la mateixa finestra la
# imatge de la webcam cada 1000/25 ms (25 fps)

while True: # Fes indefinidament
        # Capturem una imatge de l'uEye, la transformem a format opencv i la
        # mostrem per pantalla
         b=a.GrabImage()
        im = Image.fromarray(b)

        cv_im.imageData = im.tostring()
        # Esperem el temps de fotograma (200ms)
        cvWaitKey(1000/25)
        # Actualitzem la imatge de la finestra
        cvShowImage( "Example1" , cv_im )
cvDestroyWindow( "Example1" )
```

El principal avantatge de treballar amb format opencv és el poder utilitzar totes les funcions d'imatge que ens ofereixen les llibreries OpenCv. Com es mostrarà més endavant, en aquesta memòria, aquestes ens ofereixen moltíssimes possibilitats. Un bon exemple de la potència de les funcions d'aquestes llibreries és el reconeixement de cares utilitzat en aquest projecte.

5- DETECCIÓ CARES

5.1- Introducció i concepte Haar Cascade

Un procés de reconeixement pot resultar molt més eficient si està basat amb la detecció de característiques o contrastes que contenen informació sobre el tipus d'objecte a detectar. Aquest és el cas de les característiques tipus “Haar” les quals codifiquen contrastes orientats entre regions d'una imatge. Un conjunt d'aquest tipus de característiques pot ser utilitzar per codificar els contrastes amb la seva relació espacial, per exemple, per una cara humana. Les característiques “Haar” poden ser representades per coeficients de la transformada “Haar wavelet”. Aquesta és una transformada matemàtica que ens representa amb coeficients els canvis espacials que conté una imatge. Per tant, podem detectar objectes comprovant si aquests canvis espacials coincideixen amb els canvis espacials que produeix un determinat objecte.

L'algoritme HaarDetectObject() que contenen les llibreries openCV va ser proposat per Paul Viola i millorat per Rainer Lienhart. Primer que tot, un classificador (conjunt de classificadors que treballen amb coeficients Haar) és entrenat amb uns pocs centenars de mostres d'un objecte concret (p.e, un cotxe o una cara), nomenats exemples positius, els quals estan escalats a exactament la mateixa dimensió (p.e, 20x20), i exemples negatius, és a dir, imatges aleatòries de la mateixa dimensió.

Un cop es té el classificador entrenat, aquest pot ser aplicat a una regió d'interès (de la mateixa dimensió que l'usat durant l'entrenament) d'una imatge d'entrada. Aquest classificador ens retornarà un “1” si la regió pareix mostrar l'objecte desitjat (p.e, cara/cotxe), i ens retornarà un “0” altrament. Per tant, podem realitzar una exploració per tota la imatge movent la finestra de cerca per tota la imatge i comprovant a cada regió de la imatge utilitzant el classificador. El classificador ha estat dissenyat per tal de ser capaç de trobar els objectes d'interès a diferents escales, el qual, és molt més eficient que re-dimensionar la imatge d'entrada per cada escalat possible de l'objecte o cara a detectar. Per tant, per trobar un objecte de dimensió desconeguda en una imatge, serà necessari explorar la imatge varis cops amb diferents dimensions de finestra d'exploració.

Aquest classificador és conegut amb el nom *"Haar Cascade Classifier"* tot i que la seva terminologia exacta seria "*cascade of boosted classifiers working with haar-like features".* Aquest arxius solen tenir una extensió "*.xml"* i és necessari que estiguin situats en el mateix directori de treball que l'algoritme on cridarem la funció HaarDetectObject().

La paraula "cascade (cascada)" en el nom del classificador significa que el classificador resultant consisteix en varis classificadors simples (etapes) que són aplicats seqüencial-ment fins que en alguna de les etapes l'objecte és considerat exemple negatiu o que un objecte passi totes les etapes i per tant serà considerat positiu. La paraula "boosted (estimulat)" es refereix a que els classificadors de cada etapa són complexos degut a que són formats a partir de classificadors bàsics utilitzant una de les quatre diferents tècniques d'estimulació *(Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost).* Els classificadors bàsics són arbres de decisió amb com a mínim dues branques o camins. Els paràmetres tipus "Haar" són els coeficients d'entrada als classificadors bàsics. Les característiques utilitzades en un classificador particular són especificades , entre d'altres, per la forma, posició entre la regió d'interès i l'escalat, proporcions entre diferents contorns, etc.

5.2- Transformada Haar Wavelet

La transformada Wavelet és un tipus especial de transformada de Fourier on representem una senyal infinita amb termes de versions desplaçades i escalades d'una ona finita (denominada ona mare o wavelet). És a dir, la transformada Wavelet d'una funció f(t) és la descomposició de f(t) en un conjunt de funcions $\psi_{s,\tau}$ (t), que formen una base i son nombrades les "Wavelets". La transformada Wavelet es defineix com:

$$W_f(s,\tau) = \int f(t)\psi^*_{s,\tau}(t)\,dt.$$

Les Wavelets són generades a partir de el desplaçament i escalat d'una mateixa funció wavelet ψ(t), nombrada la "Wavelet mare", i es defineix com:

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right),$$

On s és el factor d'escalat, i τ és el factor de desplaçament.

Les Wavelets $\psi_{s,\tau}$ (t) generades a partir de la funció mare ψ(t) tenen diferents escalats i desplaçaments, tot i que totes tenen la mateixa forma. El factor d'escalat sempre serà major que 0. Les Wavelets són estirades quant l'escalat *s<1*, i són comprimides quant *s>1*. Així, amb diferents combinacions de valors per a *s* es poden cobrir varis rangs de freqüències.

Quant la funció f(t) és continua i les wavelets són continues amb factor d'escalat i desplaçament discrets, la transformada Wavelet resulta una serie de coeficients wavelets, el qual es coneix com Series Wavelet. (Equivalent a les series de Fourier).

La funció f(t) pot ser reconstruïda a partir dels coeficients wavelets discrets Wf(s,τ), de la següent forma:

$$f(t) = A \sum_{s} \sum_{\tau} W_f(s,\tau) \psi_{s,\tau}(t),$$

On A és una constant que no depèn de f(t).

Aquestes funcions continues amb factors *s* i *τ* discrets se les denomina Wavelets discretes. Aquest factors poden ser expressats com:

$$\mathbf{s = s_0^i} \qquad \boldsymbol{\tau} \mathbf{= k\, \tau_0\, s_0^i}$$

On l'exponent *i* i la constant *k* són nombres enters, i s_0>1 representa el pas d'escalat.

El factor de desplaçament τ depèn del pas d'escalat s, per tant, adjuntant les equacions anteriors, ens queden les Wavelets discretes expressades com:

$$\psi_{i,k}(t) = s_0^{-i/2} \psi\left(s_0^{-i}\left(t - k\tau_0 s_0^{i}\right)\right) = s_0^{-i/2}\, \psi\left(s_0^{-i} t - k\tau_0\right)$$

La possibilitat de variar el factor d'escalat s permet usar wavelets de pas d'escalat molt petits per tal de concentrar l'anàlisi amb singularitats de la senyal. Quant només els detalls de la senyal són d'interès, uns pocs nivells de descomposició són necessaris. Por

tant, l'anàlisi mitjançant transformada Wavelet ens proporciona una forma molt eficient per representar senyals transitòries.

Com a exemple, podem fer una analogia entre l'anàlisi Wavelet i un microscopi. Així, el pas d'escalat s_0 correspon a l'augment o resolució del microscopi i el factor de desplaçament τ correspon a la ubicació on es fa la observació amb el microscopi. Si volem observar petits detalls, l'augment i la resolució del microscopi haurien de ser grans, el qual ens correspon amb una *i* gran i negativa. Això ens dona lloc a una funció Wavelet molt concentrada, i a passos de desplaçament petits. Per a valors de *i* grans i positius, la Wavelet s'expandeix i els passos de desplaçament són adaptats a aquesta amplitud.

Elegint adequadament $\psi(t)$ i els paràmetres s_0, τ_0, es possible aconseguir que les funcions $\psi_{s,\tau}(t)$ formin una base ortonormal de $\mathbf{L^2(R)}$ com és el cas de $s_0=2$ i $\tau_0 = 1$.

D'aquesta forma, si les funcions wavelets discretes formen una base ortonormal, una funció f(t) de suport finit pot ser reconstruïda com una suma dels coeficoents wavletes discrets Wf(s, τ) multiplicats per les funcions de la base, és a dir:

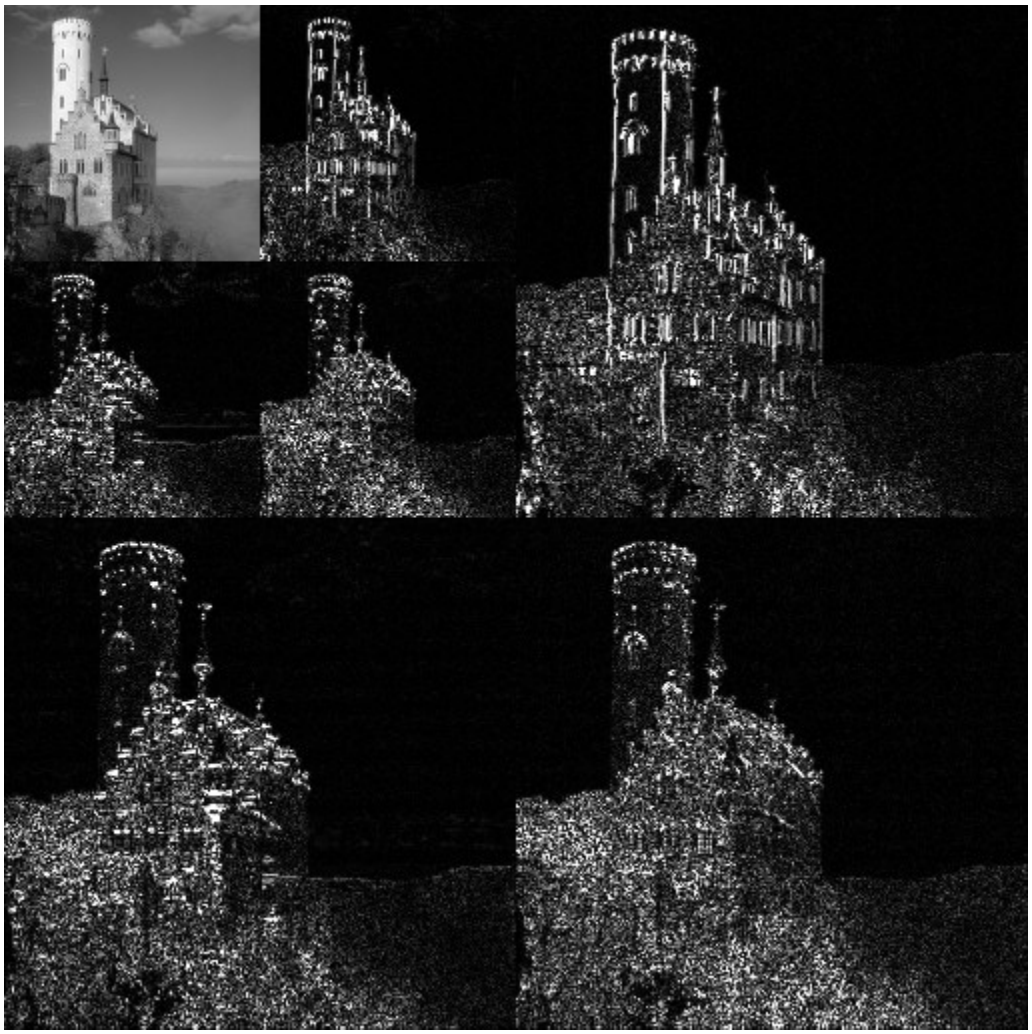
$$f(t)=\sum_{s}\sum_{\tau}W_f(s,\tau)\psi_{s,\tau}(t).$$

Una descomposició wavelet ortonormal no posseeix informació redundant i representa la senyal en forma unívoca. Una base wavelet ortonormal es possible amb wavelets amb factors de desplaçament i escalat discrets. Per tant, per aquestes funcions wavelets discretes ortogonals, els productes interns són iguals a zero:

$$\int \psi^*_{i,k}(t)\psi_{m,n}(t)\,dt=\begin{cases}1 & \text{si i=m i K=n}\\ 0 & \text{altrament}\end{cases}$$

El 1986, Meyer i Mallat demostraren que la descomposició i reconstrucció wavelet ortonormal podrien ser implementades en el marc d'anàlisi multi-resolució de senyals.

Per últim, mostrem amb una imatge on podem apreciar a simple vista com mitjançant diferents transformades wavelet podem extreure diferent tipus d'informació d'una imatge:



Com es pot apreciar, la transformada wavelet ens remarca els canvis en les imatges.

La transformada Haar, és un cas particular de la transformada Wavelet on la senyal base esta definida per:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2, \\ -1 & 1/2 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Amb aquest senyal base i uns valors adequats de s_0 i τ_0 podem analitzar la imatge d'entrada per tal de trobar els paràmetres tipus haar i comprovar si aquests passen les etapes del nostre classificador, és a dir, que es compleixin les condicions necessàries per ser considerat una mostra positiva d'un objecte o cara.

5.3- Llibreries OpenCV

OpenCv són unes llibreries lliures de visió artificial desenvolupades inicial-ment per Intel. Des de l'aparició de la seva primera versió alfa en el mes de genes de 1999, s'ha utilitzat en infinitat d'aplicacions. Des de sistemes de seguretat amb detacció de moviment, fins a aplicacions de control de processos on es requereix reconeixement d'objectes. Això es deu a que la seva publicació es va fer sota llicència BSD, que permet que siguen utilitzades lliurement per propòsits comercials i de investigació amb les condicions expressades en elles.

Open CV és una multi plataforma, existint versions per a GNU/Linus, Mac OS X i Windows. Conté més de 500 funcions que engloben una gran gama d'àrees en el procés de visió, com reconeixement d'objectes, calibrat de càmeres, visió estereoscòpica i visió robòtica.

El projecte pretén proporcionar un entorn de desenvolupament fàcil d'utilitzar i altament eficient. Això s'ha realitzat, realitzant la seva programació amb llenguatges C i C++ optimitzats, aprofitant a més, les capacitats que ens aporten els processador multi nucli. OpenCv pot també utilitzar els sistemes de primitives de rendiment integrades en Intel, un conjunt de rutines de baix nivell específiques per processadors Intel (IPP).

Aquestes llibreries poden ser instal·lades des de els reposi-toris d'ubuntu. Tecleglem les següents instruccions:

maik@maik-laptop:~$ sudo apt-get install python2.6-opencv

I a partir d'aquí ja podem començar a utilitzar totes les seves funcions, que no son poques.

5.4- Algoritme Face Detect

A continuació mostrem el codi i els comentaris de l'algoritme “Facedetect.py” adjunt al directori “Algoritmes” de l'apèndix.

Primer que tot és important que per executar s'ha d'indicar el nom de la imatge d'entrada que volem analitzar per trobar cares. La forma correcta de cridar l'algoritme és amb instruccions del tipus:

python facedetec.py lena.jpg

python facedetec.py lena2.jpg

A més, és molt important que l'arxiu "haarcascade_frontalface_alt.xml" estigui en el directori de treball. En cas contrari l'algoritme de detecció de cares no disposarà de cap arxiu que contingui el classificador amb els paràmetres Haar. L'algoritme també ofereix l'opció de introduir també el nom del classificador desitjat. Per exemple:

python facedetec.py lena2.jpg haarcascade_frontalface_alt2.xml

Pel que fa a les constants de l'algoritme, *min_neighbors* fa referència a “l'exigència” que té l'algoritme per decidir si es tracta d'una cara o no. És a dir, amb un *min_neighbors* de valor baix, és possible que l'algoritme ens senyali cares on no n'hi han. En canvi, amb un *min_neighbors* alt, l'algoritme serà més *“exigent” a* l'hora de considerar-ho cara, i per tant, ens detectarà menys cares falses.

A continuació, comentem el codi per poder entendre el programa de manera sintetitzada:

```
import sys
from opencv.cv import *
from opencv.highgui import *

# Variables generals
cascade = None
storage = cvCreateMemStorage(0)
cascade_name = "haarcascade_frontalface_alt.xml"
```

```
input_name = "..lena.jpg"
# Parametres per a la deteccio Haar
min_size = cvSize(20,20)  #Tamany minim de la cara
image_scale = 1.3         # Escalat espacial de la imatge (per reduir cost computacional)
haar_scale = 1.2
min_neighbors = 2                 # Minim coincidencies
haar_flags = 0

#Definim una funcio per detectar cares
def detect_and_draw( img ):

        # Creem una imatge amb B/N buida
        gray = cvCreateImage( cvSize(img.width,img.height), 8, 1 )

        # Creem una imatge amb B/N buida i reduida per tal que el cost computacional
        # sigui mes petit
        small_img = cvCreateImage((cvRound(img.width/image_scale), cvRound
        (img.height/image_scale)), 8, 1 )

        # Copiem les dades de la imatge convertides a B/N
        cvCvtColor( img, gray, CV_BGR2GRAY )

        # Copiem la imatge redimensionada gray a small_img amb interpolacio lineal
        cvResize( gray, small_img, CV_INTER_LINEAR )

        #Equalitzem la imatge per facilitar l'anàlisi
        cvEqualizeHist( small_img, small_img )

        #Limpiem els buffers de memoria
        cvClearMemStorage( storage )

        if( cascade ): #Si hem pogut carregar l'arxiu Haarcascade_frontalface...

        #Apliquem l'algoritme de deteccio de cares
```

```
        faces = cvHaarDetectObjects( small_img, cascade, storage,haar_scale,
        min_neighbors, haar_flags, min_size )

        # En cas de cares detectades
        if faces:

                #Per a totes les cares:
                for face_rect in faces:
                # La imatge d'entrada a cvHaarDetectObjects estava redimensionada,
                # per tant, hem de reescalar els punts del rectangle abans de 'pintar-los' a
                # la imatge
                        pt1=cvPoint(int(face_rect.x*image_scale),
                                                int(face_rect.y*image_scale))
                        pt2=cvPoint( int((face_rect.x+face_rect.width)*image_scale),
                                        int((face_rect.y+face_rect.height)*image_scale) )
                        # Enmarquem la cara
                        cvRectangle( img, pt1, pt2, CV_RGB(255,0,0), 3, 8, 0 )

        #Mostrem el resultat per pantalla
        cvShowImage( "result", img )

# Petita aplicacio per poder cridar el programa amb diferents imatges d'entrada, i
# diferents plantilles de deteccio Haar*.xml

if __name__ == '__main__':

    # Si la llargada de la comanda te mes d'un argument, el primer es el nom
    # de l'arxiu Haar, i el segon el nom de la imatge
    if len(sys.argv) > 1:

        if sys.argv[1].startswith("--cascade="):
            cascade_name = sys.argv[1][ len("--cascade="): ]
            if len(sys.argv) > 2:
                input_name = sys.argv[2]
```

```
        # Si no, ens quedem amb el nom de la imatge o mostrem informacio per el cas
        # de help
    elif sys.argv[1] == "--help" or sys.argv[1] == "-h":
                print "Usage: facedetect --cascade=\"<cascade_path>\" [filename|
                        camera_index]\n"
                sys.exit(-1)

    else:
        input_name = sys.argv[1]

#Carreguem l'arxiu plantilla Haar
cascade = cvLoadHaarClassifierCascade( cascade_name, cvSize(1,1) )

if not cascade:
    print "ERROR: Could not load classifier cascade"
    sys.exit(-1)

image = cvLoadImage( input_name, 1 )

 if image:
          #Cridem la funcio detecta cares
        detect_and_draw( image )
        cvWaitKey(0)

#Destruim la finestra de la imatge
cvDestroyWindow("result")
```

6- LABJACKPYTHON

6.1- Instal·lació mòdul

LabjackPython és un mòdul dissenyat per a Python per poder comunicar-nos amb dispositius Labjack mitjançant un llenguatge d'alt nivell com és el cas del Python. Aquesta plataforma ha estat adaptada per als tres sistemes operatius principals. En el cas de Windows, utilitzarem els UD Drivers (http://labjack.com/support/windows-ud) i en el cas de Linux o Mac OS X, utilitzarem els Exodrivers (http://labjack.com/support/linux-and-mac-os-x-drivers).

En el cas de Ubuntu 10.04 o superior, podem instal·lar el mòdul simplement teclejant una serie d'instruccions amb la consola d'aplicacions (ctrl+alt+t) comentades a continuació:

```
$ sudo apt-get install build-essential     # Instal·lem les llibreries build-essential
$ sudo apt-get install libusb-1.0-0-dev    # Llibreries de programador per ports usb
$ sudo apt-get install git-core             # Instal·lador de paquets git-core
$ git clone git://github.com/labjack/exodriver.git  # Ens descarreguem l'exodriver
$ cd exodriver/liblabjackusb/    #Ens situem en el directori on l'hem descarregat
$ make                            # Preparem per compilar
$ sudo make install               # Compilem els drivers
$ cd ..
$ sudo cp 10-labjack.rules /etc/udev/rules.d/.   # Copiem el directori rules.d
$ sudo udevadm control --reload-rules            # Exodriver instal·lat
$ cd
# Descarreguem el mòdul LabJackPython
$ git clone git://github.com/labjack/LabJackPython.git
$ cd LabJackPython/      # Ens situem al directori
$ sudo python setup.py install   # Instal·lem el mòdul
```

Per tal de comprovar que tot s'hagi instal·lat correctament, podem teclejar les següents instruccions:

```
$ python
>>> import u3
>>> d = u3.U3()
>>> d.configU3()
```

Si no ens dona cap error a l'hora d'executar aquestes instruccions, significarà que s'ha instal·lat tot correctament.

6.2- Funcionament mòdul

En el cas d'aquest projecte, hem estat utilitzant el model de LabJack U3. Tot i això, la majoria de codi mostrat a continuació ens serveix per a altres dispositius com el U6 o U9 aplicant petites variacions.

6.2.1- Obrir LabJack

Per obrir l'aparell (en el nostre cas LabJack U3), hem d'importar les llibreries u3 i crear una variable de la classe u3.U3(), és a dir:

```
>>> import u3
>>> d = u3.U3()
```

El constructor de la classe U3 provarà automàticament d'obrir el primer dispositiu U3 que trobi connectat a l'ordenador. Aquest sistema és bo en cas de tenir connectat només un dispositiu.

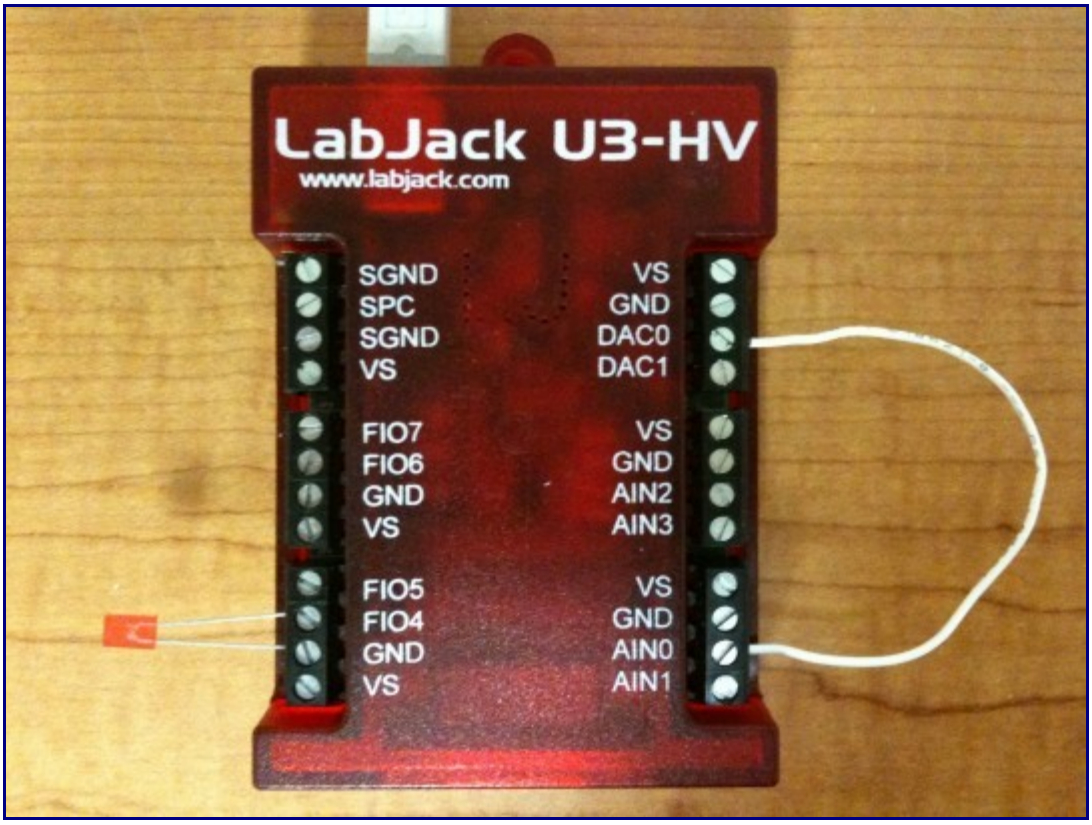
A continuació, és important remarcar que hi han dues formes de treballar amb LabJackPython. La primera, consisteix en treballar amb instruccions d'alt nivell amb registres, la segona, treballar directament amb instruccions de baix nivell. En el cas de la nostra aplicació hem programat utilitzant únicament instruccions de baix nivell degut a que només necessitem sortides digitals.

6.2.2- Operacions amb registres

Per tal de poder-ho explicar d'una forma ràpida de comprovar el resultat, podem realitzar el següent muntatge.

Passem un cable del convertidor DAC0 a l'entrada analògica AIN0. Per altra banda, connectem un LED entre la sortida FIO4 i la massa GND (totes les masses de l'aparell són comunes).

En la següent imatge podem veure el senzill muntatge per fer la prova.



Habilitem el convertidor DAC0 a 1.5V, i llegim el valor de l'entrada AIN0:

```
>>> DAC0_REGISTER = 5000
>>> d.writeRegister(DAC0_REGISTER, 1.5) # Habilitem DAC0 a 1.5 V
1.5
>>> AIN0_REGISTER = 0
```

```
>>> d.readRegister(AIN0_REGISTER)     # Llegim AIN0
1.5252180099487305
```

En aquest exemple, estem utilitzant LabJack Modbus per saber a quin registre pertany cada entrada/sortida desitjada. Aquests registres poden ser consultats al següent link ofert per el fabricant: http://labjack.com/support/modbus. Busquem el full d'Exel titulat “LabJack ModBus Map” situat a baix de tot i comprovem, per exemple, el numero de registre corresponent a DAC1 és el 5002.

A continuació, encenem el LED activant FIO4 a nivell alt i l'apaguem amb nivell baix.

```
>>> FIO4_STATE_REGISTER = 6004
>>> d.writeRegister(FIO4_STATE_REGISTER, 1) # Activar FIO4
1
>>> d.writeRegister(FIO4_STATE_REGISTER, 0) #  Desactivar FIO4
0
```

El valor de FIO4_STATE_REGISTER també ha estat consultat al mapa Modbus. Si el LED està connectat amb la polarització correcta, s'hauria d'encendre/apagar en funció de l'estat de FIO4.

Per utilitzar FIO4 com a entrada digital, primer l'hem de configurar com a entrada, i després podem llegir el seu estat.

```
>>> FIO4_DIR_REGISTER = 6104
>>> d.writeRegister(FIO4_DIR_REGISTER, 0)  # Configurem FIO4 com a entrada digital
0
>>> d.readRegister(FIO4_STATE_REGISTER)    # Llegim estat FIO4
0
>>> # Desconnectar el LED
>>> d.readRegister(FIO4_STATE_REGISTER)    #  Llegim estat FIO4
1
```

Quant el LED està connectat, llegim un '0'. Quant el LED està desconnectat, la

resistència “pull-up” de l'entrada causa un estat '1'.

Per a més informació sobre com treballar amb ModBus, llegir registres amb paral·lel, o entrar amb el protocol entre ModBus i el port USB podeu consultar-ho a la web del fabricant mostrada anteriorment.

6.2.3- Operacions amb instruccions de baix nivell

Utilitzarem el mateix muntatge que en l'apartat anterior per tal de comprovar el correcte funcionament de les instruccions de baix nivell.

A continuació es mostren les principals funcions de la classe U3 per poder utilitzar els convertidors, entrades analògiques i E/S digitals.

La primera funció a explicar seria *getFeedback().* Per a cada comanda diferent, existeix una classe corresponent en l'arxiu u3.py. En el següent exemple es mostra com utilitzar els convertidors i entrades analògiques amb una resolució de 8 bits.

```
>>> DAC0_VALUE = d.voltageToDACBits(1.5, dacNumber = 0, is16Bits = False)
>>> d.getFeedback(u3.DAC0_8(DAC0_VALUE))       # Activar DAC0 a 1.5 V
```

Com bé sabem, un convertidor de 8 bits pot representar 255 nivells de tensions. En el cas de LabJack aquestes van entre 0 i 4.95 V. La variable DAC0_VALUE ens representa el valor amb bits que desitgem que ens representi el convertidor. Per tant la funció d.voltageToDACBits() ens serveix per “estalviar-nos” fer el càlcul manualment.

En el cas de llegir una entrada analògica, també s'utilitza la instrucció getFeedback(). Tot i això, aquesta ens retorna el valor amb format 16 bit sense signe, per tant, serà necessari convertir-ho a nivell de voltatge analògic.

```
>>> ain0bits, = d.getFeedback(u3.AIN(0)) # Llegir el valor dels bits de AIN0
>>> print ain0bits
37584
# Els convertim a valor analògic
>>> ainValue = d.binaryToCalibratedAnalogVoltage(ain0bits, isLowVoltage = False)
```

```
>>> print ainValue
1.501376
```

La coma darrere la variable ain0bits és deguda a que la funció getFeedback() ens retorna les dades amb format llista, per tant, és necessari utilitzar la coma. La funció binaryToCalibratedAnalogVoltage() ja desempaqueta automàticament el primer valor de la llista.

Degut a que l'operació de llegir una entrada analògica és molt comuna, la classe U3 ens ofereix la funció d.getAIN() que ens crida les dues funcions anteriors amb una sola instrucció. És a dir:

```
>>> ainValue = d.getAIN(0)  # Llegim AIN0 amb una sola instrucció
>>> print ainValue
1.501376
```

Configurar l'estat d'una E/S digital, també pot ser fet mitjançant les comandes *getFeedback()*. Les més rellevants són *BitStateRead(), BitStateWrite(), BitDirRead() i BitDirWrite().*

Primer que tot, comprovem quines entrades estan configurades com a analògiques i quines com a digitals.

```
# Comprovació FIOs
>>> configDict = d.configIO()
>>> configDict["FIOAnalog"]
63
# (63=111111) Per tant serien del (0-5) analògiques i la resta digitals.
# Configurem els FIO (0-3) a analògic (15 = 1111 binari) i la resta digitals
>>> d.configIO(FIOAnalog = 15)
>>> d.getFeedback(u3.BitDirWrite(4, 1))   # Configurar FIO4 a sortida digital
[None]
>>> d.getFeedback(u3.BitStateWrite(4, 1)) # FIO4 a nivell alt
[None]
>>> d.getFeedback(u3.BitStateWrite(4, 0)) # FIO4 a nivell baix
[None]
```

La funció getFeedback() també accepta una llista de comandes feedbacks, per tant, podem cridar-la amb múltiples arguments simultàneament. Per exemple, configurem FIO4 com a sortida digital i la posem a nivell alt.

```
>>> outputDirCmd = u3.BitDirWrite(4, 1)
>>> outputHighCmd = u3.BitStateWrite(4, 1)
>>> cmdList = [outputDirCmd, outputHighCmd]
>>> d.getFeedback(cmdList) # Ens passa una llista de comandes
[None, None]
>>> d.getFeedback(outputDirCmd, outputHighCmd) # O passar les comandes com a arguments
[None, None]
```

Degut a que és una operació que s'utilitzarà molts cops, les funcions de la classe U3 ens ofereixen la instrucció *d.setFIOStat()* per poder-ho fer directament.

```
>>> d.setFIOState(4, 1) # Posa FIO4 a nivell alt amb una sola instrucció
>>> d.setFIOState(4, 0) # Posa FIO4 a nivell alt amb una sola instrucció
```

Quant passem múltiples comandes de feedback a la funció getFeedback(), aquestes són processades seqüencial-ment. A continuació mostrem un exemple on comprovem l'estat de FIO4 (configurat com a sortida anteriorment), configurar-lo com a entrada i llegir el valor un altre cop.

```
# COMANDES MÚLTIPLES AMB GETFEEDBACK()
#    Llegir FIO4's
#    Configurar FIO4 com a entrada
#    Llegir FIO4 de nou
>>> d.getFeedback(u3.BitDirRead(4), u3.BitDirWrite(4, 0), u3.BitDirRead(4))
[1, None, 0]
```

Ara que FIO4 està configurada com a entrada, podem llegir el seu estat.

```
>>> d.getFeedback(u3.BitStateRead(4)) # Llegir estat FIO4
[0]
# Desconnectar el LED
```

```
>>> d.getFeedback(u3.BitStateRead(4))
[1]
# Connectar el LED
>>> d.getFeedback(u3.BitStateRead(4))
[0]
```

Aquesta operació pot ser efectuada de manera més còmoda, i sense necessitat de configurar FIO4 com a entrada amb l'instrucció getFIOState():

```
>>> d.getFIOState(4)
0
```

En aquesta memòria s'ha mostrat com realitzar operacions bàsiques sobre els convertidors, entrades analògiques i E/S digitals. La majoria d'instruccions de baix nivell són comandes de Feedback, per tant, poden ser passades com a instruccions getFeedBack(), la qual és una funció realment flexible que ens proporciona una gran llibertat a l'hora de programar. Per a més informació es pot consultar a la web del fabricant.

Per últim, mostrarem com configurar el nostre aparell mitjançant instruccions de baix nivell. Primer que tot utilitzarem la comenda configU3() per extreure la informació més important del nostre aparell.

```
>>> print d.configU3()
{
   'LocalID': 5,
   'SerialNumber': 32003XXXX,
   'DAC1Enable': 1,
   'EIODirection': 0,
   'DeviceName': 'U3-HV',
   ...
}
```

La funció configU3() configura el nostre aparell a els seus valors per defecte. Per canviar aquesta configuració (únicament per a la sessió actual), podem utilitzar

configIO(). Per exemple, per activar els 5 primer FIOs com a entrades analògiques, cridem la funció amb els següents arguments:

```
>>> d.configIO(FIOAnalog = 0x1F)
{'DAC1Enable': 0, 'FIOAnalog': 31, 'EIOAnalog': 0, 'TimerCounterConfig': 64}
```

El valor 0x1F és una màscara on els 5 bits més baixos estan posats a 1 mentre que la resta estan a 0. Si volem que els canvis no es s'esborrin al acabar la sessió, serà necessari utilitzar l'instrucció configU3() en lloc de configIO():

```
>>> d.configU3(FIOAnalog = 0x1F)
{...
'FIOAnalog': 31,
...
}
>>> d.close()
>>> # Desconnectar l'aparell i tornar-lo a connectar
>>> d.open()
>>> d.configU3()
{...
'FIOAnalog': 31,
...
}
```

Com podem comprovar, els canvis s'han guardat correctament.

En cas de necessitar més documentació sobre comandes de baix nivell amb LabJackPython , es pot consultar els exemples i fonts de codi que ofereix el fabricant. Un dels arxius més complet seria el “u3.py module”. Aquest pot ser trobat al següent link:

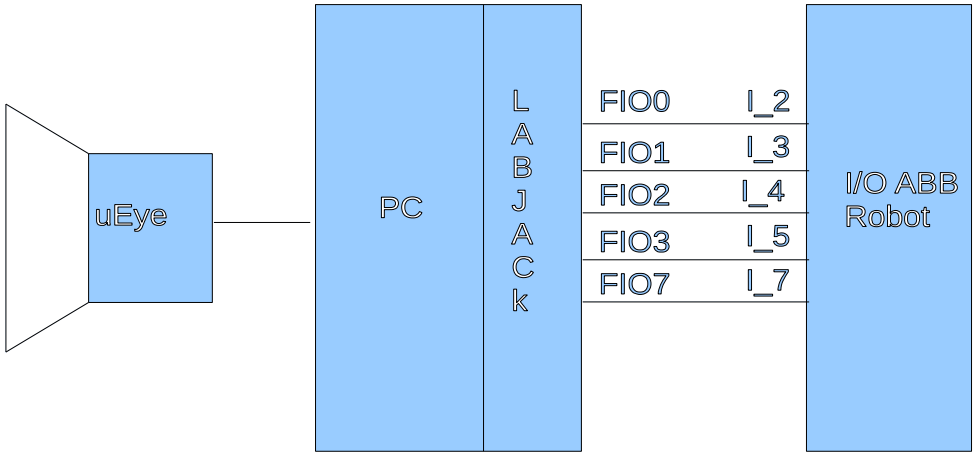
https://github.com/labjack/LabJackPython/blob/master/src/u3.py

7- ALGORITME PYTHON

Anteriorment, s'ha mostrat com instal·lar i utilitzar la càmera web uEye, com instal·lar i canviar el valor dels registres de l'adaptador USB-LabJack, les constants, sintaxis i funcions de l'algoritme de detecció de cares, els petits detalls sobre funcionament dels algoritmes per poder ajustar-los a les nostres necessitats, etc.

Abans de posar-nos en l'explicació del codi, és de vital importància entendre exactament que és el que necessitem que faci l'algoritme. Com s'ha explicat en l'apartat "Visió Global", aquest projecte consisteix en que el robot s'orienti automàticament en direcció a una cara humana. Amb altres paraules, l'objectiu del robot és mou-res per tal que la cara detectada es vegi per el mig de la pantalla. Per tant, si la cara està a la dreta de la imatge, el robot tindrà que girar en sentit horari, si la cara està a dalt, el robot interpolarà linealment en l'eix z (dalt), etc.

En el cas del nostre robot, l'única manera de comunicar-se entre l'ordinador portàtil i el robot és mitjançant les entrades i sortides digitals del robot. Això és degut a que la uVic no disposa actualment de la targeta per comunicar-se amb altres PCs i encara no s'ha pogut desenvolupar un protocol de comunicacions mitjançant TCP/IP. Per tant, en el nostre cas, el robot variarà la posició amb petits increments en funció de el que es rebi a les entrades digitals. Aquest és un esquema del nostre muntatge:



Cada sortida FIO dóna una ordre de desplaçament, FIO0 representa rotació en sentit horari, FIO1 rotació en sentit antihorari, FIO2 interpolació lineal +z (dalt), FIO3

interpolació lineal -z (baix) i FIO7 indica la presència de cara.

Existeix una comanda especial codificada com FIO2 i FIO3 activats a 1. Aquesta significa que les dimensions de la cara detectada són grans, és a dir, que la cara detectada està a prop de la càmera. En aquest cas, el robot s'encongira interpolant linealment per tal de poder seguir gravant la cara sense que aquesta es desenfoqui degut a que la distància és massa curta.

Per tant, el nostre algoritme realitza indefinidament la següent cadena d'accions: Capturar fotograma, explorar l'existència de cara, modificar l'estat dels FIO en funció de la posició on es troba la cara, tornar a repetir el proces.

A continuació comentem el codi de l'algoritme. L'explicació del significat de les constants d'aquest algoritme es troba al final d'aquest apartat.

La funció *detect_and_draw* analitza la el fotograma en busca de cares. Si es troba una cara, retornem el valor dels dos punts del rectangle que envolta la cara. En cas de no haver detecció de cara, la funció ens retorna els dos punts a (0,0).

Aquest dos punts són un dels paràmetres d'entrada necessàries per la funció *lab_Jack.* Aquesta s'encarrega de canviar els valors dels FIO en funció de la posició i el tamany de la cara.

```
import sys
import ueye
import numpy
import Image
import u3
from opencv.cv import *
from opencv.highgui import *

# Constants Glovals
cascade = None
storage = cvCreateMemStorage(0)
cascade_name = "haarcascade_frontalface_alt.xml"
```

```
margex = 100
margex2= 120
margey = 45
cara_max = 200
cara_min = 120

# Parameters for haar detection
min_size = cvSize(50,50)
image_scale = 1.3
haar_scale = 1.2
min_neighbors = 35
haar_flags = CV_HAAR_DO_CANNY_PRUNING

# Definim la funcio detecta cares
def detect_and_draw( img ):

    gray = cvCreateImage( cvSize(img.width,img.height), 8, 1 )
    small_img = cvCreateImage((cvRound(img.width/image_scale),
                               cvRound (img.height/image_scale)), 8, 1 )

    cvCvtColor( img, gray, CV_BGR2GRAY )

    cvResize( gray, small_img, CV_INTER_LINEAR )

    cvEqualizeHist( small_img, small_img )

    cvClearMemStorage( storage )

    #Posem els dos punts a 0.
    pt1= cvPoint(0,0)
    pt2= cvPoint(0,0)
    if( cascade ):
        faces = cvHaarDetectObjects( small_img, cascade, storage,
```

```
                                haar_scale, min_neighbors, haar_flags, min_size )

        # Si hi ha deteccio de cares
      if faces:
         for face_rect in faces:
            # Calculem els punts del rectangle
            pt1 = cvPoint( int(face_rect.x*image_scale), int(face_rect.y*image_scale))
            pt2 = cvPoint( int((face_rect.x+face_rect.width)*image_scale),
                        int((face_rect.y+face_rect.height)*image_scale) )


               #Pintem la cara
               cvRectangle( img, pt1, pt2, CV_RGB(255,0,0), 3, 8, 0 )

   #Mostrem per pantalla
   cvShowImage( "Example1", img )
   # Retornem els dos punts de l'ultima deteccio
   return pt1,pt2



# Definim la funcio per indicar la direccio de desplasament al robot.
# S'ha de tenir amb compte que un '0' a la sortida del labjack
# significara un '1' a l'entrada del robot degut al montatge amb els transistors.
# Necessari per problemes d'inisialitzacio de variables.
marge= margex

def lab_jack( pt1,pt2,tamany,encongit ):

    # Primer decidim en quin estat ens trobem mitjansant una comparacio amb histeresis
   # de l'altura de la cara
   if (pt2.y-pt1.y)>cara_max:
        encongit = True

   if (pt2.y-pt1.y)<cara_min:
```

```
        encongit = False

    if encongit==False:
        marge=margex
    else:
        marge=margex2

     # Comencem amb estat normal
     # Calculem el centre de la cara
    punt= cvPoint( pt1.x+int((pt2.x-pt1.x)/2),pt1.y+int((pt2.y-pt1.y)/2))

        #Eix X
    # Si el centre de la cara esta a la dreta de la imatge activem horari
    if punt.x>((tamany[0]/2)+marge):
       d.setFIOState(0, state = 0)
        d.setFIOState(1, state = 1) # Per seguretat

    # Si el centre de la cara esta a la esquerra de la imatge activem antihorari
    if punt.x<((tamany[0]/2)-marge):
        d.setFIOState(1, state = 0)
        d.setFIOState(0, state = 1) # Per seguretat

    # Si estem centrats en la dimensio x desactivem horari i antihorari
    if (punt.x<=((tamany[0]/2)+marge))and (punt.x>=((tamany[0]/2)-marge)):
        d.setFIOState(1, state = 1)
        d.setFIOState(0, state = 1)

        #Eix y
    # Els bits que representen l'eix y, no poden ser cambiats si estem en estat encongit.
    # Per tant, nomes els canviarem quant no estem en estat robot encongit

    # Si estem en estat normal, orientem el robot
    if encongit==False :
            # Si el centre de la cara esta a dalt de la imatge activem dalt
```

```
            if punt.y<((tamany[1]/2)-margey):
                d.setFIOState(2, state = 0)
                d.setFIOState(3, state = 1) # Per seguretat


            # Si el centre de la cara esta a baix de la imatge activem baix
            if punt.y>((tamany[1]/2)+margey):
                d.setFIOState(3, state = 0)
                d.setFIOState(2, state = 1) # Per seguretat

            # Si estem centrats en la dimensio y desactivem dalt i baix
            if (punt.y>=((tamany[1]/2)-margey))and (punt.y<=((tamany[1]/2)+margey)):
                d.setFIOState(3, state = 1)
                d.setFIOState(2, state = 1)

    else:
        # Cas especial
        # Encongim el robot
        d.setFIOState(3, state = 0)
        d.setFIOState(2, state = 0)

    # Un cop configurat el desplasament desitjat, donem l'ordre al robot
    d.setFIOState(7, state = 0)
    return encongit

#_________________ALGORITME_PRINCIPAL_________________

if __name__ == '__main__':

        #Obrim el labjack
        d = u3.U3()
        # Capturem una imatge de l'uEye i la transformem a format opencv
        a=ueye.Cam()
        b=a.GrabImage()
```

```
    im = Image.fromarray(b)
    tamany=im.size
    cv_im = cvCreateImage(tamany, 8, 3)
    cv_im.imageData = im.tostring()

    cvNamedWindow("Example1", CV_WINDOW_AUTOSIZE)   #Creem una finestra per
                                                    # mostrar la imatge
    cvShowImage( "Example1" , cv_im )

    # Carguem l'arxiu Haar*.xml
    cascade = cvLoadHaarClassifierCascade( cascade_name, cvSize(1,1) )
    encongit = False

    # Interrumpim el programa fins que l'usuari premi una tecla
    cvWaitKey(0)

encongit = False # Comencem amb estat normal
    # creem un bucle on s'anira actualitzant i mostrant en la mateixa finestra la
    # imatge de la webcam , i es cridara la funcio lab_jack
    # cada 1000/25 ms (25 fps)
    while True:

        # Capturem una imatge de l'uEye, la transformem a format opencv i cridem
        # la funcio detecta cares
        b=a.GrabImage()
        im = Image.fromarray(b)
        cv_im.imageData = im.tostring()
        punt1,punt2=detect_and_draw( cv_im )

        # Si es detecta cara
        if (punt1.x+punt2.x)>0:
            # Cridem la funcio lab_jack
            encongit=lab_jack( punt1,punt2,tamany,encongit )
        else:
```

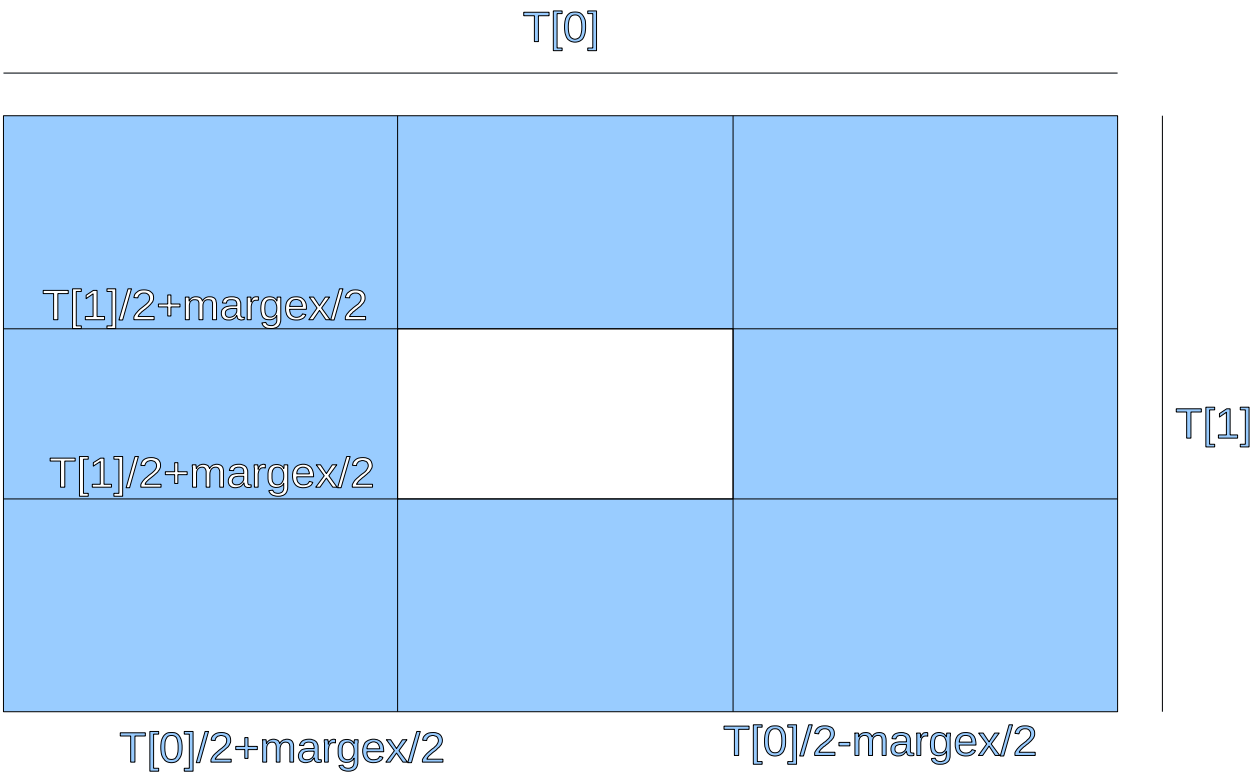
```
                d.setFIOState(7, state = 1)  #Set FIO7 to state '0' (el robot no es
                                             # moura)

        cvWaitKey(1000/25)

    d.close()
    cvDestroyWindow( "Example1" )
```

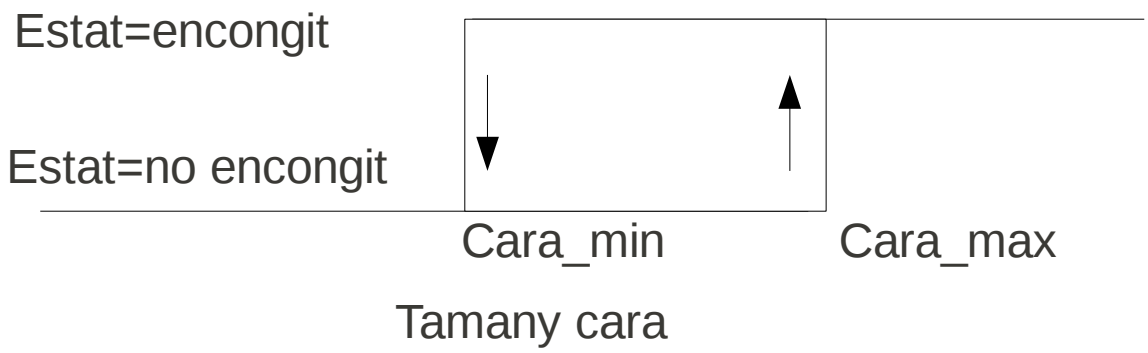
Les constants definides al principi de l'algoritme serveixen per solucionar els problemes que em van sortir al provar l'algoritme en un robot real.

El primer problema que sortia era quant la cara detectada estava al mig de la imatge. El robot entrava amb una ressonància infinita. Aquesta ressonància era deguda a que la càmera detectava, per exemple, la cara 20 píxels a l'esquerra, el robot girava un grau en sentit antihorari. Al finalitzar el desplaçament, la càmera detecta la cara 25 píxels a la dreta, el robot gira un grau en sentit horari. La càmera tornarà a detectar la cara 20 píxels a l'esquerra i es repetirà el procés indefinidament. Per solucionar-ho hem de definir cada dimensió amb tres parts en lloc de dos. És a dir, esquerra, centre i dreta. En el següent dibuix s'entén el significat físic de les constants marge.



Evidentment, quant estem en la zona blanca no mourem el robot. El valor de margex2 és el marge de comparació amb l'eix x quant el robot es troba en estat encongit. Això és degut a que quant el robot estava amb estat encongit ens apareixien unes ressonàncies que s'han solucionat utilitzant un marge diferent.

El segon problema que em va sortir provant l'algoritme amb el robot va ser la transició entre estat encongit i estat no encongit. El problema consistia en que al apropar-nos al robot, el tamany de la cara passava el llindar i el robot s'encongia. Al encongir-se el robot, la càmera retrocedia i el tamany de la cara tornava a passar el llindar i el robot tornava a l'estat normal. I així indefinidament. Per solucionar el problema, l'algoritme decideix l'estat del robot mitjançant una comparació d'histèresis en lloc d'una comparació normal. D'aquesta manera, solucionem el problema. Aquest és el funcionament d'aquesta comparació:



El que ens representa aquest diagrama és que els nivells de comparació depenen de quin estat estem. Per exemple, quant estem en estat encongit, per canviar d'estat hem de passar inferiorment el llindar Cara_min. Quant estem en estat no encongit, per canviar d'estat hem de passar el llindar Cara_max. D'aquesta manera queda solucionat el problema comentat anteriorment.

8- ROBOT ABB

8.1- Introducció a la robòtica

Un robot industrial és una màquina electrònica manipuladora multi-funcional, re-programable, capaç de moure càrregues, matèries, peces, eines o dispositius especials, a través de moviments i trajectòries variables, programades per realitzar diferents tasques.

Cada robot està dissenyat i construït per poder realitzar una sèrie de tasques específiques. Els principals camps més destacats d'aplicació de la robòtica són:

1. Assistència, on el robot realitza operacions amb relació directa amb l'home amb un context variable i “personal”. Dins aquest camp es pot aplicar el seu ús en pròtesis, ortosis, rehabilitació, ...
2. Exploració, on el robot desenvolupa tasques del tipus informatiu (cerca, localització, supervisió,...) o tasques físiques (extracció, presa de mostres, recuperació d'objectes...). Aquestes tasques es poden desenvolupar en ambients marítims, terrestres, militars, de l'espai, o en mineria.
3. Producció, àmbit de la robòtica industrial, on el robot es troba en un entorn que és pràcticament invariant. En aquest àmbit els robots solen estar dissenyats per realitzar operacions de manipulació d'objectes (càrrega, descàrrega, manutenció,...), fabricació (tall, deformació, fresat, recobriment, ...), assemblat, (soldadura, inserció, ...), verificació (mesura, inspecció,...).

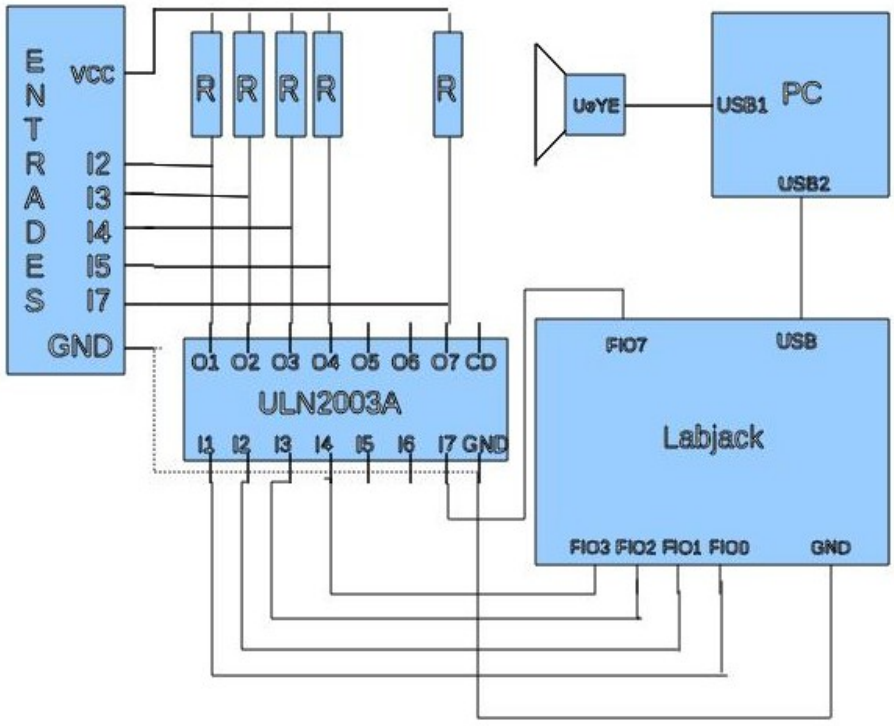
Es defineixen els atributs d'un robot, com cada una de les propietats o qualitats que aquest pot tenir i que el diferencien dels altres. Com a atributs es poden trobar:

1. La mobilitat, la qual està altament associada a la forma del robot.
2. La governabilitat, és la capacitat que té un robot per ser controlat des de fonts exteriors.
3. Autonomia és la capacitat de que disposa un robot per captar, conèixer i/o reconèixer el seu entorn. La capacitat de prendre decisions o planificar tasques per sí sol. I La capacitat d'adaptació, cooperació i aprenentatge.
4. La polivalència és la capacitat que té un robot per portar a terme diferents tasques amb més o menys diferència.

8.2- Connexions

La placa d'entrades/sortides del *robot ABB* de la Universitat de Vic treballa amb senyals digitals de 0-15Volts mentre que el *LabJack* treballa amb voltatges de l'ordre de 0-3,5 Volts. És per això que necessitem dissenyar un circuit per adaptar els nivells de voltatge i aïllar (mitjançant transistors) la font d'alimentació de l'ordenador portàtil amb la font d'alimentació de la placa d'entrades/sortides.

En el nostre cas s'ha muntat ajudant-nos d'un driver de transistors *ULN2003A*. Aquest xip té 7 entrades i la massa a la part inferior on connectarem les sortides del *LabJack.* A la part superior del xip hi han 7 sortides que varien la seva impedància amb funció del que hi ha a l'entrada corresponent. És a dir, si a l'entrada 2 hi ha un '1' (nivell 3.5 V), la sortida 2 tindrà una impedància baixa. Per tant, la millor alternativa que tenim és muntar-lo amb unes resistències de *pull-up* de valor 2k ohms.



Quant al FIO0 li enviem un '1', a la sortida O1 si mostra una baixa impedància el qual fa que a la entrada I2 del robot veiem un '0'. Quant enviem un '0', a la sortida O1 si mostra una alta impedància, per tant no circularà intensitat el que farà que a la entrada I2 del robot es mesura Vcc, és a dir, estat lògic '1'. Per tant, aquest és un circuit inversor.

8.3- Entorn de programació RobotStudio.

Per programar el robot industrial de la uVic, hem utilitzat el programa RobotStudio. Aquest ens permet simular qualsevol situació real amb un ordinador comercial. És a dir, primer programem utilitzant només un ordinador qualsevol i realitzem la simulació d'aquest. Un cop ens funciona el programa correctament, podem passar a provar-lo en el robot amb mode manual. Podem dir que RobotStudio s'ajusta perfectament a les nostres necessitats.

A continuació es mostra una breu descripció sobre el funcionament del programa. El primer que s'ha d'aprendre és treballar amb les eines que ens ofereix cada pestanya de treball. En el nostre cas, podem treballar utilitzant només les pestanyes següents:

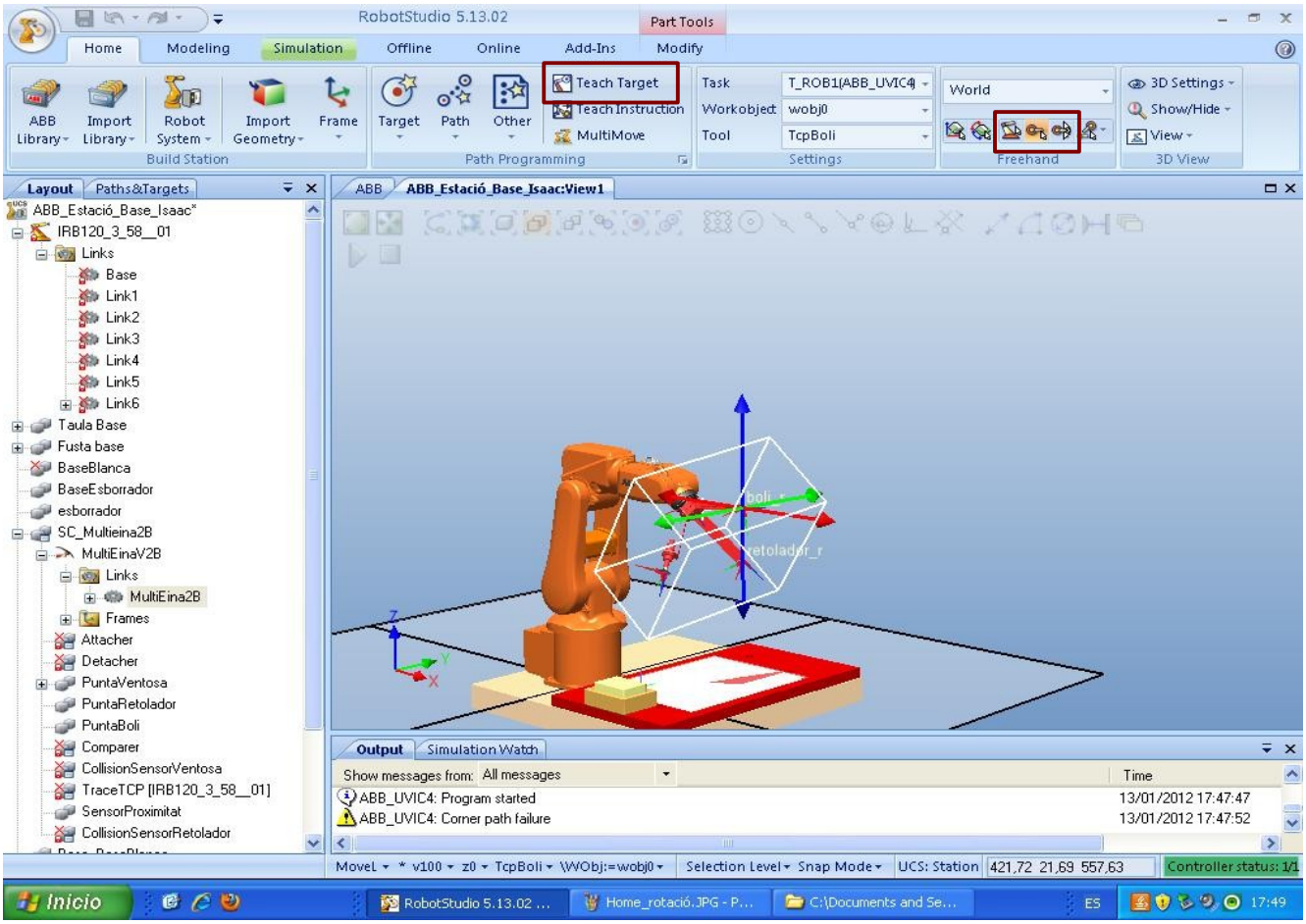
Home: Pestanya on podem mou-re el robot amb el ratolí per tal de buscar posicions. A més, ens ofereix eines per importar objectes.

Simulation: Pestanya on realitzarem les simulacions del funcionament del programa dissenyat.

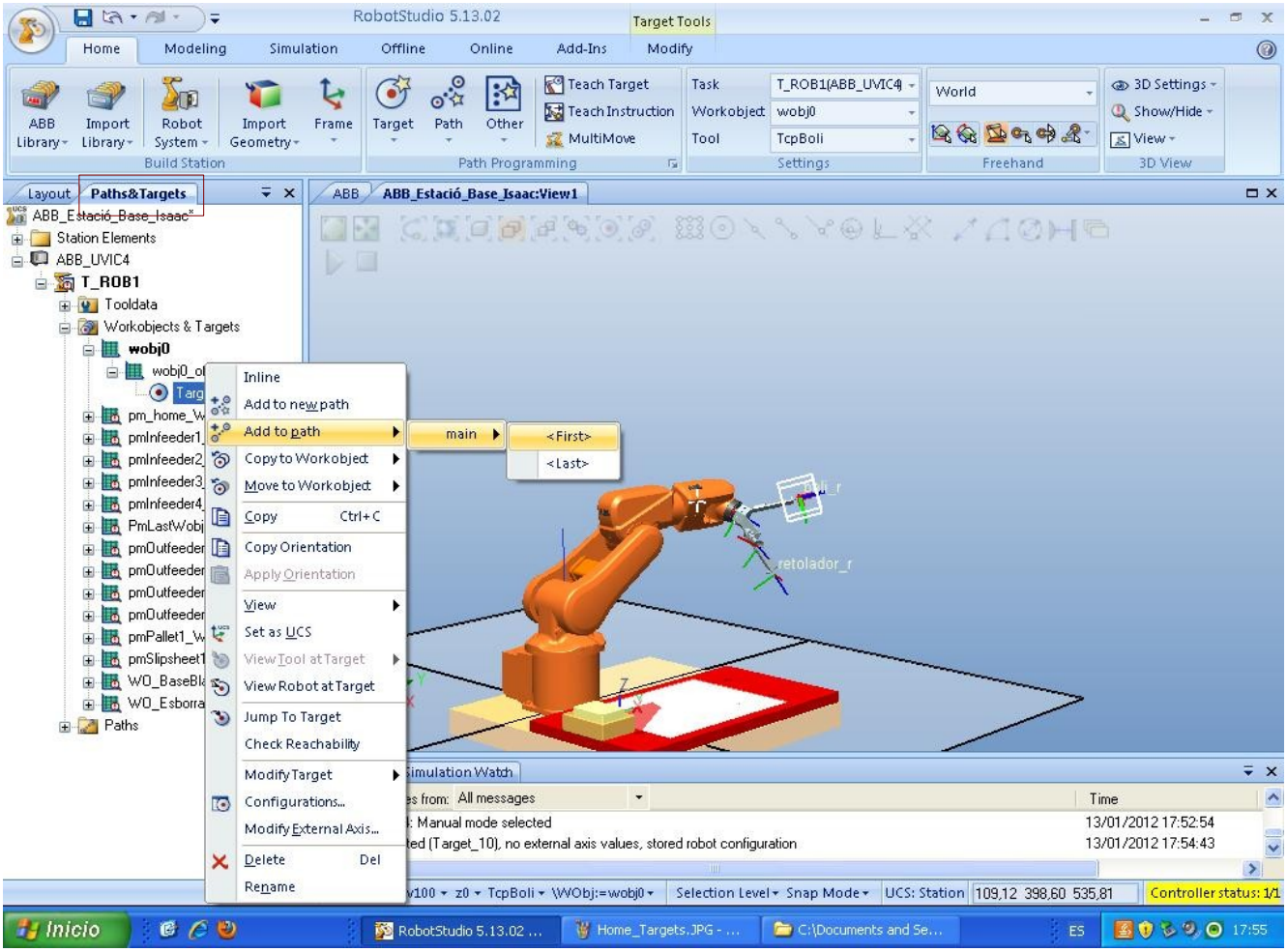
Offline: Pestanya on editarem el codi del nostre algoritme i configurarem algunes opcions de la simulació.

Per utilitzar per primer cop el programa, es aconsellable obrir algun programa existent. El primer que té que estat definit abans de començar a programar és l'entorn del robot, és a dir, si hi ha alguna taula o altres objectes en la seva zona de treball... És per això que es aconsellable treballar a partir d'alguna aplicació existent, d'aquesta forma ja tindrem l'entorn del robot modelat.

Un cop es té l'entorn de treball modelat, podem mou-re lliurement el robot i memoritzar les posicions trobades. Per moure el robot, cliquem a una de les 3 icones de moviment del robot. Aquestes són: moviment per eixos, interpolació lineal i interpolació circular. Cliquem a una de les tres eines, seleccionem la part que es desitja moure del robot i el movem amb el ratolí. En la següent imatge podem veure el resultat i les icones:



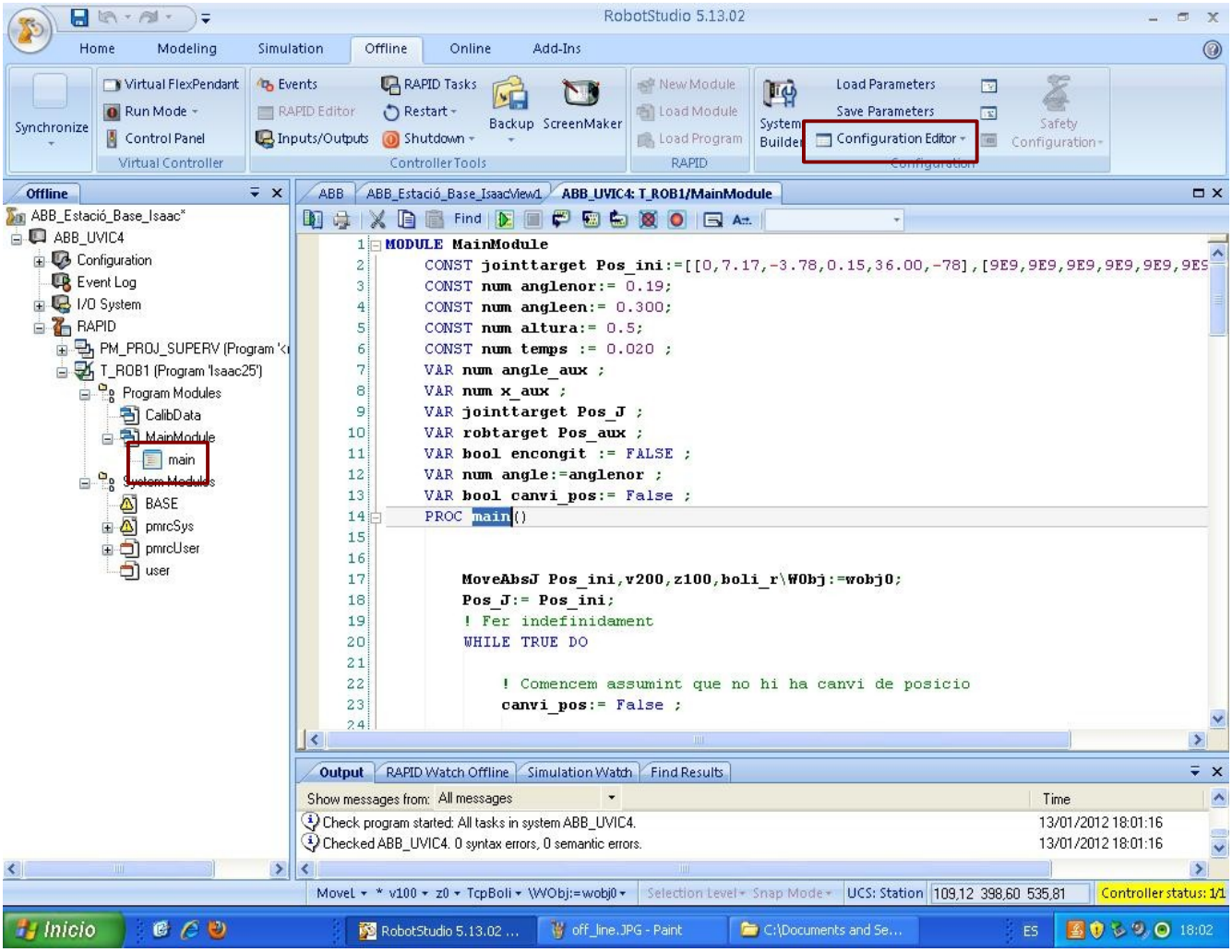
Un cop hem situat el robot en la posició desitjada utilitzant les eines que ens ofereix el programa, podem memoritzar la posició clicant al botó “Teach Target”. Aquesta posició ens apareixerà a la pestanya “Paths & Targets” de la finestra d'arxius. En cas de voler programar una instrucció de moviment a aquella posició, podem crear la instrucció de moviment a l'algoritme del programa clicant amb el botó dret a la posició i seleccionant “Add to path”. En la següent imatge es mostra el resultat.



Com veurem més endavant, en la pestanya Offline a l'arxiu “Main” s'ens ha afegit l'instrucció *moveL Target10,v100,z0:...*

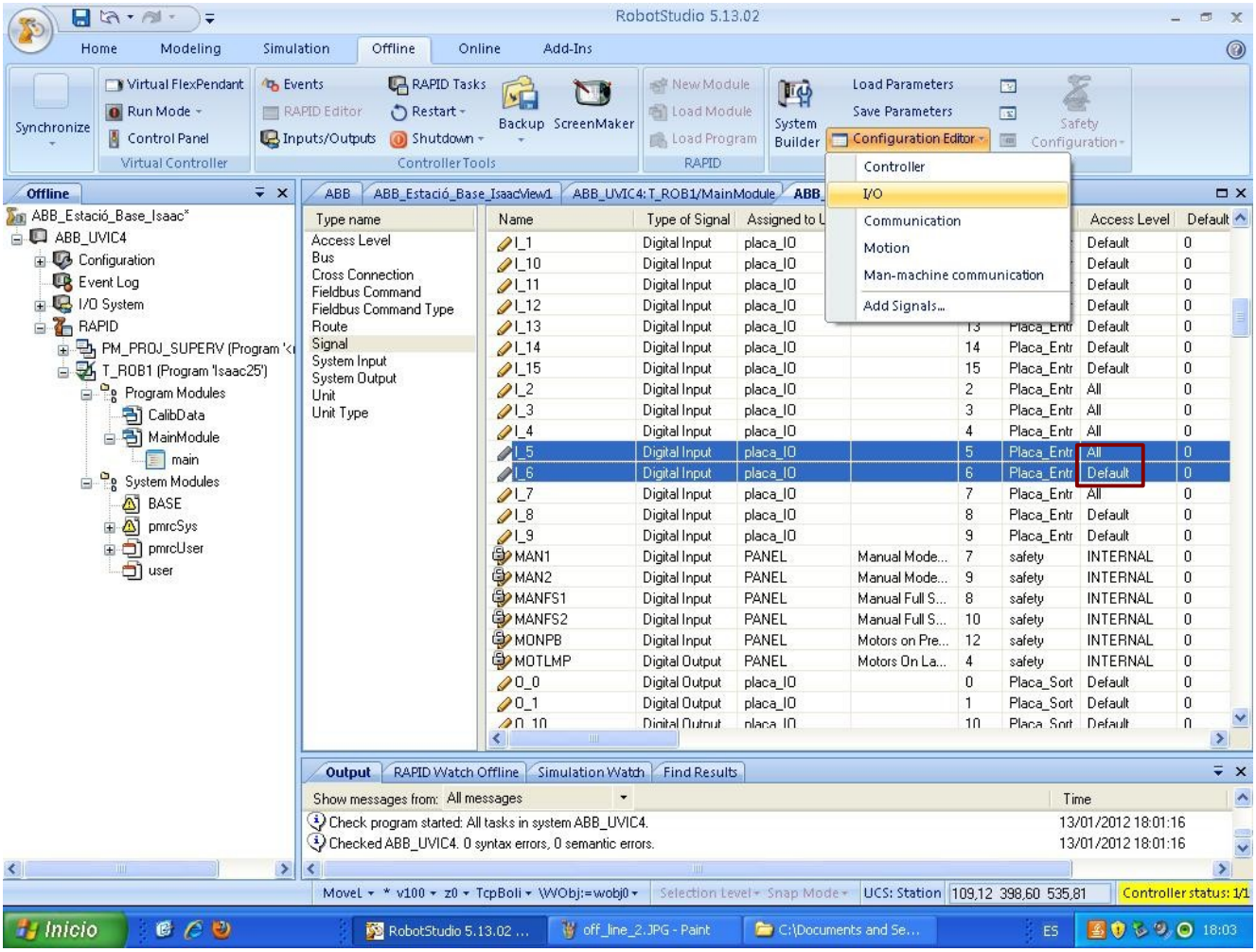
Si el que volem és crear una aplicació on el robot vagi de posició en posició podem buscar les posicions manualment, gravant-les *(Teach Target)* i afegint-les al Main (*add to Path*). En el nostre cas, el robot actua amb funció de les entrades digitals i està programat amb estructura condicional. Per tant, serà necessari programar utilitzant el llenguatge Rapid. Aquest és un llenguatge d'alt nivell pensat per la creació d'algoritmes per a robots.

En la pestanya *Offline* podem canviar el codi del nostre programa. Aquest és troba a el requadre “main” de la finestra d'arxius:



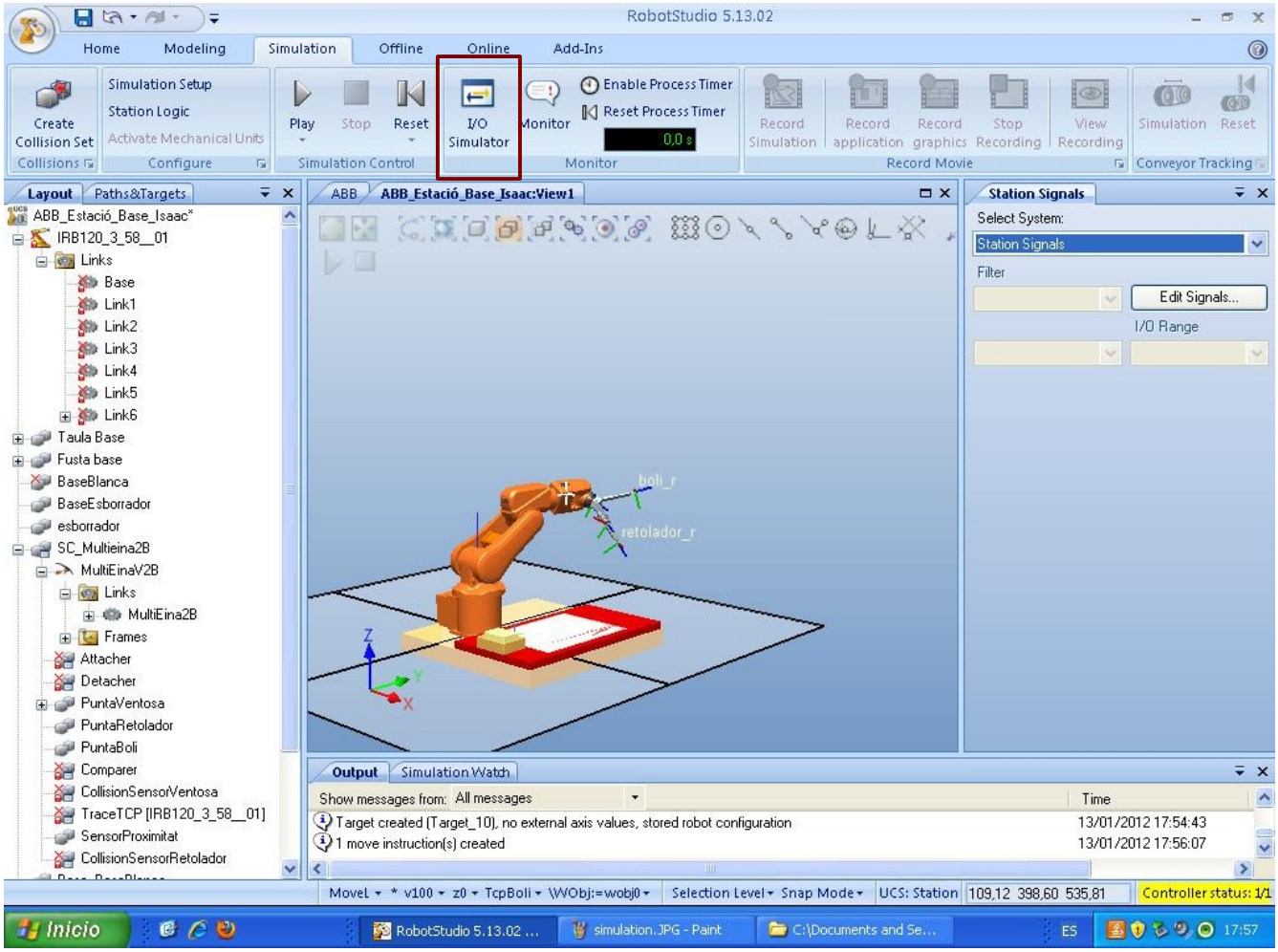
En el cas de voler treballar amb entrades i sortides digitals, serà necessari donar-los els permisos necessaris per poder ser canviat el seu estat des de la pestanya de simulació.

Per configurar aquestes entrades o sortides cliquem al botó *Configuration editor* situat a la barra d'eines superior. Seleccionem I/O i canviem el “Acces level” de les entrades que volem utilitzar a *All.*



Un cop tenim l'algoritme desenvolupat el compilem teclejant Con*trol+Shift+s.* Si aquest no ens dóna cap error, podem passar a simular el nostre projecte.

En la pestanya *simulation* és on podem realitzar la simulació del programa clicant a l'icona *play*. En el cas de voler simular entrades i sortides digitals, premeu el botó I*/O simulator* seleccionem el mòdul ABB_UVIC4—placa_IO. I ja podem simular les entrades i veure el valor de les sortides.



Tot i que els resultats de la simulació són molt aproximats, no són exactes. Per tant, és convenient tenir en compte que les velocitats de desplaçament seran diferents a la realitat. Tot i això, ens serveix per fer una bona primera aproximació del que serà l'algoritme final.

8.4- Algoritme RAPID

Abans de començar a explicar el codi és necessari explicar les constants i variables que té l'algoritme.

Pos_ini (tipus jointtarget): Aquesta constant representa la posició inicial del programa. És a dir, cada cop que re-inicialitzem el programa, el robot anirà a aquesta posició. El format *jointtarget* consisteix en la descripció d'una posició mitjançant els angles dels eixos del robot i coordenades externes (en el nostre cas, no utilitzades).

a*nglenor:* És l'increment o decrement que s'aplica a l'angle *J1 (angle del primer eix del robot)* en cada cicle per orientar-lo amb el pla XY, és a dir, girar horari o antihorari el robot sencer. Per poder canviar l'angle *J1* d'una posició, és necessari que aquesta sigui amb format *jointtarget.* I per moure'ns a una posició *jointtarget* hem d'utilitzar l'intrucció *movabsJ* en lloc de la típica *movabs*

a*ngleen:* Exactament el mateix que *anglenor* quant el robot es troba amb estat encongit.

a*ltura:* És l'increment o decrement que s'aplica a la coordenada *z.* Per tal de poder realitzar aquest increment, és necessari que la posició sigui amb format *robtarget.*

temps: és el temps d'espera del robot cada cicle del bucle per garantir que vagi sincronitzat amb la càmera, 40 ms és el temps de frame de la càmera (25 fps).

En el cas d'aquest algoritme, s'alterna entre canvis mitjançant els eixos del robot i mitjançant interpolacions lineals en un pla. És per això, que necessitem anar transformant la nostra posició entre tipus *jointtarget (per eixos)* i *robtarget (per coordenades).* La resta de variables són per mantenir valors en cas de necessitar recuperar-los com ja es veurà en l'algoritme.

```
MODULE MainModule
! Declaracio de constants i variables
CONST jointtarget Pos_ini:=[[0,7.17,-3.78,0.15,36.00,-78],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST num anglenor:= 0.19;
CONST num angleen:= 0.300;
```

```
CONST num altura:= 0.5;
CONST num temps := 0.040 ;
VAR num angle_aux ;
VAR num x_aux ;
VAR jointtarget Pos_J ;
VAR robtarget Pos_aux ;
VAR bool encongit := FALSE ;
VAR num angle:=anglenor ;
VAR bool canvi_pos:= False ;
! Inici de programa
PROC main()
        ! Ens situem en la posicio inicial
        MoveAbsJ Pos_ini,v200,z100,boli_r\WObj:=wobj0;
        ! Actualitzem la posicio actual
        Pos_J:= Pos_ini;
        ! Fer indefinidament
        WHILE TRUE DO
                ! Comencem assumint que no hi ha canvi de posicio
                canvi_pos:= False ;
                ! Si hi ha cara detectada
                IF I_7=1 THEN
                        ! Si I_2, girem en sentit horari
                        IF I_2=1 THEN
                                ! Decrementem l'angle dessitjat (si no estem al limit)
                                IF      Pos_J.robax.rax_1>-110 THEN
                                        Pos_J.robax.rax_1:= Pos_J.robax.rax_1-angle;
```

```
                ! Marquem canvi de posicio
                canvi_pos:= True ;
        ENDIF
ENDIF

! Si I_3, girem en sentit antihorari (si no estem al limit)
IF I_3=1 THEN
        IF      Pos_J.robax.rax_1<40 THEN
                Pos_J.robax.rax_1:= Pos_J.robax.rax_1+angle;
                ! Marquem canvi de posicio
                canvi_pos:= True ;
        ENDIF
ENDIF

! Si I_4, i no estem en estat encongit movem eix z cap a dalt
IF (I_4=1) AND (encongit)=FALSE THEN
        ! Convertim la posicio a format robtarget
        Pos_aux := CalcRobT (Pos_J,boli_r\WObj:=wobj0);
        ! Incrementem altura linealment (si no estem al limit)
        IF      Pos_aux.trans.z <650 THEN
                Pos_aux.trans.z:=Pos_aux.trans.z+altura ;
                ! Marquem canvi de posicio
                canvi_pos:= True ;
        ENDIF
        ! Convertim la posició a format joint target
        Pos_J := CalcJointT (Pos_aux,boli_r\WObj:=wobj0);
```

```
ENDIF

! Si I_5, i no estem en estat encongit movem eix z capa baix
IF (I_5=1) AND (encongit)=FALSE THEN
        ! Pasem la posicio a format robtarget
        Pos_aux := CalcRobT (Pos_J,boli_r\WObj:=wobj0);
        ! Incrementem altura (si no estem al limit)
        IF      Pos_aux.trans.z >475 THEN
                Pos_aux.trans.z:=Pos_aux.trans.z-altura ;
                ! Marquem canvi de posicio
                canvi_pos:= True ;
        ENDIF
        ! Actualitzem la posició a format joint
        Pos_J := CalcJointT (Pos_aux,boli_r\WObj:=wobj0);
ENDIF

! Cas molt especial
! Si I_4 i I_5 vol dir que el tamany de la cara es molt gran per tant
! retrocedim el robot
IF ((I_4+I_5)=2)AND((I_2 +I_3)<2) THEN
        ! Si encara no estem en l'estat especial, l'apliquem
        IF encongit=FALSE THEN
                ! Guardem el valor de l'angle actual
                angle_aux := Pos_J.robax.rax_1 ;
                ! El situem alineat amb l'eix x
```

```
                Pos_J.robax.rax_1:=0 ;
                ! El convertim a coordenades robtarget
                Pos_aux := CalcRobT (Pos_J,boli_r\WObj:=wobj0);
                ! Guardem la posicio x per quant tornessim a l'estat
                ! natural
                x_aux := Pos_aux.trans.x ;
                ! Retrocedim unicament l'eix x
                Pos_aux.trans.x := 300 ;
                ! Calculem la posicio amb format joint
                Pos_J := CalcJointT (Pos_aux,boli_r\WObj:=wobj0);
                ! Recuperem l'angle inicial
                Pos_J.robax.rax_1 := angle_aux   ;
                ! Marquem canvi de posicio
                canvi_pos:= True ;
                ! Marquem que estat amb estat encongit
                encongit := TRUE ;
                angle:= angleen ;
        ENDIF

ENDIF


! Si estem en estat encongit i la cara es torna a allunyar, tornem a la
! posicio d'abans d'encongir
IF encongit AND ((I_4+I_5)<2) THEN
        ! Guardem el valor de l'angle actual
        angle_aux := Pos_J.robax.rax_1 ;
        ! El situem alineat amb l'eix x
```

```
                    Pos_J.robax.rax_1:=0 ;
                    ! El convertim a coordenades robtarget
                    Pos_aux := CalcRobT (Pos_J,boli_r\WObj:=wobj0);
                    ! Tornem l'eix x a la posicio previa al canvi d'estat
                    Pos_aux.trans.x := x_aux ;
                    ! Calculem la posicio amb format joint
                    Pos_J := CalcJointT (Pos_aux,boli_r\WObj:=wobj0);
                    ! Recuperem l'angle inicial
                    Pos_J.robax.rax_1 := angle_aux   ;
                    ! Marquem canvi de posicio
                    canvi_pos:= True ;
                    ! Indiquem que ja ens trobem en estat normal
                    encongit := FALSE ;
                    angle:= anglenor ;
                ENDIF
                ! Un cop calculada la posicio desitjada, ens movem (si es que hi ha
                ! hagut algun canvi )
                IF canvi_pos THEN
                    MoveAbsJ Pos_J,v300 ,z0,boli_r\WObj:=wobj0;
                    ! Temps d'espera per sincronitzar amb la camera
                    WaitTime temps ;
                ENDIF
            ENDIF
        ENDWhile
ENDPROC
ENDMODULE
```

9- CONCLUSIONS

Primerament, voldria fer un especial agraïment a Ricardo Amèzquita i a Jordi Serra per haver desenvolupat les llibreries necessàries per fer funcionar la càmera uEye amb python i haver modelat tot l'entorn de treball del robotstudio. A més de la seva paciència i col·laboració contestant tots els meus correus electrònics referits a problemes que hem van sorgir durant aquest projecte. Moltes gràcies.

L'objectiu més important per mi d'aquest projecte era aprendre sobre processat d'imatge i sobre programació de Robots. A més, he pogut gaudir d'experimentar amb un robot real en lloc d'una simple simulació fent el meu treball molt més interessant i productiu.

També, he agafat experiència en el camp de la programació de robots industrials i en resoldre problemes reals que no passen en una simple simulació. També s'ha demostrat de cara a futures aplicacions que es poden obtenir resultats molt útils en l'extracció d'informació a partir d'una càmera web.

Tot i això, m'agradaria fer referència a alguns problemes que vaig tenir com ara; a l'hora de fer funcionar la càmera uEye amb python, transformar-la a format opencv, ajustar els paràmetres de les funcions d'opencv, instal·lacions fallides per falta de llibreries, etc. Alguns problemes es resolen amb una simple comanda o funció que costa dues setmanes d'investigació per trobar-la i això m'ha fet adonar de la importància de ser persistent i creatiu amb el meu treball i esforç. En alguns moments m'he sentit molt estancat per que els resultats no s'ajustaven a lo esperat, però finalment, utilitzant el mètode assaig-error he aconseguit trobar combinacions de constants que resolen alguns dels problemes.

Per acabar, m'agradaria dir que el treballar amb un entorn real, m'ha motivat molt en el meu dia a dia i m'ha ajudat a ser molt més constant i metòdic en la meva feina. Destacar que tant l'estudi de recerca inicial com la realització del projecte m'ha aportat un conjunt de competències personals que hem motiven molt a l'hora d'encarar el meu futur.

10- FUTURS PROJECTES

1- Una bona idea de cara a un futur projecte consisteix en aplicar algoritmes de visió estereoscòpica mitjançant una única *webcam* acoblada a un braç robot, però desplaçada a una distància coneguda per tal de poder comparar entre les dues imatges capturades en cada posició. Es pot treballar amb el robot i el *LabJack* com s'ha fet en aquest projecte per sincronitzar els desplaçaments del robot. A més, les llibreries *opencv* ens ofereixen moltes funcions de calibrats de càmera, funció de visió estereoscòpica, etc.

2- Una altra possible aplicació, consisteix en una càmera de vídeo vigilància que detecta, captura una imatge i intenta identificar la/les cara/es de l'individu/a que passa per una determinada porta. Tenim molta feina avançada pel que fa a funcionament de la càmera *uEye* i sobre com detectar les cares, el qual pot ser un interessant projecte sobre reconeixement facial.

11- BIBLIOGRAFIA

- Bradsky, Gary i Kaehler, Adrian (2008): *Learning OpenCv*

  O'reilly media, edició setembre 2008. Idioma: anglès

- Lutz, Mark (1999): *Learning Python*

  O'reilly media, edició octubre 2007. Idioma: anglès

- J. Norberto Pires (2007): *Industrial Robots Programming*

  Springer Science, edició febrer 2007. Idioma: anglès

11.1- Webgrafia

Relacionats amb la càmera uEye:

1. *http://www.ids-imaging.com/*
2. *http://translate.google.es/translate?hl=es&sl=en&tl=es&u=http%3A%2F%2Fopencv.willowgarage.com%2Fdocumentation%2Fpython%2Freading_and_writing_images_and_video.html&anno=2*
3. *code.google.com/p/pyueye/*
4. http://www.photonics.com/Article.aspx?AID=49567

Relacionats amb el LabJack:

1. http://labjack.com/support/labjackpython
2. http://labjack.com/support/linux-and-mac-os-x-drivers
3. http://labjack.com/u3/specs

Relacionats amb Python:

1. mundogeek.net/tutorial-python/
2. opencv.willowgarage.com
3. docencia-eupt.unizar.es/ctmedra/tutorial_opencv.pdf
4. www.trozosdecodigo.com/2009/06/26/face-detection/
5. www.cse.iitk.ac.in/users/.../OReilly%20Learning%20OpenCV.pdf

Relacionats amb el robot ABB

1. rab.ict.pwr.wroc.pl/irb1400/overviewrev1.pdf
2. www.abb.com/robots