LandMark Detection Cam

Germán Giménez Gallardo

Resum— El projecte aporta una solució perquè els robots autònoms siguin capaços d'orientar-se en el seu entorn. Per a complir aquest objectiu es faran servir unes marques o etiquetes col·locades per l'entorn on és mourà el robot i una càmera USB per tal que el robot sigui capaç de llegir aquestes marques. Un dels punts importants del projecte és el fet que el sistema ha de poder implementar-se en un robot que tingui un cost aproximat d'uns 200 €. Això vol dir que s'ha de poder fer servir components amb un pressupost limitat. En aquest cas he fet servir una càmera USB d'uns 7 € i marques simples que es poden imprimir o pintar a qualsevol lloc.

Paraules clau—Robot mòbil, Robot autònom, Marques, Blobs, Detecció de blobs, Navegació

Abstract— The project provide a solution for autonomous robots to orient themselves in their surroundings. To meet this objective trademarks used or labels placed by the environment in which the robot will move and a USB camera , so that the robot can be able to read those marks. One of the important points of the project is that the system must be able to be implemented in a robot that has an estimated cost of 200 €. This means that should be able to use components with a limited budget. In this case I used a USB camera that costs 7 € and simple marks can be printed or painted cualsevol place.

Index Terms—Mobile robot, Autonomous robot, Marks or Labels, Blob, Blob detection, Navigation.

1 Introducció

Avui dia l'ús de robots mòbils és més freqüents del que la majoria de persones pensem. El seu ús s'ha estès des de vehicles automatitzats com trens o metro fins a robots que mouen materials dintre d'un magatzem o drons que permeten filmar imatges o transportar coses. Per altra banda tots aquests robots mòbils, si són autònoms, necessiten algun dispositiu que els permeti rebre informació del seu entorn per poder interactuar-hi.

Ja que les càmeres són actualment un dels sensors més comuns i poden ser utilitzades amb diverses finalitats no és cap sorpresa que la majoria de robots autònoms les utilitzin.

L'objectiu del treball és crear un mòdul detector de marques que es pugi adaptar a models a escala de robots autònoms. Això planteja certs reptes, ja que els vehicles han de ser petits i els sensors que s'utilitzen no podran ser els mateixos que en els més grans. Aquest últim punt també és important, ja que el model ha de ser de baix cost i això afectarà els diferents components del robot.

La manera més simple d'orientar a aquests robots en un entorn conegut és fent servir balises col·locades estratègicament de manera que aquestes aportin informació al robot sobre aquest entorn i que ha de fer. Aquestes balises poden ser de dos tipus, passives o actives, depenent de si la balisa ha de realitzar algun tipus d'acció perquè el robot rebi la informació. En el nostre cas ens hem decantat per fer servir balises passives, concretament etiquetes llegibles amb la càmera del robot.

Un punt important és que el nostre robot ha de tenir un cost econòmic baix. Amb un pressupost elevat es pot tenir accés a recursos que fan senzill el fet que un robot mòbil interactuï amb el que té al seu voltant. Com la idea és que el preu del robot no sigui gaire elevat treballarem amb certes limitacions econòmiques. L'objectiu és que el cost total del robot sigui inferior a 200 €, amb aquest pressupost hem de muntar tot el robot no només el nostre sistema. Per aquests motius farem servir càmeres USB o PAL bastant econòmiques i que, per conseqüència, no són de tanta qualitat. De totes maneres que la qualitat de la càmera sigui baixa no ens afecta gaire, ja que encara que fos molt bona estaríem limitats per la capacitat computacional del robot i trigaria massa a processar les dades que li passés la càmera. Per altra banda no tindria sentit que ens gastéssim els diners que estalviem del robot en la infraestructura perquè es mogui de manera autònoma així que farem servir marques molt simples i que fàcilment es podrien imprimir.

2 Objectius

Es vol crear un sistema que permeti a un robot reaccionar a diferents tipus de marques a l'entorn, de manera que es pugui moure sol i anar fent diferents accions de manera autònoma. Així per què el robot fos autònom només hauríem de col·locar les etiquetes a on sigui necessari i el robot sol aniria fent el que nosaltres volguéssim.

El sistema ha de fer servir una càmera USB o PAL per detectar aquestes etiquetes i ha de ser el més ràpid possible perquè el robot en moviment sigui capaç de llegir les etiquetes i reaccionar a elles.

3 PLANTEJAMENT

Ara que ja hem definit quins són els objectius del projecte i quin és el problema que volem solucionar plantejaré la solució que he desenvolupat. Els recursos dels quals disposem són una càmera USB i una BeagleBone Black, fent servir un sistema de detecció de regions analitzarem la imatge que ens retorna la càmera i en funció dels blobs que trobem, la seva mida o la distància entre ells identificarem diferents tipus d'etiquetes.

Aquest plantejament soluciona la part més general del problema, el fet que el robot pugui identificar les marques, però amb les dades que obtinguem també podem veure si la solució que proposem és prou bona a l'hora d'implementar-la de manera real. Això vol dir que tot i que el nostre sistema funciona si no és prou ràpid, o no detecta les marques prou lluny no serà vàlid fent servir un robot real.

4 ESTAT DE L'ART

Actualment els robots són molt més diversos del que es podria arribar a pensar, n'hi ha de molts tipus que poden tenir funcions totalment diferents. En aquest cas ens interessa centrar la nostra atenció en els robots mòbils.

Un robot mòbil és tot aquell robot que té alguna forma de desplaçar-se pel seu entorn, en aquest cas de manera completament lliure. Per això necessita tenir un mètode per moure's, siguin rodes, potes o hèlix per volar i a més també necessiten sensors per detectar el seu entorn i orientar-se. El seu ús pot ser molt variat, ja que poden fer pràcticament qualsevol funció i ser mòbils a la vegada, per això els podem trobar a magatzems movent objectes, investigant llocs de difícil accés, a l'agricultura o fins i tot per a netejar cases. En el nostre cas ens interessen els robots mòbils orientats principalment a treballar en interiors i en entorns coneguts.

5 COMPONENTS

El primer que hem de definir són els components que es faran servir per desenvolupar el nostre projecte. En aquest cas la placa ha sigut triada perquè és la que feia servir el robot i la que m'ha pogut facilitar la universitat. La càmera compleix els requisits de ser de baix cost tot i que es podria substituir per una de millor en funció del pressupost del projecte.

5.1 BeagleBone Black

El BeagleBone Black és el microcomputador que faré servir per realitzar totes les proves del projecte i muntar el sistema juntament amb la càmera. Com ja he dit ha sigut triat per conveniència, però la idea és que es pot fer amb altres microcomputador, fins i tot pot ser necessari fer-ne servir de més potents perquè tot funcioni correctament. De totes maneres aquest microcomputador encaixa bé en el pressupost per a un robot de baix cost com en el que s'implementaria aquest sistema. Un altre motiu pel qual ha sigut triat és que ja havia treballat prèviament amb una BeagleBone Black.

En aquest cas la placa té instal·lat Debian 7 com a sistema operatiu i té instal·lat a més OpenCV i Video4Linux per interactuar amb la càmera i tractar imatges. Per a muntar el nostre sistema només hi hem hagut de connectar la càmera USB. A la realitat, el microcomputador aniria connectat al robot per fer-lo funcionar.

5.2 Camera USB

He triat fer servir una càmera USB perquè són les més comuns i normalment són prou econòmiques, a l'igual que la BeagleBone Black pot ser substituïda per un altre tipus de càmera si fos necessari o per unes de millor qualitat si és pogués. He fet servir Video4Linux per veu-re les especificacions concretes de la càmera.

```
root@beaglebone:~# v4l2-ctl --all
Driver Info (not using libv4l2):
        Driver name   : gspca_zc3xx
        Card type     : USB Camera (046d:08da)
        Bus info      : usb-musb-hdrc.1.auto-1
        Driver version: 3.8.13
        Capabilities  : 0x85000001
                Video Capture
                Read/Write
                Streaming
Format Video Capture:
        Width/Height  : 640/480
        Pixel Format  : 'JPEG'
        Field         : None
        Bytes per Line: 640
        Size Image    : 115790
        Colorspace    : JPEG (JFIF/ITU601)
JPEG compression:
        Quality: 75
        Markers: 0x00000018
                Define Huffman Tables
                Define Quantization Tables
Video input : 0 (gspca_zc3xx: ok)
Streaming Parameters Video Capture:
        Frames per second: invalid (0/0)
        Read buffers     : 2
Priority: 2
root@beaglebone:~#
```

6 COM IDENTIFICAR MARQUES

La primera cosa que hem de decidir és com agafarem les imatges a través de la càmera, donat que treballarem sobre una BeagleBone Black amb Debian podem fer servir o Video4Linux o OpenCV, com farem servir OpenCV pel tractament posterior de la imatge he optat per fer servir aquesta llibreria.

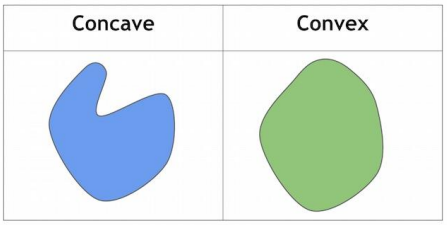
Un cop sabem les característiques de la càmera el primer que hem de fer és capturar imatges amb aquesta càmera, amb OpenCV només hem de definir amb quin dispositiu volem capturar les imatges, en aquest cas com la nostra càmera és el dispositiu 0 farem servir *cv2.VideoCapture(0)*, i posteriorment llegir-les amb la funció *read()*. Després amb *cv2.VideoWriter* guardarem tots els frames que anem capturant en un vídeo en format .avi, això és opcional i només serveix perquè nosaltres puguem observar que està veient la càmera. Un cop tenim les imatges de la càmera hem de veure si n'hi ha etiquetes, per això farem servir el detector de blobs que té el mateix OpenCV.

6.1 Deteccio de blobs

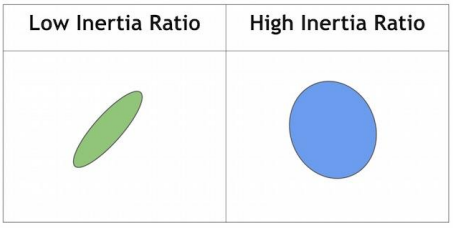
Blob és l'abreviació de *Binary Large Object*, són agrupacions de píxels a una imatge que comparteixen una propietat comú com el color o la forma, l'objectiu principal d'un detector de blobs és identificar aquestes regions de la imatge. Per detectar aquestes agrupacions de píxels el detector primerament transforma la imatge en diverses imatges binàries, el nombre de les quals pot variar en funció dels paràmetres que li passem, i a cada imatge binaria agrupa els píxels segons el criteri de cerca de blobs que ens interessi. Després ajunta els blobs de les diferents imatges binàries si són prou aprop uns d'altres i per ultim calcula el centre i el radi d'aquests blobs finals.

Per al nostre projecte un cop tenim les imatges de la càmera hem de veure si hi ha etiquetes, per això farem servir un detector de blobs. OpenCV té un detector de blobs propi que és el que farem servir per identificar les marques de les nostres imatges, tot i que el funciona-ment de la majoria és quasi idèntic així que es podria haver creat un.

El detector de blobs de OpenCV té tres paràmetres principals que podem modificar detectar els blobs segons uns criteris o uns altres. Per una banda tenim el filtre per color, modificant el paràmetre *blobColor* la funció identificarà els blobs d'un color o d'un altre, després podem diferenciar-los per mida definint una àrea mínima i màxima. També podem buscar blobs segons la seva forma, dins d'aquest filtre tenim diversos paràmetres com la circularitat, la convexitat o la relació d'inèrcia. La circularitat simplement indica com de semblants han de ser els blobs a un cercle, però la convexitat i la relació d'inèrcia són una mica més complicades



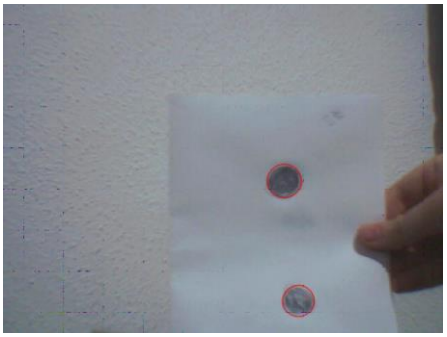
La convexitat fa referencia a la forma del blob, indicant si les corves que formen el blob sobresurten cap a l'exterior o a l'interior de l'àrea del blob. Podem definir uns valors màxims i mínims per a aquesta convexitat en funció de la forma que busquem.



La relació d'inèrcia ens indica com d'allargada és la forma del blob que busquem. Igual que la convexitat es defineix un mínim i un màxim, però en aquest cas el mínim més petit serà 0 que seria una línia i el màxim seria 1, només podem definir els màxims i mínims entre aquests valors.

Sabent els paràmetres ja podem crear el detector amb els que nosaltres desitgem, en el nostre cas activarem el filtre per color i definirem el color del blob com el més fosc possible. Per a les proves que he fet, si no modifiques la resta de paràmetres, el detector per omissió busca blobs circulars. En el nostre cas ja ens serviria per a fer les proves que volem.

Amb el detector creat només hem de passar la imatge capturada per la càmera i ens retornara uns *keypoints*, que inclouen el centre del blob i el diàmetre entre altes. Aquests *keypoints* els farem servir per dibuixar un con-torn al voltant del blol perquè al veure la imatge puguem identificar quins blobs ha detectat. Com en el cas de guardar les imatges en un vídeo això no seria necessari un cop estigui funcionant el robot, ja que només serveix perquè puguem veure els resultats.



Un cop tenia la detecció de blobs vaig fer diferents proves per veure quant temps trigava i quines eren les limitacions de distància i mida dels punts que podia detectar.

Mitjana Temps Total	Mitjana Temps Capturar Imatge	Mitjana Temps de Detecció
247,677 ms	17,252 ms	230,293 ms

En aquesta taula podem observar una mitjana dels temps d'execució del programa només amb la detecció de blobs, després de diverses proves he determinat que la mida o distància del blob no afecta de manera significativa al temps que triga a detectar-lo així que aquesta mitjana està feta amb diferents tipus de blobs.

Com es pot veure la majoria del temps és utilitzat pel detector de blobs, he provat modificar diferents paràmetres per intentar millorar el temps de detecció però no he aconseguit millores apreciables.

Distancia/ Mida(diàmetre)	20cm	1.5m	2m	4m
4mm	Visible	No Visible	No Visible	No Visible
1cm	Visible	Visible	Errors	No Visible
2cm	No Visible	Visible	Visible	Errors
4cm	No Visible	Visible	Visible	Visible

Aquesta taula mostra els resultats de les proves realitzades per determinar la distancia i mida dels punts que pot identificar el detector de blobs. Es pot observar que el detector falla en detectar punts massa petits, o no els detecta o els detecta malament i tampoc detecta punts massa grans que estiguin massa a prop.

Aquest últim punt no ens afecta gaire, ja que és poc probable que hagi de detectar un gran blob a tan poca distància, en canvi sí que ens és útil mirar la relació entre la distància i la mida per determinar quina hauria de ser la mida de les etiquetes en funció de quines distàncies volem que el robot les detecti.

Similar a la taula anterior vaig fer proves amb el diàmetre del blob detectat i la distancia, per tal de veure la seva relació i poder així calcular una distància aproximada entre la càmera i el blob en funció de la seva mida, en aquest cas es van fer servir punts d'1 cm de diàmetre per fer les proves.

Distancia/ Mida(diàmetre)	20cm	1.5m	2m	4m
1cm	52px	8,5px	6,4px	No Visible

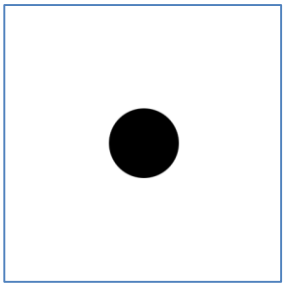
Amb això podem veure que la relació és aproximadament la inversa és a dir si la mida és el doble, la distància serà la meitat. Aquest càlcul no és del tot exacte, ja que el diàmetre que ens dóna el detector pot variar, en aquest cas per les proves vaig fer una mitjana del diàmetre de tots els punts.

6.2 Tipus de marques

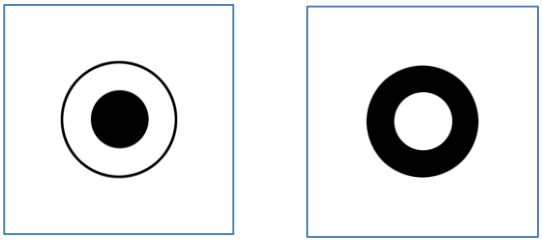
Un punt important ara que ja podem identificar blobs amb les imatges de la càmera és veure quins tipus de marques podem detectar. Com ja hem dit les marques són balises passives que col·locarem a l'entorn perquè el robot les llegeixi i obtingui certa informació. En el nostre cas seran unes etiquetes relativament simples perquè siguin fàcils de crear.

Tot i això hem de definir com seran les etiquetes, ja que haurem de tenir en compte la forma, mida, color per poder identificar-les i a més això afectar a la informació que ens poden aportar.

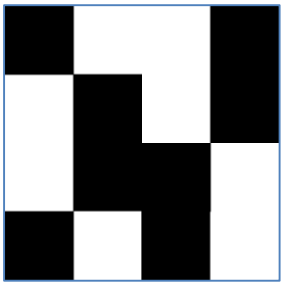
- **Punts**: Són la marca més senzilla de totes, són fàcils de detectar, però és difícil saber si els punts que s'han detectat formen part de la marca o són punts que ha detectat de l'entorn. A més no aporten massa informació perquè només pot variar el nombre de punts que hi ha.



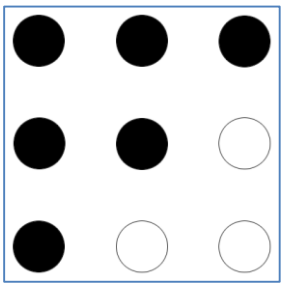
- **Dianes**: Són similars als punts, però en aquest cas és més difícil confondre les dianes per punts agafats per error de l'entorn. En contra tenim el fet que també són més difícils de detectar al ser blobs dintre de blobs.



- **QR Like**: Són marques similars a un codi QR però més simples, de manera que són més ràpides de llegir i identificar. Aquests es basarien en una quadrícula on algunes caselles estarien pintades de negre i altres no, normalment n'hi ha algunes marques o caselles que sempre estan pintades per poder determinar la posició de la resta. Aquestes etiquetes poden donar molta més informació que les anteriors, ja que es poden crear més marques diferents. El principal problema que tenen és que per a la seva forma el detector de blobs pot tenir problemes per detectar la posició d'algunes caselles si per exemple dues caselles contigües són negres.



- **QR Like amb punts**: Solució al problema de les marques QR Like pel nostre sistema, en comptes de fer servir una quadrícula es fan servir una sèrie de punts alineats que actuen com les caselles de la quadrícula, això permet identificar correctament els punts i diverses combinacions per poder crear marques diferents. En aquest cas faré servir unes marques de 3x3 punts, amb tres punts de les cantonades sempre marcats que ens serviran per poder determinar la posició dels altres punts. Això ens deixa amb 26 possibles combinacions amb els punts restants.

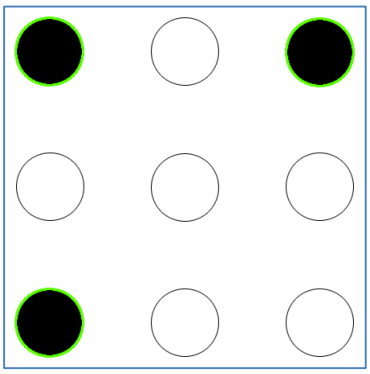


6.3 Identificacio de marques

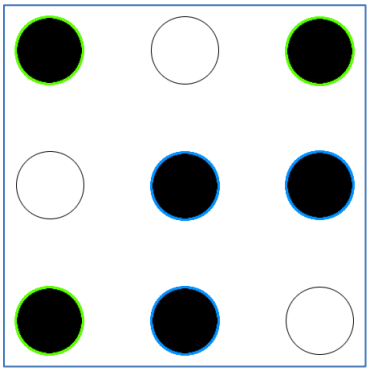
Un cop ja tenim definides les marques podem començar amb l'algoritme per identificar-les. He fet servir les marques QR Like amb punts, ja que són les més fàcils d'identificar. Les etiquetes creades per fer les proves eren de 3x3 punts, això és l'equivalent a una matriu binaria, només hem de localitzar a quines posicions hi ha punts i a quines no. El detector de blobs ens retorna una llista amb tots els blobs que ha detectat, amb això podem saber la seva posició i mida.

El primer que farem serà eliminar tots els punts erronis que hagi pogut detectar. Com sabem aproximadament la disposició dels punts i que com a mínim te que haver-hi tres punts col·locats en una posició especifica podem saber de manera prou fiable si un punt pertany a l'etiqueta o no. El primer que fem és mirar el seu diàmetre i comparar-lo amb el de la resta de punts, en cas de no ser similar a com a mínim tres dels punts podem descartar directament aquest punt, ja que seria massa gran o petit per formar part de la marca. Però això pot suposar un problema si es detecten suficients blobs erronis d'una mida similar, així que mirarem també la seva posició. Comprovem la seva posició i mirem si té a prop altres punts, com que sabem que la distància mínima entre el centre de dos punts serà igual al diàmetre de dos punts i la màxima serà igual al de quatre punts podem veure si té veïns dintre d'aquest rang. En cas que no sigui de la mida correcta o no tingui prou veïns per a formar part d'una etiqueta eliminem el punt.

Un cop tenim només els punts que pertanyen a una etiqueta hem d'identificar quins són els tres punts que ens ajudaran a localitzar la posició de la resta. Sabem a quina distància han d'estar aquests punts entre ells i quina posició han de tenir entre ells així que podem buscar tres punts que compleixin aquestes condicions entre ells.



Ara només hem de localitzar la posició de la resta de punts a la matriu fent servir els punts de referència. Això és tan simple com comparar les coordenades x i y del punt en qüestió amb les dels que ja tenim, sabem així a quina fila i columna de la matriu correspon el punt. Un cop fet això ja tenim la matriu que equivaldria a l'etiqueta que tenim a l'entorn i només seria qüestió de fer que el robot actues en funció de la marca corresponent.



Tot això és més complicat si volem detectar més d'una etiqueta a cada imatge, en aquest cas hem de detectar les diferents marques pels seus punts de referència i agrupar els punts per a cada etiqueta. El problema és que si dues etiquetes estan massa juntes pot donar errors a l'hora de identificar-les.

A més gràcies a la relació que hem trobat entre la mida del punt i la distància amb la càmera podem dir aproximadament a quina distància es troba l'etiqueta de la càmera. Això si, això només servirà si la mida dels punts és d'1 cm de diàmetre, en cas contrari s'haurien de fer proves per calibrar-lo bé.

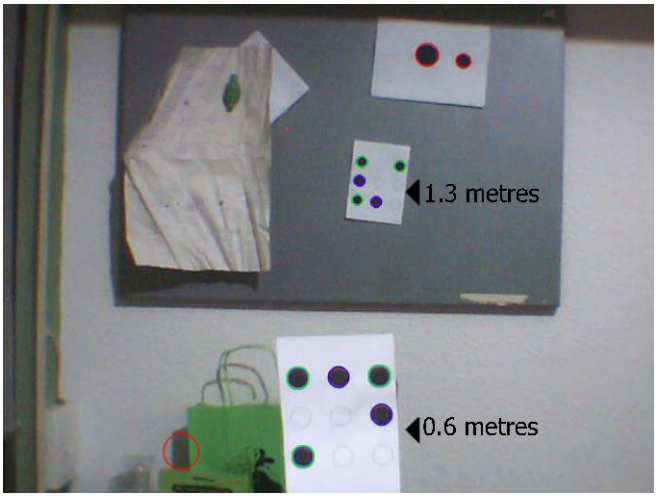
Mitjana Temps Total	Mitjana Temps Capturar Imatge	Mitjana Temps de Detecció	Mitjana Temps de Identificació
279,5ms	17,834ms	252,651ms	8,906ms

Amb la detecció de marques i la seva identificació ja implementades he tornat a fer les proves per mesurar el temps d'execució del programa. Com es pot observar el temps d'identificació és molt baix en comparació amb el de detecció dels blobs que continua sent el punt crític, en aquest cas la mitjana ha augmentat perquè, en aquest cas, les proves s'han fet amb marques i en diversos entorns que podien donar punts erronis per veure si es filtraven correctament. Al haver de detectar més blobs i en alguns casos sent aquests més difícils de detectar.

6 Resultats

El sistema que hem muntat compleix tots els objectius que havíem proposat al principi, és capaç de detectar i identificar marques i a més pot calcular aproximadament distàncies. Amb això, un robot seria capaç de moure's llegint les etiquetes. He fet una última prova per mostrar els resultats finals del projecte. En aquesta he col·locat dues marques a diferents distàncies juntament amb uns punts perquè facin de blobs erronis.

El sistema ens mostra tots els blobs que ha detectat encerclats en diversos colors que mostren el tipus de punt que són. Els vermells són aquells punts detectats però que no compleixen les característiques per formar part d'una etiqueta i per consegüent son descartats. Els punts verds i blaus formen part de les etiquetes, els verds són els que indiquen els punts de referència de cada marca i els blaus la resta de punts de l'etiqueta..



Com es pot observar, el sistema els detecta correctament, però només amb això no podem assegurar que el sistema sigui vàlid per al robot. Com ja hem dit, s'ha de tenir en compte la velocitat a la qual ho detecta per veure si realment pot ser útil. En aquest cas he mesurat el temps des de la BeagleBone Black i després des d'un ordinador per veure la diferència entre els temps d'un i un altre per veure fins a quin punt este condicionat per la potència del processador.

Plataforma	Mitjana Temps Total	Mitjana Temps Capturar Imatge	Mitjana Temps de Detecció	Mitjana Temps de Identificació
BeagleBone Black	300,5ms	17,9ms	267,4ms	15,1ms
PC	85,9ms	68,6ms	16,5ms	0,7ms

La taula mostra dades pràcticament iguals a les que han donat les altres proves, en aquest cas hi ha més temps dedicat a la identificació pel fet que hi ha dues marques per identificar, a més ens ha retornat una aproximació bastant bona de la distància a la qual es troben aquestes marques, la més propera a 0,63 m i l'altra a 1,25 m com ja he dit aquests valors no són del tot exactes, ja que es fa servir una mitjana de les mides dels punts de la marca, i aquesta mida no és 100% exacte, ja que es basa en el diàmetre dels blobs identificats. Aquesta detecció de distàncies és important, ja que nomes identificar l'etiqueta pot donar lloc a errors quan el robot s'hagi d'orientar. Per exemple si hi hagués una que el robot interpretes com "Gira a la dreta", aquesta informació és inútil si no sabem a quina distància es la marca. Tot i això, aquesta funcionalitat es pot ignorar si el mateix robot té sistemes per calcular distancies o les marques no han de donar ordes al robot perquè es mogui.

El temps d'execució és prou elevat, sobretot tenint en compte que la placa també haurà de controlar al robot, això pot augmentar encara mes els temps i condicionar el temps de reacció del robot. Segueix sent possible però la velocitat del robot es veuria afectada. Per posar un exemple, la velocitat del robot normalment és de 20cm/s, 40cm/s com a molt, aixo vol dir que el robot recorreria com a mínim 0,12 metres abans de reconèixer la marca, això s'ha de tenir en compte a l'hora de veure si haurem de millorar el temps d'execució perquè el robot pugui complir la seva funció.

Si mirem els resultats del mateix codi executat aquest cop a un ordinador, el canvi del processador és prou notable sent tres vegades més ràpid. En aquest cas el punt crític ja no és la detecció de blobs sinó que passa a ser la captura de les imatges, sent més lenta fins i tot que en el cas de la BeagleBone. Les proves s'han fet a un ordinador que tenia Windows 10 com a sistema operatiu, és possible que amb altres sistemes basats en Linux com el que té la BeagleBone poguéssim millorar aquest temps. Tot i això aquestes dades són útils per saber que en cas que necessitéssim més velocitat a l'hora d'executar la detecció de marques es pot aconseguir amb versions més potents d'aquesta placa o similars.

7 CONCLUSIÓ

En aquest projecte s'ha demostrat la viabilitat de crear un sistema de detecció de marques a l'entorn, en aquest cas orientat a un robot mòbil però que fàcilment es pot aplicar a més tipus d'aplicacions. Un exemple seria aplicacions per mòbils que ajudin a orientar-se en grans espais, ja que la identificació d'aquestes marques és rapida i potser practica per aquest tipus de funcions. La solució proposada es vàlida i compleix els objectius proposats. Evidentment es pot millorar el rendiment en qüestió de temps d'execució o errors millorant els components del sistema.

A més s'ha explicat com funciona la detecció de blobs amb OpenCV tot i que es pot aplicar a la detecció de blobs en general, mostrant com identifica els blobs i els diferents paràmetres que podem tocar per identificar diferents tipus de blobs. En relació amb això s'ha aprofundit en el coneixement de la llibreria OpenCV, ja que és la que hem fet servir per capturar les imatges, processar-les i, en aquest cas, per poder veure els resultats mostrar-les de diferents formes, en temps real, com un vídeo o com imatges.

Una de les possibles millores seria el fet d'habilitar la detecció de blobs per colors a part del negre, això augmentaria la quantitat d'opcions per a les marques que tenim donant més possibles variants. També es podria perfeccionar el sistema de detecció de distàncies o afegir un detector d'infrarojos per a aquesta funció, ja que saber de forma precisa la distància de les etiquetes ajudaria a saber millor quan reaccionar. Amb millors càmeres es podria millorar la distància de detecció i reduir la taxa d'errors a l'hora de detectar els blobs que, a certa distància en funció de la seva mida no es detecten bé.

AGRAÏMENTS

Agraïments especialment al meu tutor del treball Lluis Ribas Xirgo per l'ajuda i orientació a l'hora de definir el projecte i també els consells per fer-lo. A la Universitat Autonoma de Barcelona agrair-li el facilitar-me els materials necessaris per a poder realitzar el projecte, en aquest cas la BeagleBone Black.

BIBLIOGRAFIA

[1] Web de OpenCV.
http://docs.opencv.org/trunk/index.html#gsc.tab=0
[2] Guies de captura d'imatges de Derek Molloy.
http://derekmolloy.ie/beaglebone-images-video-and-opencv/
http://derekmolloy.ie/beaglebone/beaglebone-video-capture-and-image-processing-on-embedded-linux-using-opencv/
[3] StackOverflow.
http://stackoverflow.com/
[4] Guia per detectar blobs amb OpenCV.
http://www.learnopencv.com/blob-detection-using-opencv-python-c/