

Robot Industrial Stäubli TX60 com a classificador de residus

Ricard López Olivares

Resum– La robòtica ha transformat la nostra vida quotidiana de diverses maneres. Des de robots domèstics que realitzen tasques de la llar fins a sistemes automatitzats a indústries i hospitals, els robots estan cada vegada més presents. Han millorat l'eficiència i la precisió en la fabricació i l'atenció mèdica, i han facilitat la realització de tasques repetitives i perilloses. A més, els avenços en la intel·ligència artificial han permès el desenvolupament de robots autònoms i col·laboratius, que interactuen amb els humans de manera més natural.

En aquest treball s'ha utilitzat un robot Stäubli Tx60 i s'ha incorporat la visió per computador, aquest robot té dues comunicacions una amb un PLC, que li diu la velocitat de la cinta i si hi ha objecte, en cas que hi hagi un objecte enviarà un senyal al robot i aquest a l'ordinador per posar en marxa la visió per computador i classificar l'objecte, un cop classificat s'envia al robot el resultat i mou l'objecte a la seva zona.

Paraules clau– Robot, Tx60, Stäubli, antropomòrfic, intel·ligència artificial, classificador, xarxa neuronal, residus, industrial, DenseNet-121, Val3, Python, Sockets.

Abstract– Robotics has transformed our daily lives in various ways. From domestic robots that perform household tasks to automated systems in industries and hospitals, robots are becoming increasingly present. They have improved efficiency and accuracy in manufacturing and healthcare, and have facilitated the execution of repetitive and dangerous tasks. Furthermore, advancements in artificial intelligence have enabled the development of autonomous and collaborative robots that interact with humans in a more natural way.

In this project, a Stäubli Tx60 robot was used, incorporating computer vision. This robot has two communications: one with a PLC, which informs it of the conveyor belt's speed and detects the presence of an object. If an object is detected, a signal is sent to the robot and the computer activates computer vision to classify the object. Once classified, the result is sent back to the robot, which then moves the object to its designated area.

Keywords– Robot, Tx60, Stäubli, anthropomorphic, artificial intelligence, classifier, neural network, waste, industrial, DenseNet-121, Val3, Python, Sockets.

1 INTRODUCCIÓ

EN l'era de la Indústria 4.0, la necessitat de solucions innovadores per abordar els reptes del reciclatge i la sostenibilitat es fa cada vegada més urgent. En aquest emocionant projecte, es busca crear un canvi significatiu en combinar la tecnologia de xarxes neuronals amb un robot de reciclatge d'avantguarda. L'objectiu principal és revolucionar la forma en què es gestionen els residus

i fomentar una economia circular més eficient. Aquest robot autònom està dissenyat per identificar i classificar de manera precisa i eficient els materials reciclables, com ara plàstics, vidres, metalls, paper, cartró, etc, en corrents de deixalles.

La clau d'aquest projecte rau en la implementació de xarxes neuronals, que permeten al robot aprendre i adaptar-se a diferents tipus de materials a mesura que es troben en el procés de reciclatge. La integració en el marc de la Indústria 4.0 proporciona un avantatge addicional en permetre el monitoratge en temps real i l'anàlisi de dades, cosa que optimitza encara més el procés de reciclatge i permet identificar i solucionar problemes de manera eficient. En millorar l'eficiència i la precisió en la classificació dels materials reciclables, aquest enfocament innovador té com

• E-mail de contacte: ricardlopezolivares@gmail.com
• Menció realizada: Computació
• Treball tutoritzat per: Carlos García Calvo (Departament de Ciències de la Computació)
• Curs 2022/23

a objectiu maximitzar la recuperació de recursos i reduir la quantitat de residus que acaben en abocadors. En última instància, aquest projecte representa un pas endavant cap a un futur més conscient i responsable pel que fa a la gestió dels recursos naturals i la cura del medi ambient.

El projecte el podem dividir en tres grans parts, aquestes són:

- Robòtica: En aquest apartat trobaríem el robot utilitzat, Säubli TX60 [1], amb la seva controladora, CS8C [2], una pinça pneumàtica i el simulador, Stäubli robotics suite 2022 [3].
- Sistema de visió: Aquest apartat és el que intenta que el robot pugui classificar els objectes obtinguts. L'apartat consta de la xarxa neural empleada, DenseNet-121 [4] i càmera d'un telèfon mòbil, Iphone XR [5].
- Sistemes industrials: En aquest apartat trobem una cinta per transportar els objectes d'un punt a un altre, amb un motor trifàsic, un PLC 1200 AC, un variador de velocitat de la marca Omron MX2-A2002 i un sensor d'objectes de la marca Omron E3FA-DP12.

2 OBJECTIUS

En el projecte tenim dos objectius principals, el primer és crear una simulació de tot el treball final, aquesta s'ha de semblar el més possible a la realitat, el segon és la creació d'aquest projecte en un àmbit real.

Com aquests dos objectius són massa general/globals, els que s'ha fet és dividir aquests en objectius més petits, aquest son: realització d'identificació d'objectes, buscar un programa simulador que ens permeti recrear el món real, crear l'escenari de la simulació, utilització d'una pinça pneumàtica, familiaritzar-se amb el llenguatge i codificació d'un robot industrial, connexió entre les diferents parts i enviar dades entre elles.

(Consultí l'annex A.1 per veure més informació sobre els objectius.)

3 PLANIFICACIÓ

En aquest projecte s'ha decidit fer una divisió en tasques els objectius, ja que no tots els objectius formen part del projecte final, sinó que són passos necessàries per poder arribar al producte final. D'aquesta manera el projecte ens proporciona una flexibilitat de fer les tasques sense que depengui una amb un altre i paral·lelitzar unes amb un altre, és a dir, fer més d'una tasca al mateix temps.

Un cop tenim totes les tasques vaig plantejar utilitzar una metodologia àgil o agile, la qual s'adapta a les necessitats del projecte en aquest punt, la metodologia aplicada s'anomena Kanban [6], aquesta consisteix a tenir un taulell amb diferents columnes, i totes les tasques se situen a la primera columna i aquestes només es mouran de columna segons el desenvolupament de la tasca, un exemple del taulell es pot veure a la Figura 1.

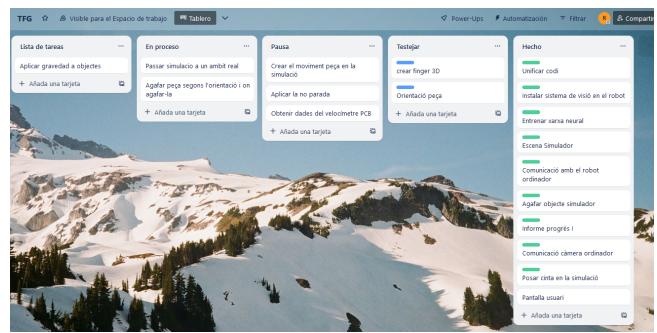


Fig. 1: Exemple de taulell kanban

(Consultí l'annex A.2 per veure més informació sobre la planificació.)

4 ESPECIFICACIONS I TECNOLOGIES EMPLEADES

Per realitzar aquest projecte s'han necessitat diversos components esencials, aquest son:

- **Robot Stäubli TX60:** S'ha decidit aquest robot per les seves característiques i la seva àrea de treball, aquesta última es refereix a tots els punts on el robot pot arribar sense cap mena de problemes, la seva forma és un torus, ja que d'aquesta forma el robot evita la col·lisió amb si mateix. Les següents figures mostren l'àrea de treball.

(Consultí l'annex A.2.1 per veure més informació sobre les característiques del TX60.)

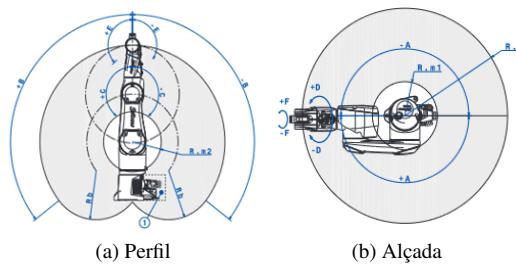


Fig. 2: Àrees de treball TX60

- **Controlador CS8C:** Sempre que utilitzem un robot industrial es necessita una controladora, aquesta s'encarrega d'enviar-li al robot les instruccions que ha de fer, la controladora sempre va amb un comandament manual per controlar el robot manualment.

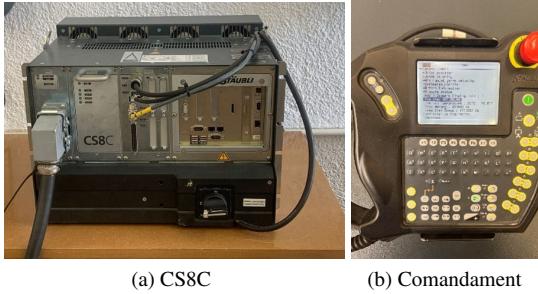


Fig. 3: Controladora CS8C i comandament

- **Simulador Stäubli robotic suite:** Aquest software ens permet treballar únicament amb els robots de l'empresa Stäubli, però d'una manera fàcil, amigable i el més real possible, ja que es poden simular tots els components del robot i entrades del controlador per fer una simulació el més pròxim a la realitat possible, també ens permet programar des de la posada en marxa del robot fins a l'aturada, optimització del rendiment, detecció i previsió de col·lisions, calibració, entre molts altres avantatges. Aquest simulador utilitza el mateix llenguatge que el robot real anomenat, VAL 3 [7].
- **PLC 1200 AC i variador de velocitat Omron Mx2-A2002:** Un PLC és un dispositiu electrònic utilitzat en l'automatització industrial per a controlar i supervisar altres sistemes i processos a temps real. Aquest tenen uns seguits d'entrades i sortides, poden ser digitals o analògiques, que es poden configurar per rebre informació de sensors o dispositius externs. Hi ha diferents estils de llenguatges per a un PLC, en aquest cas he fet servir un llenguatge de programació en blocs funcionals amb el programa TIA PORTAL[17].

Per altra banda, tenim el variador de velocitat Omron Mx2-AC2002, aquest dispositiu ens permet modificar la velocitat d'un motor trifàsic que dona un rang de potència d'entre 0.4 KW a 15 KW.



Fig. 4: PLC 1200 AC i variador de velocitat Omron Mx2-AC2002

- **Sensor d'objectes Omron E3FA-DP12:** És un dispositiu dissenyat per detectar la presència o absència d'objectes en aplicacions industrials. Utilitzant la tecnologia de detecció per reflexió difusa, aquest sensor és capaç de detectar objectes sòlids en una varietat d'entorns i condicions.

• **Xarxa Neural:** La xarxa neural és qui dóna la intel·ligència de classificar en aquest projecte, aquesta s'ha implementat mitjançant TensorFlow [8]. TensorFlow és una biblioteca de codi obert utilitzada per l'aprenentatge automàtic d'intel·ligència artificial. Proporciona un conjunt d'eines i funcions per construir i entrenar models de xarxes neuronals. Tensorflow és molt emprat en aplicacions de processament d'imatges, llenguatge natural, etc.

- **Socket i IvCAM:** Les connexions entre dispositius són essencials per poder transmetre dades entre aquests, en aquest projecte es poden veure 2 tipus de connexions:

– **Sockets** [9]: Aquest tipus de connexió ens permet la comunicació entre processos/dispositius a través d'una xarxa, la connexió funciona amb una estructura client - servidor. La connexió empleada en el projecte són sockets TCP.

– **IvCAM** [10]: Aquesta connexió ens permet transferir dades d'imatges en temps reals. Per ajudar amb aquesta tecnologia hi ha un programa amb el mateix nom el qual ens ajuda a connectar la càmera d'un dispositiu a un altre, aquests dispositius han d'estar en la mateixa xarxa.

- **GitHub** [11]: A l'hora de fer un projecte, ajuda molt tenir un control sobre les versions, veure un historial de canvis i poder anar enrere sempre que en necessiti, per tant, GitHub ha sigut una eina bastant útil en el desenvolupament del projecte.

5 DESENVOLUPAMENT

En aquesta secció s'explica amb més deteniment, com s'ha desenvolupat el projecte durant aquest temps i per quines fases ha passat el mateix.

També cal mencionar que algunes d'aquestes fases es dividiran en més subseccions, pel fet que el projecte té 2 objectius finals (una simulació i un producte físic) o perquè és una secció bastant global.

5.1 Inici i planificació

En primer lloc, s'ha determinat l'enfoc del projecte i en l'àmbit en el qual està enfocat amb la proposta. Tenint en compte els elements que tenim a l'abast.

Un cop decidit l'àmbit en el qual es desenvolupa el projecte vaig fer una identificació d'objectiu, tal com s'explica en la secció 2 d'aquest document. A continuació, totes les tasques que han sortit es posen en un taulell i es va seguir la metodologia kanban, en cas que surti alguna tasca a posteriori o es vulgui eliminar, no afectaria, al ser una metodologia àgil es poden afegir o eliminar tasques sense cap problemes, però sempre seguint els principis de kanban.

Un cop tinc la vista global amb la majoria de les tasques que surten en un inici, vaig buscar eines i tecnologies que puguin resoldre les tasques a fer, sempre amb un mínim de dues tecnologies, d'aquesta forma si la primera no dona resultats tinc la segona i així evito fer tantes recerques per buscar de noves.

5.2 Primeres passes

En l'apartat anterior he mencionat que vaig buscar informació sobre les tasques per poder-les realitzar, i em vaig trobar que en algunes parts importants del projecte no es podia aplicar coneixements adquirits anteriorment, una situació d'aquest estil és amb el robot TX60 que funciona amb un llenguatge anomenat Val 3, no es pot fer servir un altre llenguatge. Per tant, el que he fet abans de començar amb el projecte va ser una fase de preparació i adquisició de coneixements nous.

Per obtenir aquests coneixements nous, he dut a terme diferents programes, llegir i entendre codi d'altres projectes (sobretot projectes del tutor, ja que és un llenguatge industrial i les empreses no pengen codi a internet), l'exemple que uniria totes les proves tracta d'una connexió socket amb el PLC - robot - ordinador, el qual mostra per pantalla els valors obtinguts i també es mou el robot.

(Consultí l'annex A.3 per veure el codi utilitzat en la prova de comunicació entre Stäubli i ordinador.)

5.3 Classificador d'objectes

En aquest apartat s'explica la xarxa neural empleada amb el dataset que s'ha utilitzat.

5.3.1 Dataset

El dataset o conjunt de dades, és el més important a l'hora de fer un aprenentatge en una intel·ligència artificial. Aquest ha d'estar ben construït, ha de ser representatiu i ha de tenir la màxima qualitat possible per poder avaluar, entrenar i testejar models d'aprenentatge automàtic, per tant, s'ha de proporcionar els exemples necessaris perquè el model aprengui patrons i pugui generalitzar les dades, d'aquesta forma podrà ser un model capaç d'afrontar noves dades i ser el més precís i confiable possible.

Per crear el dataset que s'utilitza en el projecte, s'ha agafat 2 datasets ja creats, amb moltes imatges d'objectes per a reciclar, com aquest dos datasets, estan estructurats de manera diferent, i molts dels productes, no es poden obtenir en un àmbit europeu, he optat per fer una combinació dels dos. Aquest dos datasets són:

- **Trashnet** [12]: Aquest dataset consta de 2527 imatges, les quals estan classificades en 6 classes, aquestes són: vidre, paper, cartó, plàstic, metall i productes sanitaris.
- **Waste classification data** [13]: És un dataset que conté 22500 imatges, classificades en dos grups aquest són: orgàniques i reciclables.

La combinació dels datasets va donar com a resultat un dataset de 6283 imatges, les quals estan dividides en 4683

entrenament, 800 validació i 800 test. Totes aquestes seccions es divideixen en els colors dels contenidors, groc, verd, blau i marró, en tots els apartats anterior s'ha intentat que tinguin el mateix nombre de dades, d'aquesta manera s'evita a descompensació de les classes. Les imatges seleccionades dels datasets abans mencionat, són de diferents posicions, mides, colors, formes, contextos, això pel fet que a l'hora de classificar ens donarà millor resultat.



Fig. 5: Imatges d'exemple contenidor groc del dataset



Fig. 6: Imatges d'exemple contenidor blau del dataset



Fig. 7: Imatges d'exemple contenidor verd del dataset



Fig. 8: Imatges d'exemple contenidor marró del dataset

5.3.2 Xarxa neural

La xarxa neural és una part molt important del projecte, i el primer contacte amb la visió per computador del projecte. Aquesta és essencial per donar la intel·ligència al robot per poder classificar objectes.

Per aconseguir-ho he utilitzat una xarxa neural convolucionals (CNN) que forma part de la família DenseNet, aquesta xarxa es caracteritza pel seu disseny dens i connectivitat directa entre les capes. La xarxa utilitzada és una DenseNet-121, una xarxa preentrenada amb la qual es divideix en dues parts:

- Pooling: És el nom que es dona a l'agrupació de diverses capes convolucionals.

- Unitats denses: Aquestes es componen de capes convolucionals en cascada, on cada capa rep connexions directes de totes les capes anteriors.

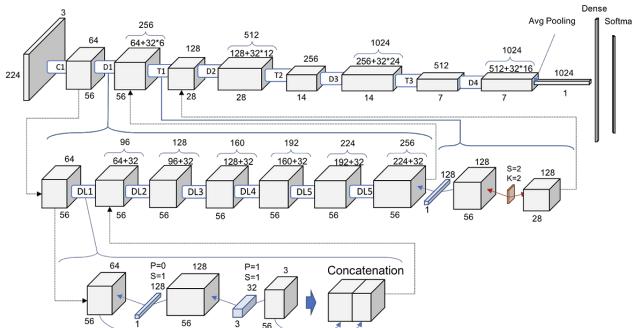


Fig. 9: Estructura d'una xarxa DenseNet

DenseNet-121 té un total de 121 capes, aquí s'inclouen les capes de convolució, agrupació i les capes completamente connectades. A mesura que les dades flueixen per la xarxa es van extraient característiques a diferents nivells d'abstracció , el qual ens permet obtenir bastant detall i característiques a alt nivell de les imatges d'entrada.

S'ha optat per aquest tipus de xarxa, pel fet que té un alt rendiment en les tasques de classificació d'imatges i no només en la classificació sinó que també ha donat molt bons resultats en detecció d'objectes, reconeixement d'objectes i en la segmentació semàntica.

5.3.2.1 Entrenament

A l'hora d'entrenar la xarxa neural s'ha de tenir molts conceptes a tenir en compte, aquest poden ser opcionals com és el cas del preprocessament d'imatges..., i d'altres que són obligatoris, però són difícils de trobar com els hiperparàmetres *batch*, *epoch* i *steps*.

Per evitar un sobreentrenament de la xarxa, les imatges passen per un preprocessament on els realitza una augmentació de les dades, una rotació, un zoom, un redimensionament i un flip horitzontal. Amb això aconsegueixo una generalització i abstracció, pel fet que la xarxa rep una mateixa imatge de diferents formes i serà capaç de generalitzar millor i, per tant, classificarà millor una imatge que mai hagi vist.

Per altra banda, també he hagut de modificar els pesos de les capes 34, ja que és una xarxa preentrenada i els pesos predefinits per defecte, no servien en aquest cas, la modificació dels pesos és automàtica mitjançant aquesta aprenent.

(Consultí l'annex A.4 per veure el codi utilitzat a l'hora de crear la xarxa neural.)

5.4 Orientació d'objecte

L'orientació d'objectes és una eina en la visió per computador per determinar en quina direcció estava un objecte en una imatge o vídeos. Aquest tipus de codi s'enfoca a identificar i classifica l'orientació dels objectes detectats.

Com he mencionat abans, la intenció de l'algorisme es detecta l'orientació de l'objecte, en aquest projecte n'és útil per saber en quina orientació el robot ha d'agafar el mateix, així doncs pot agafar la majoria dels objectes que se li presentin.

A grans trets, l'algorisme fa un blur a la imatge recollida i a una imatge del fons (imatge de l'escena sense cap objecte), seguidament es fa una resta de les imatges per veure les diferències, la diferència serà l'objecte introduït a la cinta, per acabar és binaritzat i es fa una erosió i dilatació, d'aquesta manera podem obtenir la silueta de l'objecte, un cop tenim aquesta imatge processada es busca el contorn i es calcula l'angle.

(Consultí l'annex A.5 per veure el codi utilitzat a l'hora d'obtenir l'orientació d'un objecte.)

5.5 Simulació

Tal com es menciona en els objectius, un dels objectius finals és la creació d'una simulació en el programa Stäubli Robotic Suite 2022 (SRS). Aquest software ens permet treballar únicament amb els robots de l'empresa Stäubli, però d'una manera fàcil, amigable i el més real possible, ja que es poden simular tots els components del robot i entrades del controlador per fer una simulació el més pròxim a la realitat possible, també ens permet programar des de la posada en marxa del robot fins a l'aturada, optimització del rendiment, detecció i previsió de col·lisions, calibració, entre molts altres avantatges. Aquest simulador utilitza el mateix llenguatge que el robot real (VAL 3), d'aquesta manera es pot fer un port directe sense haver de transcriure el codi.

L'escena creada conta de diversos objectes 3D, en la qual trobem el robot, el manipulador, un cub (simula un objecte), papereres de diferents colors i una taula. L'escena intenta que sigui el més fiable a la realitat possible, tenint en compte les dimensions, etc. No només intenta reflectir la realitat en l'escena, sinó que en la funcionalitat també, és a dir, aquesta simulació té les mateixes prestacions que el codi pel robot físic, com la paral·lelització dels codis, comunicació amb els altres elements del projecte, etc.

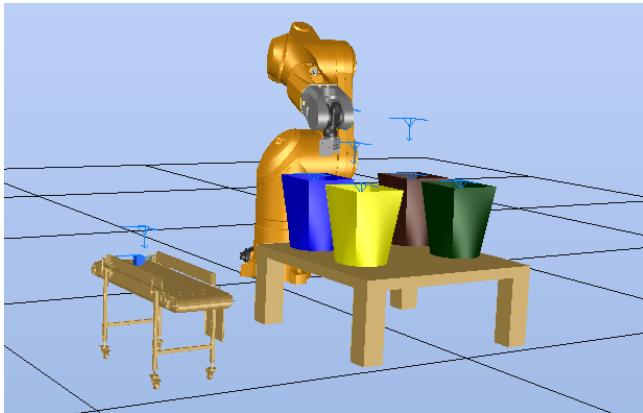


Fig. 10: Escena del simulador Stäubli Robotic Suite

5.6 Món físic

Anomenaré nom físic al conjunt d'elements físic que es poden manipular físicament, és a dir, el món real. El món físic és un món complicat per a visió per computador, ja que si no tenim un entorn molt controlat, pot sorgir problemes a l'hora d'obtenir una imatge i s'hauria de realitzar un preprocessat a la imatge obtinguda, si això passa es perd bastant de temps en arreglar la imatge, pel fet que s'ha de buscar un preprocessat que funcioni amb la majoria d'objectes, per evitar això he intentat crear un entorn el més ideal possible. Per aconseguir aquest entorn ideal he dir a terme diversos canvis en l'escena física.

El primer canvi es tracta de la posició d'on es capta la imatge de l'objecte, en aquest punt tenia diverses opcions, però he optat en implementar la més versàtil i estable de totes, aquesta tracta de crear una estructura independent, la qual es pugui acoblar al lateral de la cinta. Aquesta és una composició d'elements metàl·lics amb una impressió 3D al final. Els elements metàl·lics són uns tubs, dissenyats per acoblar i desacoblar estructures el més ràpid possible.

L'estructura té forma de 'C', amb aquesta forma el robot pot entrar dintre de l'estructura a l'hora d'agafar l'objecte de la cinta. En el punt més alt de l'estructura, tenim una impressió 3D la qual em permet col·locar el telèfon mòbil, que aquest es qui s'encarrega de captar la imatge de l'objecte.

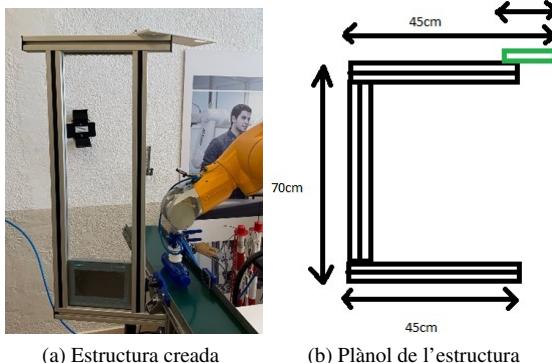


Fig. 11: Estructura

El segon canvi que he realitzat és amb la llum, a causa

del fet que és una habitació molt il·luminada entrava molta claredat i molta llum, la claredat ens va bastant bé, així la imatge surt més nítida i es pot identificar millor l'objecte, en canvi, la llum no és tan bona idea que passi aquesta pot ocasionar reflexos i ombres, les quals poden ser perjudicials a l'hora de classificar l'objecte. Per solucionar aquest problema es va posar una tela blanca bastant porosa, així la llum entrant és una llum més tenua i es manté la claredat de la sala.

Per acabar, vaig canviar d'eina en el robot, pel fet que una vàlvula de succió, és l'eina que hi havia abans, no seria capaç d'agafar la majoria d'objectes, només pel fet que la vàlvula agafa objectes relativament plans, en canvi, una ampolla de plàstic no la podria agafar, ja que és un cilindre i quedarien forats buits per on sortiria l'aire, així que vaig canvia d'eina, aquesta és una pinça que es compon de dues estructures, la primera és un gripper [14] que es va demanar a l'empresa Stäubli i el segon és una impressió 3D per tindre una major distància i millor adherència.

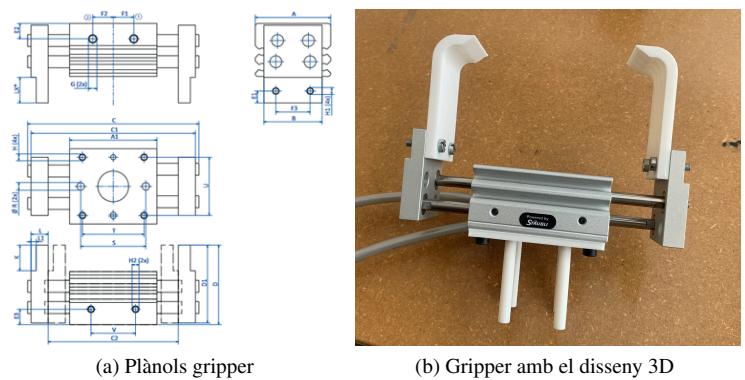


Fig. 12: Pinça

(Consultí l'annex A.7 per veure les mides del gripper i el plànol de la impresió 3D.)

5.7 Connexió entre components

Una part important del projecte és la connexió entre els dispositius, aquestes connexions són important per enviar dades entre ells i fer diferents accions depenen d'aquestes, en aquest projecte he utilitzat 2 tipus de connexions, com he comentat en l'apartat 4, aquestes són, els sockets i IvCAM.

La primera és la connexió socket, aquest tipus de connexions es fan servir a l'hora d'enviar dades entre dos dispositius, ja que són connexions bidireccionals, amb una estructura client-servidor i en cas de fer servir els sockets/TCP, com en aquest projecte, tens una garantia que la dada enviada és la correcta i que el destinatari ha rebut el missatge. Aquest tipus de connexió està implementada en dos llocs del projecte, aquest són:

- PLC- Controlador:** EL PLC envia una dada, dient si hi ha un objecte o no a la cinta, la dada enviada és un caràcter en ASCII, concretament una D (valor 100 en la taula ascii), indicant que hi ha un objecte i una C (valor 99 en la taula ascii), indicant el contrari. El controlador no envia res al PLC.
- Controlador - Ordinador:** El controlador emet un senyal, per dir que està en funcionament, un cop està la connexió establerta, els dos components s'esperen fins que hi hagi un objecte a la cinta, un cop sindica que hi ha un objecte (la controladora ha rebut un 100, mitjançant la connexió anterior), aquest dos intercanviaran dades per indicar a quin contenidor va l'objecte. El valor que indica el contingut és un valor ascii (48,49,50 o 51), representant (groc, blau, verd o marró respectivament).

El segon tipus de connexió és una connexió IvCam, la qual ens permet intercanviar un gran cùmul de dades entre una càmera i un dispositiu mitjançant la tecnologia sense fils. Aquesta connexió s'utilitza només quan s'ha de realitzar la fotografia, és a dir, només la podem veure en la connexió de la càmera amb l'ordinador. Per dur a terme aquesta connexió s'ha fet servir el programa IvCAM d'esesoft, per fer que funcioni el programa només s'ha d'instal·lar en els dos dispositius, els dos dispositius han d'estar connectats a la mateixa xarxa, i obrir el programa en els dos dispositius per portar a cap la connexió.

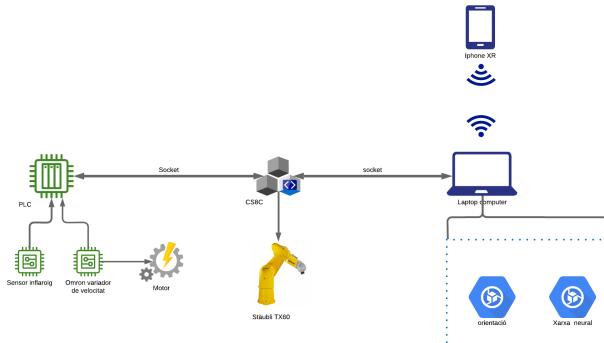


Fig. 13: Diagrama de connexions

6 TESTS

En aquest apartat es enumeron tots els testos que s'ha arribat a fer en aquest projecte.

6.1 Xarxa neural

En la xarxa neural he fet diversos tipus de proves, aquestes proves no consisteixen en funcions de testeig, ja que en una xarxa neural no té gaire sentit posar un testing del codi, però sí del resultat que dona, per tant, el testeig realitzat és sobre els resultats que retorna la xarxa, aquest s'han realitzat en diferents espais de temps que ha durat el projecte. Els testos es poden dividir en tres grans grups.

- El primer és el menys rellevant, ja que és un test amb imatges del propi dataset, ja sigui de l'apartat de test, validació o entrenament. Realment, d'aquests testos no es poden treure gaires conclusions, pel fet que la xarxa neural ja ha vist dos d'aquest grup per aprendre.
- La segona prova amb la xarxa neural va ser amb imatges externes al dataset, però aquests imatges van ser tractades per només veure l'objecte que m'interessa, el fons aplicat és un color uniforme, més o menys del mateix color de la cinta. Aquests testos són més interessants per extreure conclusions i comprovar el real funcionament de la xarxa.

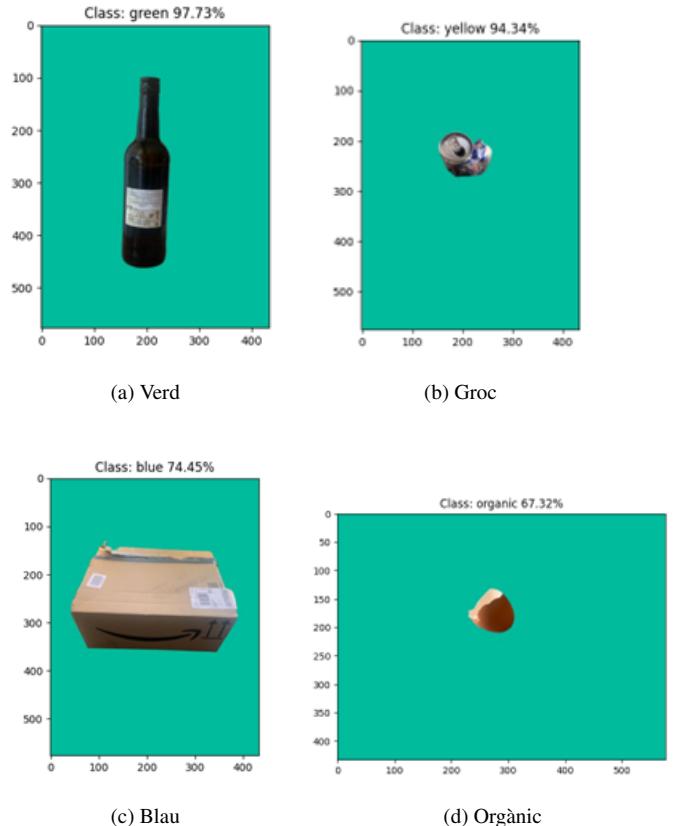
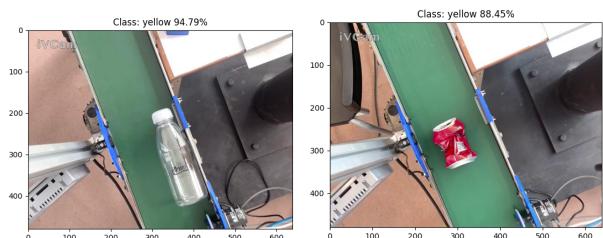


Fig. 14: Test objectes que no estan en el dataset amb fons uniforme

- La tercera i última prova és amb objectes sense un fons uniforme, aquestes proves es van fer per veure la reacció de la xarxa en passar-li una imatge amb un fons bastant diferent entre si, i comprova que reconeix de la mateixa manera l'objecte a classificar.



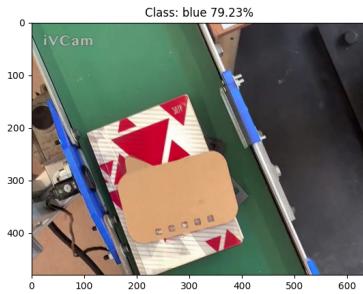


Fig. 15: Test objectes que no estàn en el dataset amb fons

6.2 Orientació d'objectes

Per testejar l'orientació d'objectes, he seguit el mateix procediment que en l'apartat anterior, és a dir, he fet la comprovació per veure el funcionament amb imatges de prova, aquestes imatges són extretes directament de com és l'esceina final, per tant, es pot veure l'objecte sobre la cinta.

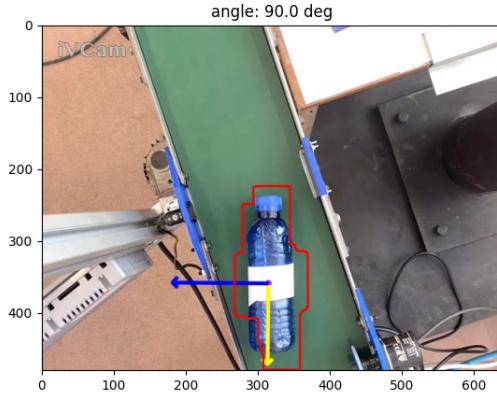


Fig. 16: Test Orientació

7 RESULTATS

7.1 Xarxa neural

El resultat de la xarxa neuronal s'ha evaluat utilitzant una matriu de confusió, una eina fonamental per analitzar el rendiment del model de classificació. La matriu de confusió proporciona una visió detallada de les prediccions del model en cada classe objectiu. S'ha observat, que la classificació és bastant correcte, tenint en compte els falsos positius i falsos negatius, això indica que la xarxa pot classificar imatges sense problemes.

En particular, s'ha de destacar la classe "organic" la classe "green" aquestes són les classes que tenen millors resultats a l'hora de classificar, ja que es confonen bastant poc.

No obstant això, s'ha identificat una major dificultat en la classificació de la classe "yellow". Aquesta classe en particular requereix més atenció i possiblement es beneficiaria d'un enfocament d'entrenament addicional o la inclusió de més dades representatives, pel fet que la classe se confon amb la classe "green", dedueixo, que és perquè tots dos són objectes transparents, de colors i poden tenir

formes bastants semblants.

En general, la matriu de confusió reflecteix un rendiment bastant sòlid del model en la tasca de classificació d'imatges.

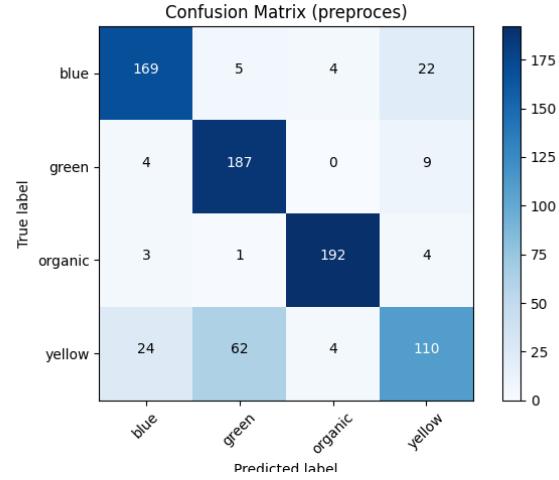


Fig. 17: Confusion Matrix

7.2 Simulació i món físic

El resultat del projecte en aquest àmbit és similar en els dos casos, ja que són més o menys complementàries i es pot dir que compleix la funció esperada seguint un flux, aquest és el següent: s'inicia el programa de la xarxa neural i el robot, un cop els dos encesos el robot es mourà fins a arribar a un punt, on esperarà que el PLC mira si hi ha un objecte i envia el resultat, de si ha detectat un objecte o no. A continuació farà dues tasques al mateix temps, la primera és tramestre a l'ordinador (on escroba tot el codi amb la xarxa neural) un senyal indicant que pot fer una fotografia, aquests li tramestrarà el senyal al telèfon mòbil i es realitzarà la fotografia i la guardarà, aquesta serà processada en la xarxa neural i per últim remetrà el resultat, la segona tasca que es realitza en paral·lel és el moviment del robot aproximant-se a la cinta per agafar l'objecte, aquest l'agafarà i espera en un punt a sobre de les papereres, esperant a rebre el resultat de la xarxa neural, un cop el rebi, deixarà el robot a la paperera corresponent i tornarà a començar.

El flux abans comentat el podem veure a la següent figura, figura 18.

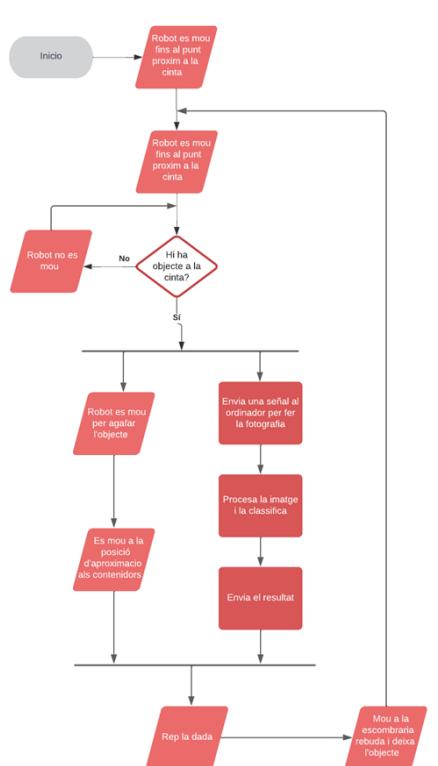


Fig. 18: Flux de tot el projecte

(Consultí l'annex A.6 per veure el codi implementat.)

8 CONCLUSIONS I MILLORES

Gràcies a aquest projecte s'ha pogut descobrir un gran potencial en la Visió per computador, aplicant-la a un món que cada cop agafa més força com és la indústria 4.0. La xarxa neuronal DenseNet-121 és un clar exemple que es pot aplicar xarxes neuronals a molts àmbits, ja que no són gens rígides/enfocades a un únic treball, sinó que és una eina molt flexible i amb un potencial enorme, si volem un exemple, aquest projecte n'és un, en el qual s'ha aplicat una xarxa neural, que pot classificar en 4 classes, però afegint imatges al dataset sobre noves classes la mateixa xarxa podria ser una bona opció, només faria falta que tornés a aprendre.

També he pogut experimentar la importància d'una metodologia àgil i tenir un pla B, segurament hi ha projectes que no és tan important, però en aquest en concret ha sigut bastant crucial el fet de poder anar avançant mentre alguna altra tasca no estava completa, un exemple d'això és el gripper el qual va triga bastant tard a l'hora de rebre'l, no obstant es podia continuar avançant per un altre camí i les proves que ha calgut fer, s'han fet amb una altra eina com un succionador, que encara no sigui el mateix em permetia poder avançant i fer les proves necessàries.

Com a millora personal en aquest projecte el que faria és l'obtenció de l'objecte sense que la cinta hagi d'aturar-se, el qual estava previst d'implementar, però no s'ha pogut

realitzar pel fet que la targeta que va acoblada al CS8C i pugui llegir un encoder, no ha arribat. Un altre millora que es podria fer és afegir diversos sensors, com sensors de pes..., splits per posar els objectes tan centrats com sigui possible a la cinta, etc.

Per acabar, voldria mencionar que això té un propòsit com a eina per ajudar al reciclatge, ja que vivim en una època plena de residus, i els recursos no són il·limitat, per tant, amb aquest projecte s'intenta millorar la sostenibilitat mediambiental i ajudar a les plantes de residus a poder reciclar.

AGRAÏMENTS

Vull expressar el meu més sincer agraïment al meu tutor/a, Carlos García Calvo, per la seva valuosa orientació i suport al llarg d'aquest projecte de reciclatge. La seva experiència, coneixements i dedicació han estat fonamentals per al meu creixement i èxit en aquesta iniciativa. Estic profundament agraït per la seva guia constant, la seva disposició per resoldre els meus dubtes i la seva motivació. La seva influència ha deixat una empremta en el desenvolupament projecte, i estic sincerament agraït per l'oportunitat de treballar sota la seva tutela.

REFERÈNCIES

- [1] “Stäubli tx60.” [Online]. Available: <https://robotsoneright.com/Staubli/staubli-TX60.html>
- [2] Stäubli, “Cs8c controller instructions manual,” ./Documents/CS8C Controller.pdf.
- [3] “Stäubli robotics suite 2022.” [Online]. Available: <https://www.staubli.com/hk/en/robotics/products/robot-software/staubli-robotics-suite.html>
- [4] “Densenet keras.” [Online]. Available: <https://keras.io/api/applications/densenet/>
- [5] “Características iphonexr.” [Online]. Available: <https://support.apple.com/kb/SP781>
- [6] “Kanban.” [Online]. Available: <https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban>
- [7] “Val3.” [Online]. Available: <https://usermanual.wiki/Document/val3referencemanual.275627616>
- [8] “Tensorflow.” [Online]. Available: <https://www.tensorflow.org>
- [9] “sockets.” [Online]. Available: <https://www.ibm.com/docs/es/aix/7.3?topic=concepts-sockets>
- [10] “Ivcam.” [Online]. Available: <https://www.e2esoft.com/ivcam/>
- [11] “Github.” [Online]. Available: <https://github.com/Ricardlol/Recycling-robot>

- [12] “trashnet.” [Online]. Available:
<https://github.com/garythung/trashnet>
- [13] “waste classification data.” [Online]. Available:
<https://www.kaggle.com/datasets/techsash/waste-classification-data>
- [14] Stäubli, “Especificación gripper,”
[./Documents/grippers-en.pdf](#).

APÈNDIX

A.1 Objectius

CODI	DESCRIPCIÓ
[O1]	Obtenir informació important sobre el robot TX-60 i simulador.
[O2]	Realitzar la identificació d'objectes.
[O3]	Buscar informació sobre la programació del robot i el simulador.
[O4]	Crear una simulació realista del projecte.
[O5]	Realitzar que la simulació funcioni en el món real i faci el circuit (agafar de la cinta, traslladar-ho al lloc).
[O6]	Manipulació d'un objecte estàtic, en simulació com en espai real.
[O7]	Realització d'agafar un objecte no estàtic.
[O8]	Comunicar un ordinador amb el robot, per poder enviar-li accions a realitzar.
[O9]	Utilització d'una pinça pneumàtica
[O10]	Creació de les extensions de la pinça per poder agafar objectes.

A.2 Especificacions

A.2.1 Staubli Tx60 caracteristiques

Marca	Stäubli
Model	TX60
Tipus	Braç robotic (braç antropomòrfic)
Axes (joins)	6
Carrega	3,5kg
Eixos	600 mm
Repetibilitat	0.02 mm
Pes	51 kg

A.3 Primeres pases

A.3.1 Codi ordinador python

```

1 import socket
2
3 sock = socket.socket(socket.AF_INET, socket.
4     SOCK_STREAM)
5 server_address = ('XXX.XXX.XXX.XXX', XXXX)
6 sock.connect(server_address)
7 message = ''
8
9 try:
10     message = b'0'
11     print('sending {!r}'.format(message))
12     sock.sendall(message)
13     amount_received = 0
14     amount_expected = len(message)
15     while amount_received < amount_expected:
16         data = sock.recv(16)
17         amount_received += len(data)
18         print('received {!r}'.format(data))
19
20 finally:
21     sock.close()
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

A.3.2 Codi stäubli

```

1 cls()
2 movej(JHome,tTool, mNomSpeed)
3 waitEndMove()
4 movej(JApproach1,tTool, mNomSpeed)
5 waitEndMove()
6 do
7     nTest = sioGet(sSioOrdenador, l_nNumero)
8     sioGet(sSioPLC, l_nNumPLC)
9     delay(5)
10    put("Test connect to computer :")
11    putln(nTest)
12    put(" L_nNumero :")
13    putln(l_nNumero)
14    put(" L_nNumPLC :")
15    putln(l_nNumPLC)
16
17 until l_nNumero == 13

```

A.4 Xarxa Neural

A.4.1 main.py

```

1 import tensorflow as tf
2 from tensorflow.keras.applications.densenet
3     import preprocess_input
4 import classModels
5
6 imageSize = (224, 224)
7 batch_size = 32
8 names_class = ['blue', 'green', 'organic', 'yellow']
9
10 train_path = './dataset/train'
11 test_path = './dataset/test'
12 validation_path = './dataset/valid'
13
14 train_batches = tf.keras.preprocessing.image.
15     ImageDataGenerator(
16         rotation_range=5,
17         rescale=1./255,
18         horizontal_flip=True,
19         zoom_range=0.2,
20     ).flow_from_directory(
21         directory=train_path,
22         target_size=imageSize,
23         classes=names_class,
24         batch_size=batch_size,
25         class_mode='categorical'
26     )
27
28 valid_batches = tf.keras.preprocessing.image.
29     ImageDataGenerator(
30         preprocessing_function=preprocess_input
31     ).flow_from_directory(
32         directory=validation_path,
33         target_size=imageSize,
34         classes=names_class,
35         batch_size=batch_size,
36         class_mode='categorical'
37     )
38
39 test_batches = tf.keras.preprocessing.image.
40     ImageDataGenerator(
41         preprocessing_function=preprocess_input,
42     ).flow_from_directory(
43         directory=test_path,
44         target_size=imageSize,
45         classes=names_class,
46         batch_size=batch_size,
47         shuffle=False,
48         class_mode='categorical'
49     )
50
51 test_batches_nopre = tf.keras.preprocessing.image.
52     ImageDataGenerator(
53         dtype='uint8'
54     )

```

```

47 ).flow_from_directory(
48     directory=test_path,
49     target_size=imageSize,
50     classes=names_class,
51     batch_size=batch_size,
52     shuffle=False,
53     class_mode='categorical'
54 )
55 epoch = 20
56 steps = int(train_batches.samples / 35)
57 val_steps = int(valid_batches.samples / 35)
58
59 secondModel = classModels.modelsDenseNet121(epoch
60     , steps, val_steps, train_batches,
61     valid_batches, test_batches,
62     test_batches_nopre, True)
63
64 secondModel.baseModelDenseNet121()
65 secondModel.layerFalse(34)
66 secondModel.compileModel()
67 secondModel.fit()

```

A.4.2 classModel.py

```

1 import tensorflow as tf
2 from tensorflow import keras
3
4 from keras import regularizers
5 from tensorflow.keras.layers import Flatten,
6     Dense, Dropout, Input
7 from tensorflow.keras import Model
8 from tensorflow.keras.models import load_model
9 from tensorflow.keras.optimizers import SGD
10 from sklearn.metrics import confusion_matrix
11 from tensorflow.keras.applications.densenet
12     import preprocess_input
13 from keras.callbacks import EarlyStopping,
14     ReduceLROnPlateau
15
16 import numpy as np
17
18 class modelsDenseNet121:
19     def __init__(self, epoch, steps, val_steps,
20         train_batches, valid_batches, test,
21         test_prep, optimization=False):
22         self.base_model = ''
23         self.model = ''
24         self.base_predic = ''
25         self.predic = ''
26         self.confusion = ""
27         self.historic = ''
28         self.callbackList = ''
29         self.model_prep = ""
30         self.epoch = epoch
31         self.steps = steps
32         self.val_steps = val_steps
33         self.train_batches = train_batches
34         self.valid_batches = valid_batches
35         self.testData = test
36         self.optimization = optimization
37         self.test_nopre = test_prep
38
39     def __init__(self, nameModel):
40         self.loadModel(nameModel)
41
42     def baseModelDenseNet121(self):
43         self.base_model = tf.keras.applications.
44             DenseNet121(weights='imagenet', include_top=
45             False, pooling='avg')
46
47         x = self.base_model.output
48         x = Flatten()(x)
49         x = Dense(256, kernel_regularizer=
50             regularizers.l1_l2(0.01),
51             activity_regularizer=regularizers.l2(0.01),
52             activation='relu')(x)
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

```

```

        x = Dropout(0.5)(x)
        self.base_predic = Dense(4, activation='softmax')(x)

        self.model = Model(inputs=self.base_model.
            input, outputs=self.base_predic)

    def compileModel(self):
        if self.optimization:
            self.model.compile(optimizer=SGD(
                learning_rate=0.01, momentum=0.9), loss='
                    categorical_crossentropy',
                metrics=['accuracy',
                    'mse'])
            stop = EarlyStopping(monitor='
                val_loss', patience=8, verbose=1, min_delta=1
                e-4)
            reduceLr = ReduceLROnPlateau(monitor='
                val_loss', factor=0.1, patience=2, verbose
                =1, min_delta=1e-6)
            self.callbackList = [stop, reduceLr]

        else:
            self.model.compile(optimizer='adam',
                loss='categorical_crossentropy', metrics=['
                    accuracy', 'mse'])
            stop = EarlyStopping(monitor='
                val_loss', patience=8, verbose=1, min_delta=1
                e-4)
            reduceLr = ReduceLROnPlateau(monitor='
                val_loss', factor=0.1, patience=3, verbose
                =1, min_delta=1e-4)
            self.callbackList = [stop, reduceLr]

    def fit(self):
        self.historic=self.model.fit(
            self.train_batches,
            steps_per_epoch=self.steps,
            validation_steps=self.val_steps,
            validation_data=self.valid_batches,
            epochs=self.epoch,
            callbacks=self.callbackList,
            verbose=1
        )

    def layerFalse(self, num=0):
        if num == 0:
            for layer in self.base_model.layers:
                layer.trainable = False
        else:
            for layer in self.model.layers[:num]:
                layer.trainable = False
            for layer in self.model.layers[num:]:
                layer.trainable = True

    def evaluateModel(self, preproc):
        if preproc:
            self.model.evaluate(x=self.test_nopre,
                verbose=1, steps=self.val_steps)
        else:
            self.model.evaluate(x=self.testData,
                verbose=1, steps=self.val_steps)

    def predictions(self, preproc=False):
        if preproc:
            self.predic = self.model.predict(x=
                self.test_nopre, verbose=1, steps=self.
                val_steps)
        else:
            self.predic = self.model.predict(x=
                self.testData, verbose=1, steps=self.
                val_steps)

```

A.5 Orientació d'objectes

```
1 import numpy as np
2 import cv2
3 import matplotlib.pyplot as plt
4
5 image = cv2.imread("./img/3bfce634-ab0c-4223-84c7
6 -c20df6f6b48d.png")
7 background = cv2.imread("./background.png")
8
9 imgResult = cv2.cvtColor(image, cv2.COLOR_BGR2RGB
10 )
11
12 image.blur = cv2.GaussianBlur(image, (51, 51),
13 cv2.BORDER_DEFAULT)
14 image_bw = cv2.cvtColor(image.blur, cv2.
15 COLOR_BGR2GRAY)
16
17 backblur = cv2.GaussianBlur(background, (51, 51),
18 cv2.BORDER_DEFAULT)
19 back_bw = cv2.cvtColor(backblur, cv2.
20 COLOR_BGR2GRAY)
21
22 diff = cv2.absdiff(image_bw, back_bw)
23 ret, mascara = cv2.threshold(diff, 30, 255, cv2.
24 THRESH_BINARY)
25
26 kernel_1 = np.ones((5, 5), np.uint8)
27 mascara = cv2.erode(mascara, kernel_1, iterations
28 =5)
29 mascara = cv2.dilate(mascara, kernel_1,
30 iterations=2)
31
32 kernel_2 = np.ones((7, 7), np.uint8)
33 mascara = cv2.dilate(mascara, kernel_2,
34 iterations=20)
35 mascara = cv2.erode(mascara, kernel_2, iterations
36 =10)
37
38 contours = cv2.findContours(mascara, cv2.
39 RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
40
41 rotrect = cv2.minAreaRect(contours[0])
42
43 cv2.drawContours(imgResult, contours, 0, (255,0,0)
44 ,2)
45
46 angle = rotrect[-1]
47 if angle < -45:
48     angle = -(90 + angle)
49 else:
50     angle = angle
51
52 return angle
```

```

15 names_class = ['blue', 'green', 'organic', '
    yellow']
16 model = classModels.modelsDenseNet121("modelUse")
17
18 takeFoto = False
19 stopProgram = False
20
21 if data == b'I':
22     while not(stopProgram):
23         while not(takeFoto):
24             data = sock.recv(16)
25             if data == b'R':
26                 takeFoto = True
27             leido, frame = cap.read()
28
29             if leido:
30                 nameFoto = './img/' +str(uuid.uuid4())
31             ) + ".png"
32             cv2.imwrite(nameFoto, frame)
33             angle = orientationDetector.
34             getOrientation(nameFoto)
35             sock.sendall(angle)
36
37             resultClass = model.classifyImage(
38                 names_class, nameFoto)
39             sock.sendall(resultClass)
40
41
42             else:
43                 stopProgram = True
44             takeFoto = False
45
46 cap.release()
47 sock.close()

```

A.6.2 codi prncipal controladora

```

1 begin
2     movej(JHome, tToolSuccion, mNomSpeed)
3     waitEndMove()
4     movej(JApproach1, tToolSuccion, mNomSpeed)
5     waitEndMove()
6     do
7         movev(pTable, tToolSuccion, mNomSpeed)
8         waitEndMove()
9         sioGet(sPlc, nRecvPLC)
10        if nRecvPLC == 100
11            sioSet(sSocket, 82)
12            movej(jAproachCinta, tToolSuccion,
13                  mNomSpeed)
14            waitEndMove()
15            movej(jGetCinta, tToolSuccion, mNomSpeed)
16            waitEndMove()
17            close(tToolSuccion)
18            waitEndMove()
19            movev(pApprBins, tToolSuccion, mNomSpeed)
20            waitEndMove()
21            nConnection = sioGet(sSocket, nRecvData)
22        switch nRecvData
23            case 48
24                movev(pBinYellow, tToolSuccion, mNomSpeed)
25            )
26                waitEndMove()
27                break
28            case 49
29                movev(pBinBlue, tToolSuccion, mNomSpeed)
30                waitEndMove()
31                break
32            case 50
33                movev(pBinGreen, tToolSuccion, mNomSpeed)
34                waitEndMove()
35                break
36            case 51
37                movev(pBinBrown, tToolSuccion, mNomSpeed)
38                waitEndMove()
39                break
40            default

```

A.6 Codi final

A.6.1 Codi principal ordinador

```
1 import classModels
2 import orientationDetector
3 import cv2
4 import uuid
5 import socket
6
7 tcpsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9 server_address = ('XXX.XXX.XXX.XXX', XXXX)
10 sock.connect(server_address)
11
12 data = sock.recv(16)
13 cap = cv2.VideoCapture(0)
14
```

```

39     putln("Bin selected no correct")
40     break
41 endSwitch
42   open(tToolSuccion)
43   waitEndMove()
44   nRecvPLC=0
45   clearBuffer(sPlc)
46   endIf
47   until 1==2
48 end

```

A.7 Pinça

A.7.1 Mides gripper

A [mm]	A1 [mm]	B [mm]	C [mm]	C1 [mm]	C2 [mm]	D [mm]	D1 [mm]	E1 [mm]	E2 [mm]	E3 [mm]	F1 [mm]	F2 [mm]
44	67	34	142	138	98	46	45.5	7	9	9	19.5	19.5

F3 [mm]	G [mm]	H [mm]	H1 [mm]	H2 [mm]	K [mm]	L [mm]	L1 [mm]	R [mm]	S [mm]	T [mm]	U [mm]	V [mm]
20	M5	M4	M4	M4	15	10	3	4.5	54	52	34	42

A.7.2 Mides de l'impresió 3D

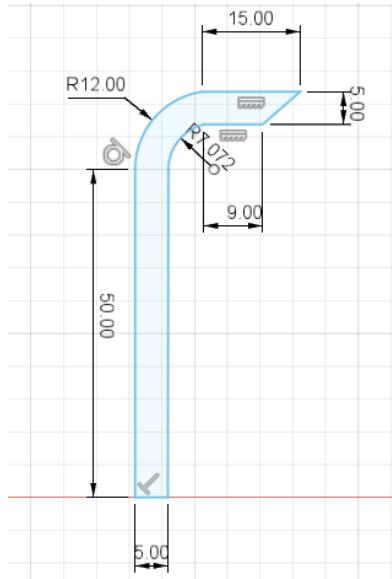


Fig. 19: Mides (en milímetres) de l'estruktura 3D