Universitat de Vic
Escola Politècnica Superior

Treball de Fi de Grau

Interacció operari-robot: Sistema de control mitjançant ROS-Industrial i visió 3D

Míriam Laliga Cervilla

Grau en Enginyeria Mecatrònica

Grau en Enginyeria Electrònica Industrial i Automàtica

Tutor: Gerard Masferrer Caralt

Vic, Juny de 2015

Índex

Índex de figures

Universitat de Vic
Escola Politècnica Superior

RESUM TREBALL FINAL DE GRAU

GRAU EN ENGINYERIA MECATRÒNICA
GRAU EN ENGINYERIA ELECTRÒNICA INDUSTRIAL I AUTOMÀTICA

Títol: *Interacció operari-robot: Sistema de control mitjançant ROS-Industrial i visió 3D*
Paraules clau: *ROS, Robòtica, Matlab, PointCloud, Kinect, Robòtica col·laborativa*
Autora: Míriam Laliga Cervilla
Tutor: Gerard Masferrer Caralt (UVIC-UCC)
Data: Juny de 2015

La indústria de la robòtica Europea necessita solucions competitives per assolir i mantenir el lideratge mundial en productes i serveis de robòtica. L'escenari d'aplicació és obert per fomentar la creativitat en relació amb el desenvolupament d'aplicacions innovadores de fabricació que involucren la col·laboració entre humans i robots.

En aquest treball, un robot industrial de 6 eixos està equipat amb una rebladora i una càmera estèreo. Un objecte es col·loca sobre la taula a la zona de treball del robot (l'objecte té forats per posar reblons). Una càmera 3D està muntada sobre del lloc de treball i una segona càmera 3D està muntada per capturar l'aproximació de l'operari.

L'objectiu del projecte és realitzar una operació (posar reblons) als forats assenyalats per l'operari. L'objectiu consta de tres sub-tasques: 1) Reconèixer el gest d'assenyalar, 2) Localitzar la peça en base a les dades CAD i 3) Localitzar la posició on l'operari està assenyalant. En aquesta tasca, no s'utilitza el robot. L'operari es troba prop de la taula i apunta cap a alguns forats de l'objecte. L'objectiu és aconseguir una posició d'aproximació als forats en relació a l'objecte atesa la geometria d'aquest. La posició i orientació de l'objecte són desconeguts i han de ser determinats a partir de les imatges proporcionades per la càmera 3D. A més, el gest d'assenyalar de l'operari ha de ser reconegut i el punt en el qual l'operari està assenyalant ha de ser estimat en relació amb l'objecte.

El projecte es basa en Robot Operating System (ROS) Industrial. El sistema està compost per un robot UR5 i dos ordinadors. Aquests elements executen nodes ROS. La comunicació, entre els diferents nodes del sistema, es realitza per mitjà de missatges de ROS.

Per localitzar la posició i orientació de la peça, s'utilitza el paquet PointCloud. Quan s'obté la posició i orientació final de la peça, la informació es publica en el tòpic /workpiece_pose.

L'ordinador amb Ubuntu utilitza el paquet Openni_tracker per obtenir l'esquelet de l'operari. Per reconèixer el gest d'assenyalar es necessita la informació continguda en els missatges: /right_elbow i /right_hand.

Utilitzant la informació proporcionada (/ right_hand, / right_elbow i / workpiece_pose) es pot determinar la posició de l'objecte en què l'humà està assenyalant.

Finalment, el robot pot posicionar-se en un punt d'aproximació d'on l'operari ha assenyalat. En futures fases del treball, utilitzant les càmeres estèreo situats a l'extrem del robot, el robot hauria de posicionar-se perfectament sobre els forats i inserir els reblons en els orificis indicats per l'operari.

Universitat de Vic
Escola Politècnica Superior

FINAL PROJECT SUMMARY

DEGREE IN MECHATRONICS ENGINEERNIG
DEGREE IN INDUSTRIAL ELECTRONICS ENGINEERING AND AUTOMATION

Title: *Worker-robot interaction: Control system using ROS-Industrial and 3D vision*
Keywords: *ROS, Robotics, Matlab, PointCloud, Kinect, Collaborative robotics*
Author: Míriam Laliga Cervilla
Director: Gerard Masferrer Caralt (UVIC-UCC)
Date: June 2015

The European robotics industry needs competitive solutions to attain and keep global leadership in robotics products and services. The application scenario itself is open, as to encourage creativity with regard to developing novel manufacturing applications involving human-robot collaboration.

In this work, a 6-DoF industrial robot is equipped at its end effector with a riveting tool and a stereo camera. An object is placed on the table in the working range of the robot. The object has holes to put rivets in. A 3D camera is mounted above the workplace and oversees everything on the table. A second 3D camera is mounted to capture the human approaching.

Project requires deriving where the holes for riveting based on a human pointing gesture. The task consists of three sub-tasks: 1) Recognizing pointing gesture, 2) Localizing objects based on CAD data and 3) Localizing position on object at which the human is pointing. In this task, the robot is not used. A human stands close to the table and point towards some rivet holes in the object. The aim is to get the approximate position of the holes in relation to the object. The geometry of the object is given. The position and orientation of the object are unknown and have to be determined from the images provided by the 3D camera. Furthermore the pointing gesture of the human has to be recognized and the point on the object at which the human is pointing has to be estimated in relation to the object.

The project is based on Industrial Robot Operating System (ROS). The system is composed by a UR5 robot and two PC's. These elements run ROS nodes. The communication, between the different nodes of the system, is released using ROS messages.

To locate the piece position and orientation, it has been used the PointCloud package. When is obtained the final position and orientation, the information is published in the topic /workpiece_pose.

Ubuntu PC run Openni_tracker package, this broadcasts the Kinect skeleton frames using tf. For pointing gesture recognition has been used /right_elbow and /right_hand. These TF's are send to ROS_Matlab Node trough a ros_message.

Using the information provided (/right_hand, /right_elbow and /workpiece_pose) can be determined the position on object at which the human is pointing.

Finally, the robot can position itself in an approach point where the worker pointed out. In the following work phases, using the stereo cameras located at the end of the robot, the robot has to position itself perfectly over the holes and it has to insert rivets in the indicated holes by the worker.

1. Introducció

1.1. Antecedents

La indústria de la robòtica Europea necessita solucions competitives per assolir i mantenir el lideratge mundial de productes i serveis de robòtica. La societat europea es beneficiarà d'aquesta posició de lideratge, fent ús d'aquests productes i serveis de robòtica que aspiren a millorar la seva qualitat de vida i les condicions de treball. L'adopció de la tecnologia robòtica ajudarà a les empreses manufactureres europees, en particular les PIME, per adaptar-se a les pressions de la competència mundial. Proveïdors de TIC, com ara fabricants de robots, integradors de sistemes i proveïdors de solucions tecnològiques són facilitadors importants per a la millora de la eficiència, l'adaptació i la sostenibilitat dels sistemes de fabricació, així com la seva integració en els processos de negociació en un context industrial cada cop més globalitzat.

Per impulsar la robòtica i la manufactura a Europa hi ha una necessitat d'una perspectiva continental sobre els temes que necessiten ser investigats i desenvolupats. Tant l'Agenda Estratègica (SRA) per la robòtica a Europa, publicada al juliol de 2009 amb l'ajuda de l' European Technology Platform (EUROP), i el full de ruta de les empreses del futur entreguen aquesta perspectiva europea i són al mateix temps perfectes exemples d'una exitosa col·laboració entre tots els integrants de la cadena de valor en la fabricació i el manteniment, és a dir: els usuaris finals, integradors de sistemes, proveïdors de tecnologia i experts en investigació. Durant el desenvolupament dels programes estratègics d'investigació de la robòtica i la manufactura, la fertilització a través d'aquests grups va portar a un full de ruta orientat a la realització de noves visions de productes i aplicacions en escenaris per a l'any 2020. Tot i així, la implementació d'aquests programes estratègics d'investigació i la realització dels escenaris d'aplicació i visió de nous productes associats requereixen un esforç conjunt entre els acadèmics i la indústria, incloses les PIME.

Com a facilitador de tot això apareix la iniciativa EuRoC (European Robotics Challenges), que proposa el llançament de diferents reptes rellevants per a la indústria. Un d'aquests reptes és en el que es basa aquest projecte, *Interacció operari-robot: Sistema de control mitjançant ROS-Industrial i visió 3D*.

1.2. Objectius

El principal objectiu del repte és desenvolupar les habilitats de percepció i cognició necessaris per a una col·laboració estreta i segura entre humans i robots utilitzant Robot Operating System (ROS) com a middleware subjacent.

El repte proposat per la iniciativa EuRoC i en el que es basa aquest projecte planteja una aplicació específica per poder dur a terme l'objectiu d'aquest treball.

El repte proposa desenvolupar un sistema de control mitjançant ROS i visió 3D per tal que, quan un operari assenyali un punt d'una peça situada dins l'espai de treball del robot, aquest s'hi aproximi per realitzar-hi una operació creant, d'aquesta manera, una interacció entre l'operari i el robot.

2. Eines utilitzades

2.1. Recursos utilitzats

Per a la realització del projecte s'han utilitzat els següents programes: Matlab 2013a per al reconeixement de la posició en què l'operari assenyala la peça i el sistema de comunicació ROS per a la comunicació amb la resta d'equips involucrats.

2.2. Robot Operating System

Robot Operating System (ROS) és un marc flexible per a l'escriptura de software per a robots. És una col·lecció d'eines i biblioteques que tenen com a objectiu simplificar la tasca de crear un comportament complex i robust d'un robot en una àmplia varietat de plataformes robòtiques.

ROS té tres nivells de conceptes: el nivell de sistemes d'arxius (Filesystem level), el nivell de computació gràfica (Computation graph level) i el nivell comunitari.

2.2.1. ROS Filesystem Level

Els conceptes del nivell de sistemes d'arxius es refereix principalment als recursos de ROS que es poden trobar al disc, com són:

Paquets (Packages): Són la unitat principal de l'organització del software a ROS. Un paquet pot contenir executables de ROS (nodes), llibreries, bases de dades, arxius de configuració, etc. Els paquets són l'element més atòmic de construcció d'elements de ROS.

Metapaquets (Metapackages): Són paquets especialitzats que només serveixen per representar un grup de paquets relacionats.

Package Manifests: Proporcionen dades sobre un paquet, normalment inclou el seu nom, la versió, les dependències, etc.

Missatges (msg): Descripció dels tipus de missatges, emmagatzemats a my_package/msg/MyMessageType.msg, defineixen les estructures de

dades dels missatges enviats amb ROS. Aquesta definició es fa creant un arxiu amb extensió ".msg".

Serveis (srv): Descripcions dels tipus serveis, emmagatzemats a my_package/srv/MyServiceType.srv, defineixen les estructures de dades de petició i resposta dels serveis amb ROS. Com en el cas dels missatges es fan creant arxius amb extensió ".srv".

2.2.2. ROS Computation Graph Level

La computació gràfica és la xarxa de processos de ROS que estan processant les dades d'igual a igual (Peer-to-Peer), això vol dir que hi pot haver comunicació entre tots els elements que s'estan executant en ROS. Els conceptes bàsics de Computació gràfica ROS són:

Nodes: Els nodes són equiparables als arxius executables en qualsevol sistema operatiu convencional. Un sistema de control del robot pot comprendre, generalment, un gran nombre de nodes. Per exemple, un node controla els servomotors del robot, un altre que realitza els càlculs de la trajectòria que seguirà, un altre proporciona una imatge en 3D de l'entorn en què es troba el robot i així successivament. Un node s'escriu utilitzant una llibreria client ROS com pot ser roscpp (escrit en C++), rospy (escrit en Pyton) o rosmatlab (escrit en Matlab).

Mestre (Master): El mestre ROS proporciona el nom de registre i les operacions de recerca per a la resta de la computació gràfica. Sense el mestre, els nodes no serien capaços de trobar els altres nodes, intercanviar missatges, ni invocar serveis.

Servidor de paràmetres (Parameter Server): Permet que les dades siguin emmagatzemades de forma centralitzada, de manera que qualsevol node pugui accedir-hi, tant per recollir informació com per emmagatzemar-la.

Missatges (Messages): els nodes es comuniquen entre si mitjançant l'enviament de missatges. Un missatge simplement és una estructura de

dades. Admeten tipus primitius estàndard (enter, coma flotant, booleans, etc.).

Per al desenvolupament d'aquest treball únicament s'ha fet servir el següent tipus de missatge:

geometry_msgs/Pose – Representació d'una posició a l'espai amb una posició i una orientació. El qual està format per:

Point position
  float64 x
  float64 y
  float64 z
Quaternion orientation
  float64 x
  float64 y
  float64 z
  float64 w

Tòpics: Els missatges s'enruten a través d'un sistema de transport amb publicació/subscripció semàntica. Un node envia un missatge publicant-lo a un determinat tòpic. El tòpic és el nom que s'utilitza per identificar el contingut del missatge. Un node que està interessat en un determinat tipus de dades es subscriurà al tòpic apropiat que les publica. Pot haver-hi múltiples publicadors i subscriptors concurrents per a un únic tòpic i un únic node pot publicar i/o subscriure's a múltiples tòpics. En general, els publicadors i subscriptors no són conscients de l'existència de la resta. La idea és pensar que un tòpic és com un bus on qualsevol es pot connectar per enviar o rebre les dades que hi passen sempre i quan siguin del tipus correcte.

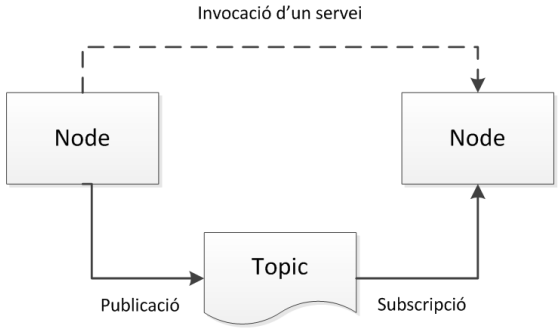

Figura 1 Esquema de comunicació entre dos nodes

Serveis (Services): El model de publicació/subscripció és un paradigma de comunicació molt flexible però no és apropiat per a les interaccions sol·licitud/resposta, cosa que requereix molt sovint un sistema distribuït. Aquestes interaccions es realitzen a través dels serveis, es defineixen per un parell d'estructures de missatges: un per la sol·licitud i un altre per la resposta. Un node proporciona un servei sota un nom i un client utilitza el servei a través de l'enviament del missatge de sol·licitud i espera la resposta. S'ha d'anar amb compte amb els serveis ja que fins que no es finalitza l'execució del node al qual es fa la trucada, l'execució queda bloquejada fins que el node retorna una resposta.

Accions (Actions)*:* Funcionen de manera semblant a un servei, però l'usuari pot cancel·lar l'acció en qualsevol moment, i no bloqueja l'execució principal. Les accions estan formades per un node servidor, que es troba a l'espera de que se li enviï la comanda per executar el seu codi, i un node client que és l'encarregat d'enviar la comanda d'inici al node servidor i en rep la resposta.

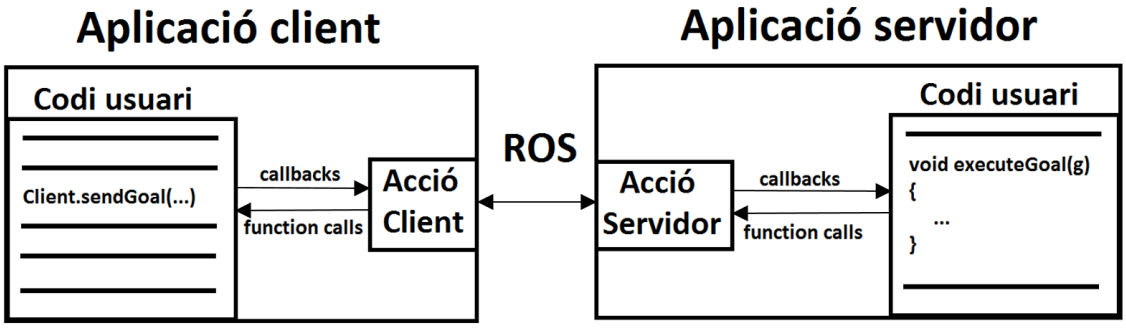

Figura 2 Diagrama de comunicació client-servidor

Bosses (Bags): Són un format per guardar i reproduir dades dels missatges ROS. Són un important mecanisme per a l'emmagatzemament de dades, com ara les dels sensors, que poden ser difícils de recollir però que són necessàries per al desenvolupament i prova d'algoritmes.

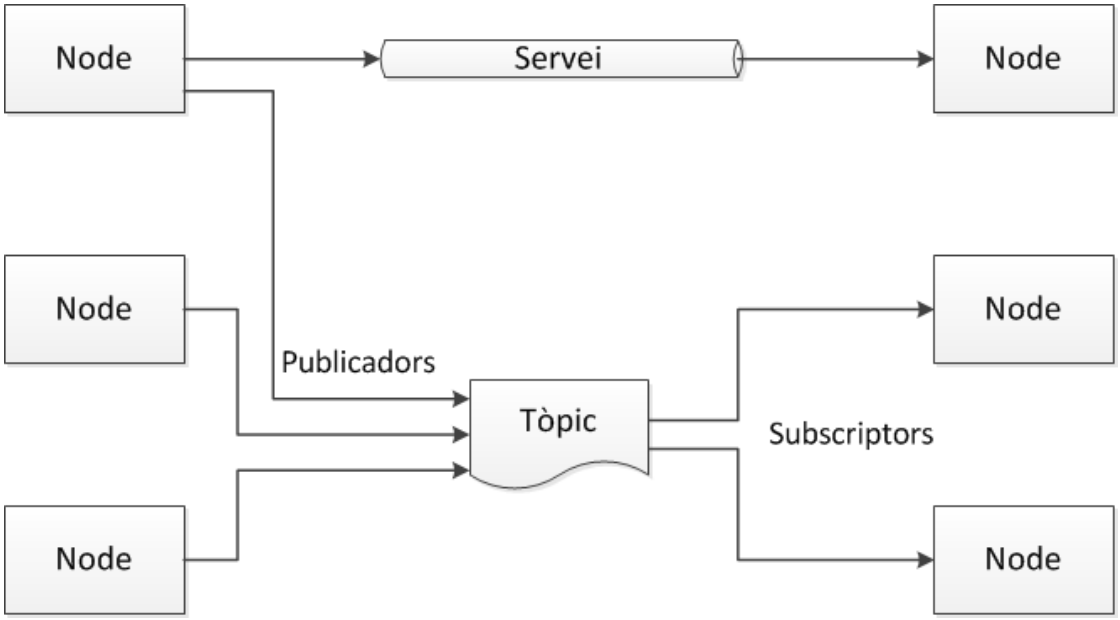

Figura 3 Esquema de comunicació de múltiples nodes

2.3. Matlab

MATLAB (abreviatura de Matrix Laboratory, "laboratori de matrius") és un software matemàtic especialment útil per a la manipulació de matrius, la representació de dades i la implementació d'algorismes en un entorn de desenvolupament integrat (IDE) amb un llenguatge de programació propi (llenguatge M).

Figura 4 Logotip Matlab

Mitjançant *toolbox* podem ampliar les capacitats del programa, disposem de múltiples eines per processament de senyal, classificació, estadística, robòtica, entre d'altres.

L'elecció d'aquest software ha estat bàsicament per l'experiència anterior, ja que ha estat un dels més utilitzats a nivell universitari. Per altra banda, també s'ha decidit utilitzar-lo perquè recentment s'ha publicat un paquet que dóna suport a Robot Operating System (ROS).

2.3.1. Robot Operating System suport de Matlab

Aquest paquet permet crear nodes ROS amb MATLAB i intercanviar missatges amb altres nodes de la xarxa ROS, incloent-hi robots i simuladors compatibles amb ROS.

Aquest paquet de suport és una extensió de la API rosjava. Inclou una nova API per la creació de nodes ROS dins de MATLAB basant-se en el mateix mecanisme de publicador/subscriptor ROS.

Les principals característiques que permet són:

- Crear nodes, publicadors i subscriptors ROS directament des de MATLAB
- Crear i enviar missatges ROS des de MATLAB
- Crear publicadors i habilitar-los per a que publiquin els seus tòpics
- Crear subscriptors que executen funcions MATLAB al rebre els tòpics
- Habilitar el llançament de mestres ROS en el host local de MATLAB.

Aquest paquet conté les següents classes per a la construcció de màsters, nodes, publicadors, subscriptors i missatges ROS:

- rosmatlab.roscore
- rosmatlab.node
- rosmatlab.publisher
- rosmatlab.subscriber
- rosmatlab.message

2.4. Robot UR5

El robot UR5 és un robot de 6 eixos d'Universal Robots que permet automatitzar tasques repetitives i perilloses amb càrregues útils de fins a 5 kg amb un radi d'acció de 850 mm. L'UR5 és perfecte per realitzar processos col·laboratius de poc pes com pick & place.

Figura 5 Robot UR5

2.5. Sensor Kinect

El sensor Kinect és un sensor composat per un sensor d'imatge i un sensor de profunditat (format per un emissor d'infraroig i un sensor d'imatges IR).

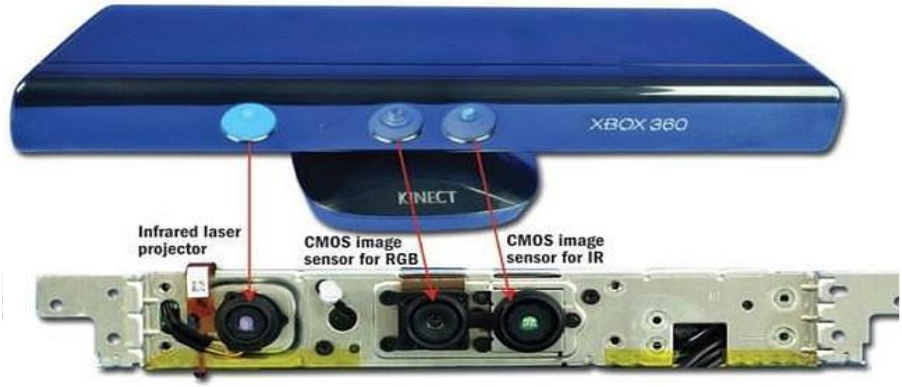

Figura 6 Sensor Kinect

3. Descripció de l'entorn

La configuració de l'entorn en què s'ha basat aquest treball ha estat la mateixa que es proposa al repte EuRoC.

Aquesta consisteix en:

- Un robot industrial de 6 eixos muntat sobre una taula (UR5).
- Una càmera 3D col·locada sobre l'espai de treball (Kinect).
- Una càmera 3D col·locada de tal manera que visualitza el robot i el seu entorn (Kinect).

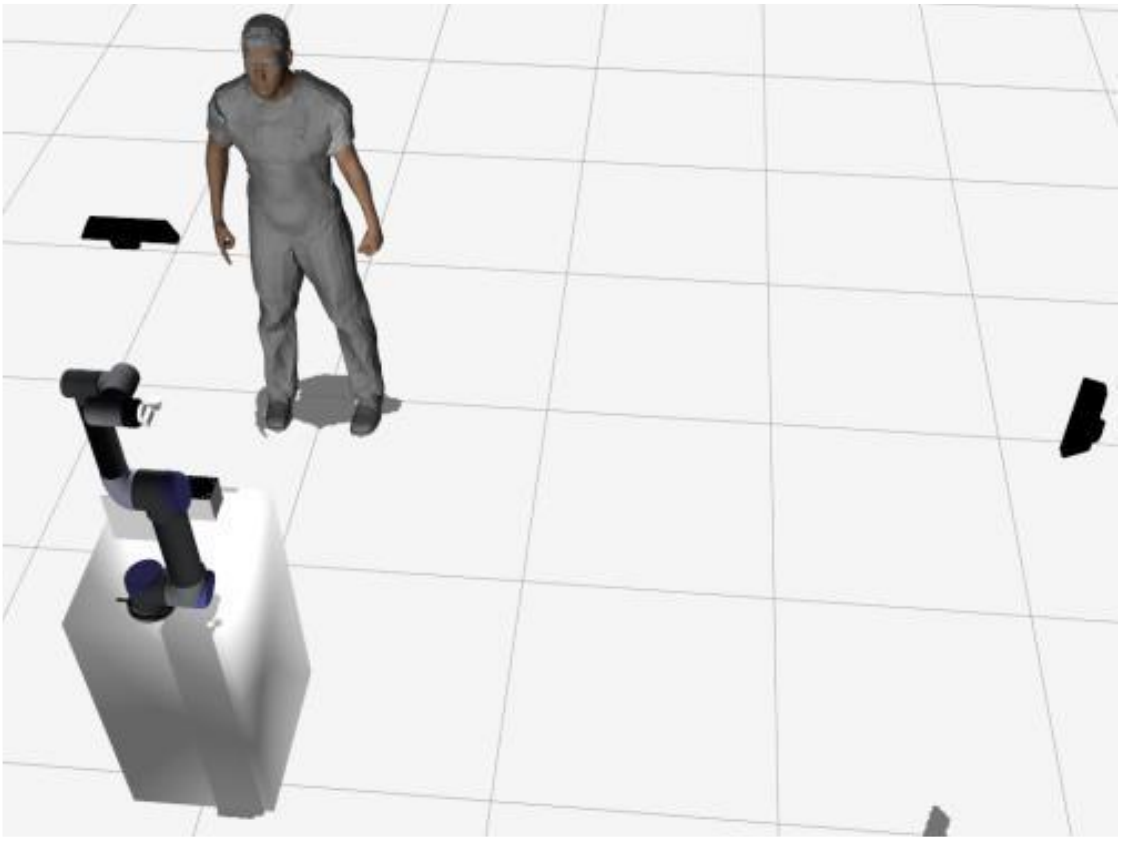

Figura 7 Captura de l'entorn virtual

Per a realitzar el projecte s'ha hagut de fer una rèplica aproximada de l'entorn de treball amb els diferents elements de la configuració, tal com es mostra a la següent figura:

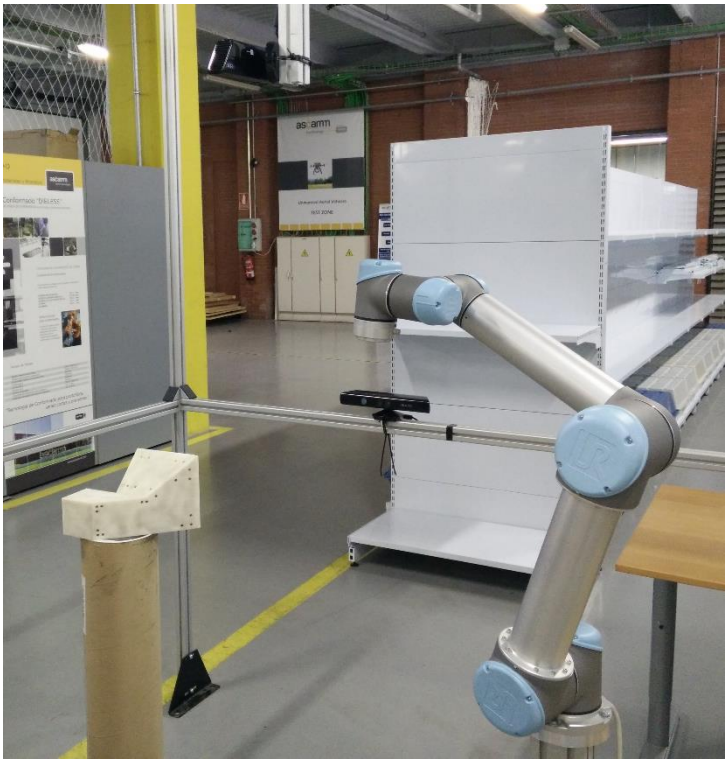

Figura 8 Fotografia de l'entorn real amb tots els elements

Per altra banda, també s'ha hagut de definir l'arquitectura del sistema. Per fer-ho, s'han hagut d'establir les diferents tasques a realitzar i repartir-les per tal de poder fer un treball conjunt, però que també permeti treballar de forma independent. Les diferents tasques que s'han definit són les següents:

- Detecció de la posició i orientació de la peça sobre el pla de treball.
- Detecció de la posició de les articulacions de l'operari (concretament el colze dret i la mà dreta).
- Determinar si l'operari està assenyalant la peça (amb les dades obtingudes en les tasques anteriors).
- Moure el robot a la posició d'aproximació on està assenyalant l'operari (amb les dades obtingudes en la tasca anterior).
- Afinar la posició i inserir l'eina als forats de la posició assenyalada.

Tenint en compte aquestes tasques també s'han definit els tòpics amb els quals s'han de comunicar:

- **/workpiece_pose**: Conté la informació de la posició i orientació de la peça. (Tipus geometry_msgs/Pose)
- **/right_hand**: Conté la informació de la posició i orientació de la mà dreta de l'operari. (Tipus geometry_msgs/Pose)
- **/right_elbow**: Conté la informació de la posició i orientació del colze dret de l'operari. (Tipus geometry_msgs/Pose)
- **/intersection**: Conté la informació de la posició i orientació de la posició d'aproximació a la què ha d'anar el robot. (Tipus geometry_msgs/Pose)

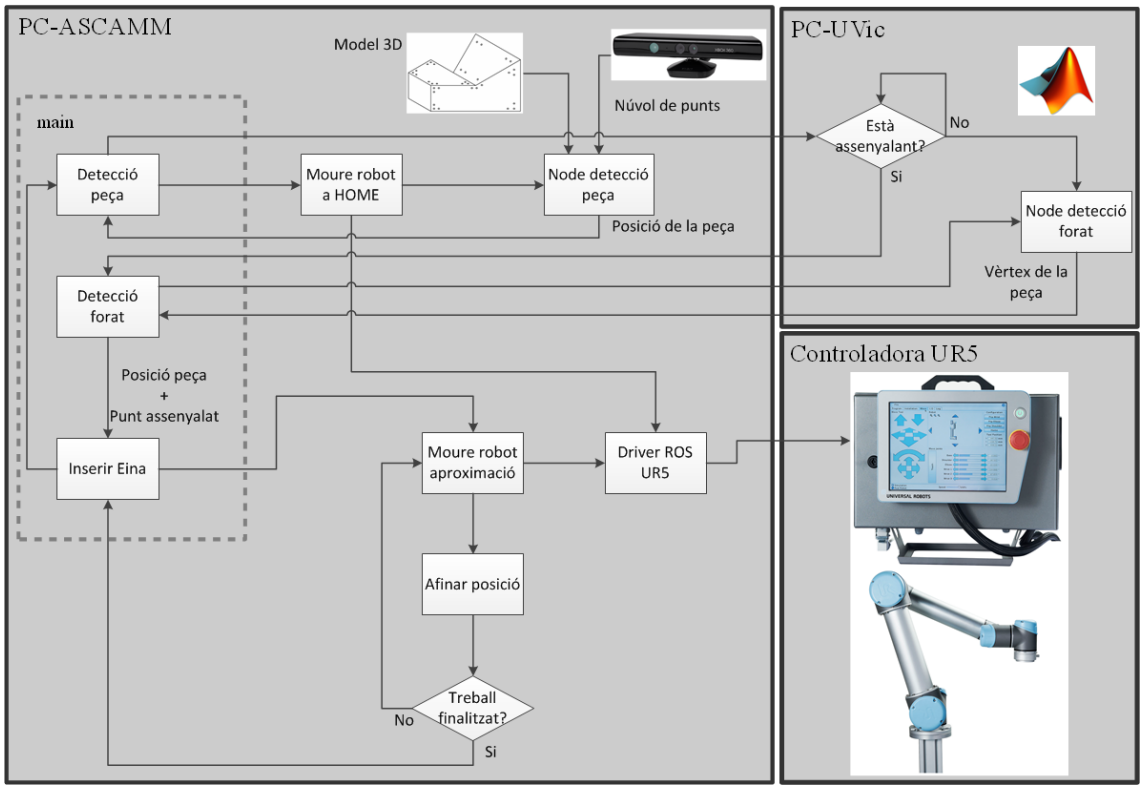

Figura 9 Arquitectura del sistema

4. Fonaments teòrics

En aquest apartat s'introduiran els diferents elements teòrics emprats per al desenvolupament del projecte.

4.1. Geometria

La geometria analítica estudia les figures geomètriques mitjançant tècniques bàsiques de l'anàlisi matemàtic i de l'àlgebra en un determinat sistema de coordenades. En el cas que es presenta, emprant la geometria cartesiana s'han obtingut les equacions següents.

4.1.1. Recta

Sent P un punt de la recta r i $\vec{u}$ el seu vector director, el vector $\overrightarrow{PX}$ té igual direcció que $\vec{u}$, per tant, és igual que $\vec{u}$ multiplicat per un escalar:

$$\overrightarrow{PX} = \lambda \cdot \vec{u} \qquad (1)$$

$$(x - x_0, y - y_0, z - z_0) = \lambda \cdot (u_1, u_2, u_3) \qquad (2)$$

$$(x, y, z) = (x_0, y_0, z_0) + \lambda \cdot (u_1, u_2, u_3) \qquad (3)$$

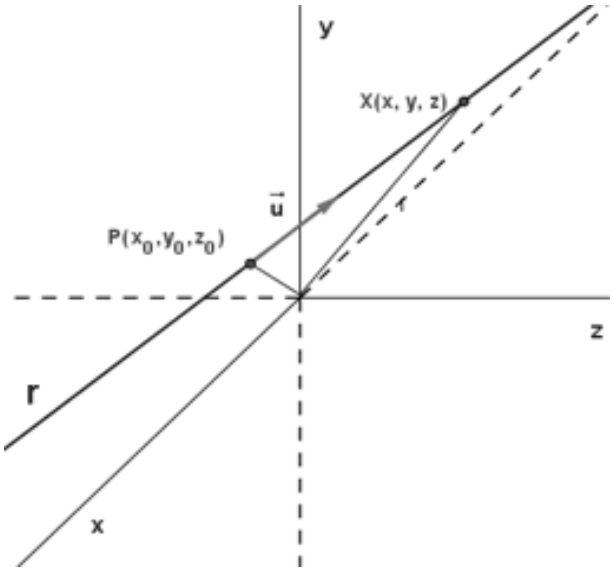

Figura 10 Representació d'una recta a l'espai definida per un punt i el seu vector director

Operant amb l'equació vectorial s'obté:

$$(x, y, z) = (x_0 + \lambda \cdot u_1, y_0 + \lambda \cdot u_2, z_0 + \lambda \cdot u_3) \qquad (4)$$

I aquesta igualtat es verifica sempre que:

$$\begin{cases} x = x_0 + \lambda \cdot u_1 \\ y = y_0 + \lambda \cdot u_2 \\ z = z_0 + \lambda \cdot u_3 \end{cases} \quad (5)$$

Aïllant i igualant λ s'obtenen les equacions contínues de la recta:

$$\frac{x - x_0}{u_1} = \frac{y - y_0}{u_2} = \frac{z - z_0}{u_3} \quad (6)$$

Si a les equacions continues de la recta es treuen els denominadors i es passa tot al primer membre, s'obtenen les equacions implícites de la recta determinades per la intersecció de dos plans:

$$\begin{cases} Ax + By + Cz + D = 0 \\ A'x + B'y + C'z + D' = 0 \end{cases} \quad (7)$$

4.1.2. Pla

Un pla queda determinat per un punt P i dos vectors amb diferent direcció. Per a que un punt P pertanyi al pla π el vector $\overrightarrow{PX}$ ha de ser coplanari amb $\vec{u}$ i $\vec{v}$.

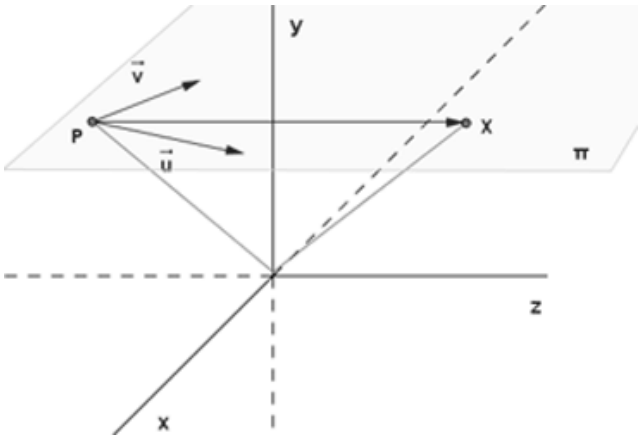

Figura 11 Representació d'un pla a l'espai definit per un punt i dos vectors directors

$$\overrightarrow{PX} = \lambda \cdot \vec{u} + \mu \cdot \vec{v} \quad (8)$$

$$(x - x_0, y - y_0, z - z_0) = \lambda(u_1, u_2, u_3) + \mu(v_1, v_2, v_3) \quad (9)$$

$$(x, y, z) = (x_0, y_0, z_0) + \lambda(u_1, u_2, u_3) + \mu(v_1, v_2, v_3) \quad (10)$$

Operant aquesta equació del pla s'arriba a la igualtat:

$$(x, y, z) = (x_0 + \lambda \cdot u_1 + \mu \cdot v_1, y_0 + \lambda \cdot u_2 + \mu \cdot v_2, z_0 + \lambda \cdot u_3 + \mu \cdot v_3) \quad (11)$$

I aquesta igualtat es verifica sempre que:

$$\begin{cases} x = x_0 + \lambda \cdot u_1 + \mu \cdot v_1 \\ y = y_0 + \lambda \cdot u_2 + \mu \cdot v_2 \\ z = z_0 + \lambda \cdot u_3 + \mu \cdot v_3 \end{cases} \quad (12)$$

Aquest sistema ha de ser compatible determinat a les incògnites λ i μ. Per tant, el determinant de la matriu ampliada del sistema amb la columna dels termes independents ha de ser igual a zero.

$$\begin{vmatrix} x - x_0 & u_1 & v_1 \\ y - y_0 & u_2 & v_2 \\ z - z_0 & u_3 & v_3 \end{vmatrix} = 0 \quad (13)$$

Desenvolupant el determinant:

$$\begin{vmatrix} u_2 & v_2 \\ u_3 & v_3 \end{vmatrix} (x - x_0) - \begin{vmatrix} u_1 & v_1 \\ u_3 & v_3 \end{vmatrix} (y - y_0) + \begin{vmatrix} u_1 & v_1 \\ u_2 & v_2 \end{vmatrix} (z - z_0) = 0 \quad (14)$$

Es donen els valors:

$$A = \begin{vmatrix} u_2 & v_2 \\ u_3 & v_3 \end{vmatrix} \quad (15) \qquad B = -\begin{vmatrix} u_1 & v_1 \\ u_3 & v_3 \end{vmatrix} \quad (16) \qquad C = \begin{vmatrix} u_1 & v_1 \\ u_2 & v_2 \end{vmatrix} \quad (17)$$

Es substitueix:

$$A(x - x_0) + B(y - y_0) + C(z - z_0) = 0 \quad (18)$$

Realitzant les operacions s'obté el valor de D:

$$D = -A \cdot x_0 - B \cdot y_0 - C \cdot z_0 \quad (19)$$

Amb el que s'obté l'equació general del pla.

$$Ax + By + Cz + D = 0 \quad (20)$$

4.1.3. Orientació de tres punts

Per determinar el sentit de gir d'un triangle definit per tres punts p_1, p_2, i p_3 s'ha de realitzar el producte vectorial de (p_1,p_2) i (p_1,p_3). El signe d'aquest producte donarà el sentit de gir del triangle.

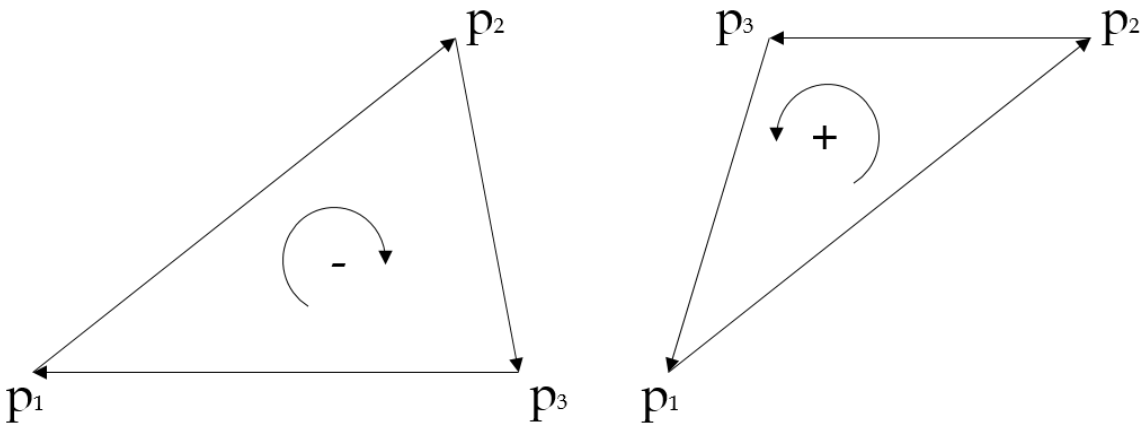

Figura 12 Sentit de gir d'un triangle

4.2. Coordenades i matrius homogènies

La representació mitjançant coordenades homogènies de la localització de sòlids en un espai n-dimensional es realitza a través de coordenades d'un espai (n+1)-dimensional. És a dir, un espai n-dimensional es troba representat en coordenades homogènies per (n+1) dimensions, de tal manera que un vector p (x, y, z) vindrà representat per p (w_x, w_y, w_z, w), on w té un valor arbitrari i representa un factor d'escala.

A partir de la definició de les coordenades homogènies es pot entendre el concepte de matriu de transformació homogènia. Es defineix com a matriu de transformació homogènia X una matriu de dimensió 4x4 que representa la transformació d'un vector de coordenades homogènies d'un sistema de coordenades a un altre.

4.2.1. Matriu homogènia 4x4

Una matriu homogènia està formada per 4 sub-matrius, una matriu de 3x3 corresponent a la rotació, una matriu 3x1 corresponent a la translació, una 1x3 per la perspectiva i una matriu 1x1 que representa un factor d'escala.

$$X = \begin{bmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotació & Translació \\ Perspectiva & Escalat \end{bmatrix} \quad (21)$$

En robòtica normalment només interessarà la matriu de translació i la matriu de rotació, agafant la matriu de perspectiva com a nul·la i la d'escalat prenent valor 1.

$$X = \begin{bmatrix} R_{3x3} & p_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotació & Translació \\ 0 & 1 \end{bmatrix} \quad (22)$$

Aquestes matrius permetran representar la posició i orientació d'un sistema OUVW rotat i traslladat respecte el sistema de referència OXYZ.

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = X \begin{bmatrix} r_u \\ r_v \\ r_w \\ 1 \end{bmatrix} \quad (23)$$

O vist des d'un altre punt de vista, traslladar i rotar un vector respecte OXYZ.

$$\begin{bmatrix} r'_x \\ r'_y \\ r'_z \\ 1 \end{bmatrix} = X \begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} \quad (24)$$

4.2.2. Translació

Per a un sistema OUVW traslladat únicament un vector

$$p = p_x\hat{i} + p_y\hat{j} + p_z\hat{k} \quad (25)$$

pel que fa al sistema fix OXYZ. La matriu homogènia serà la matriu bàsica de translació:

$$X(p) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (26)$$

Un vector qualsevol r, representat en OUVW per r_{uvw}, tindrà com a coordenades en el sistema OXYZ:

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_u \\ r_v \\ r_w \\ 1 \end{bmatrix} = \begin{bmatrix} r_u + p_x \\ r_v + p_y \\ r_w + p_z \\ 1 \end{bmatrix} \quad (27)$$

En el següent exemple es pot veure que el sistema O'UVW està traslladant el vector p(6,-3,8) respecte el sistema OXYZ. Del càlcul s'obté (r_x, r_y, r_z), les coordenades dels qual son r_{uvw}(-2, 7, 3) en el sistema O'UVW.

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -2 \\ 7 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 11 \\ 1 \end{bmatrix}$$

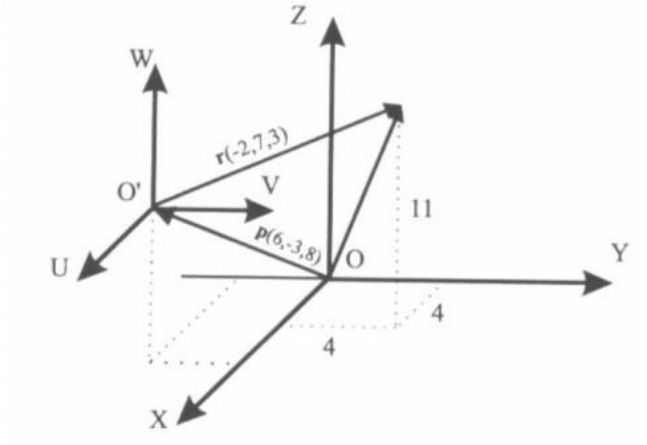

Figura 13 Exemple matriu homogènia de translació. (Barrientos)

4.2.3. Rotació

Es poden definir tres matrius bàsiques de rotació segons l'eix sobre el qual es produeix.

En un sistema O'UVW que només es troba rotat respecte OXYZ es tindrien les següents matrius segons l'eix de gir

$$Rot(x,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (28)$$

$$Rot(y,\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (29)$$

$$Rot(z,\theta)=\begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad (30)$$

Un exemple seria el següent, en què es pot observar el càlcul de les coordenades del vector r_{xyz} si $r_{uvw}(4, 8, 12)$. En un sistema OUVW girat -90º sobre l'eix OZ.

$$\begin{bmatrix}r_x\\ r_y\\ r_z\\ 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ -1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}4\\ 8\\ 12\\ 1\end{bmatrix}=\begin{bmatrix}8\\ -4\\ 12\\ 1\end{bmatrix}$$

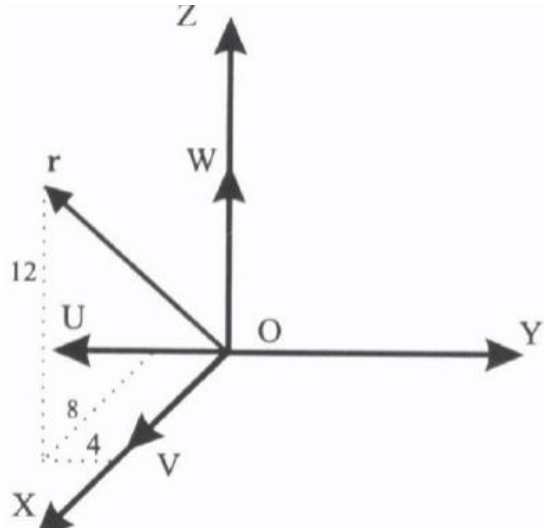

Figura 14 Exemple matriu homogènia de rotació (Barrientos)

4.2.4. Rotació i translació

Una vegada vistes les matrius de translació i rotació, és possible la combinació de les dues operacions multiplicant les matrius bàsiques. Cal tenir en compte que aquesta operació no és commutativa, per tant, no és el mateix una rotació i després una translació que una translació seguida d'una rotació.

Rotació seguida de translació:

$$Tran(p)\cdot Rot(x,\alpha)=\begin{bmatrix}1 & 0 & 0 & p_x\\ 0 & \cos\alpha & -\sin\alpha & p_y\\ 0 & \sin\alpha & \cos\alpha & p_z\\ 0 & 0 & 0 & 1\end{bmatrix} \quad (31)$$

Translació seguida de rotació:

$$Rot(x,\alpha)\cdot Tran(p)=\begin{bmatrix}1 & 0 & 0 & p_x\\ 0 & \cos\alpha & -\sin\alpha & p_y\cos\alpha - p_z\sin\alpha\\ 0 & \sin\alpha & \cos\alpha & p_y\sin\alpha + p_z\cos\alpha\\ 0 & 0 & 0 & 1\end{bmatrix} \quad (32)$$

4.2.5. Quaternions

Un quaternió Q està constituït per quatre components (q_0, q_1, q_2, q_3) que representen les coordenades del quaternió en una base {e, i, j, k}. Freqüentment es denomina part escalar del quaternió a la component q_0, i part vectorial a la resta dels components. De manera que un quaternió es pot representar com:

$$Q=[q_0,q_1,q_2,q_3]=[s,v] \quad (33)$$

On s representa la part escalar, i v la part vectorial.

Per a la utilització dels quaternions com a metodologia de representació d'orientacions s'associa el gir d'un angle X sobre el vector k al quaternió definit per:

$$Q=Rot(k,\theta)=\left(\cos\frac{\theta}{2},k\sin\frac{\theta}{2}\right) \quad (34)$$

D'aquesta associació aparentment arbitraria i gràcies a les propietats dels quaternions s'obté una important eina analítica per al tractament de girs i canvis d'orientació.

Quaternions: Matriu de transformació homogènia

El pas de quaternions a la matriu de transformació homogènia, i viceversa, es pot deduir fàcilment utilitzant com a representació auxiliar intermèdia l'eix i l'angle de rotació. A continuació s'expressa la relació directa en forma de matriu de transformació T en funció de les components d'un quaternió Q:

$$T = 2 \cdot \begin{bmatrix} q_0^2 + q_1^2 - \frac{1}{2} & q_1 \cdot q_2 - q_3 \cdot q_0 & q_1 \cdot q_3 + q_2 \cdot q_0 & 0 \\ q_1 \cdot q_2 + q_3 \cdot q_0 & q_0^2 + q_2^2 - \frac{1}{2} & q_2 \cdot q_3 - q_1 \cdot q_0 & 0 \\ q_1 \cdot q_3 - q_2 \cdot q_0 & q_2 \cdot q_3 + q_1 \cdot q_0 & q_0^2 + q_3^2 - \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \quad (35)$$

5. Algorismes desenvolupats

En aquest apartat s'entrarà en detall en els algorismes desenvolupats seguint l'arquitectura establerta a l'apartat 3 (vegeu Figura 9).

Tal com s'ha definit l'arquitectura, s'han hagut de desenvolupar diferents algorismes que es poden dividir, de forma general, en els nodes creats.

5.1. Node Detecció peça[1]

Per tal d'identificar la peça i la orientació en què s'ha col·locat, s'utilitza la Kinect que està enfocant la taula de treball.

El primer que es fa amb les dades obtingudes per la Kinect (el núvol de punts) és aplicar un Passthrougth filtre per tal d'acotar la distància de captura de la càmera.

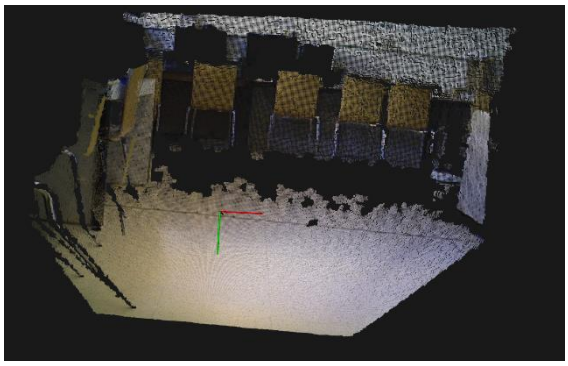

Figura 15 Abans d'aplicar el filtre

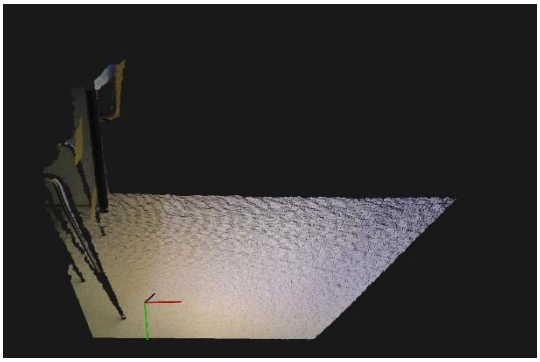

Figura 16 Després d'aplicar el filtre

Després s'aplica un filtre anomenat Voxel grid que redueix el nombre de punts per tal de fer el processat més ràpid.

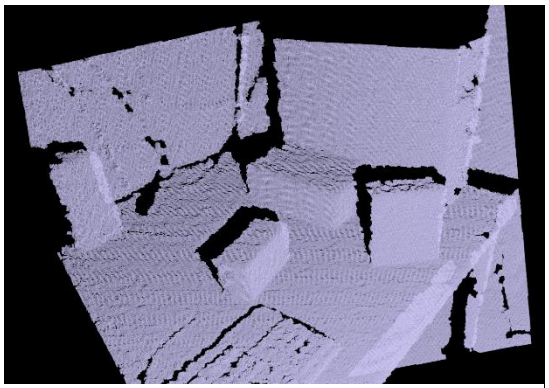

Figura 18 Abans d'aplicar el filtre Voxel gird

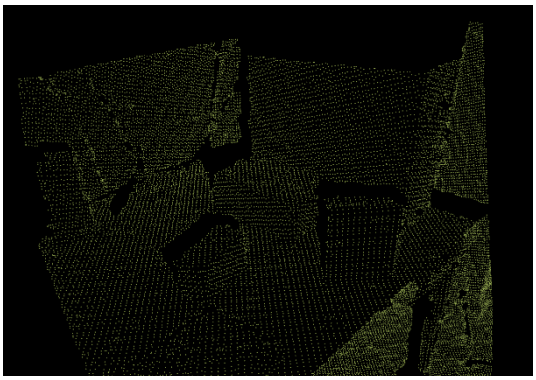

Figura 17 Després d'aplicar el filtre Voxel gird

[1] Aquest node ha estat desenvolupat per la fundació Ascamm. S'expliquen a grans trets els procediments utilitzats.

Seguidament s'aplica una funció que extreu les normals del núvol de punts.

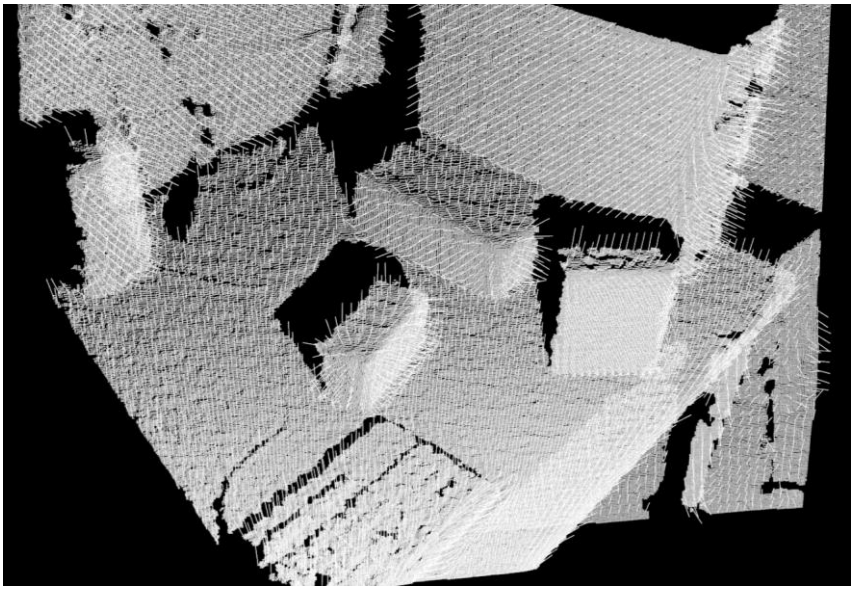

Figura 19 Representació de les normals obtingudes del núvol de punts

Després s'empra una altra funció que extreu les característiques del núvol de punts considerant les connexions de cadascun dels punts i els seus veïns i, a continuació, es fa una primera aproximació de la posició i orientació de la peça a l'espai. Seguidament es fa l'afinament de la posició i orientació de la peça i per últim es fa la conversió de les coordenades obtingudes amb referència la Kinect per referenciar-les respecte el món i es publica la posició i orientació de la peça al tòpic /workpiece_pose.

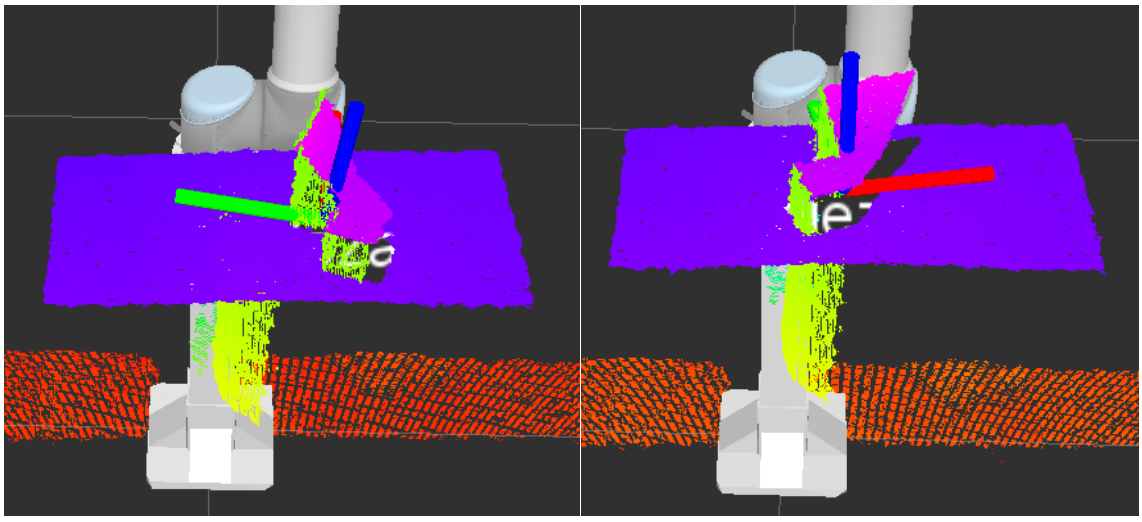

Figura 20 Exemples de posicionament de la peça a l'espai

5.2. Node Reconeixement operari[2]

Aquest node és l'encarregat d'obtenir la posició del colze dret i la mà dreta de l'operari. Per fer-ho s'ha utilitzat un node anomenat openni_tracker[3].

Aquest node és capaç de reconèixer el nombre d'usuaris que hi ha davant seu i, quan detecta que un usuari s'ha col·locat en posició Psi, comença el seguiment d'aquest publicant la posició de cadascuna de les seves articulacions respecte la Kinect.

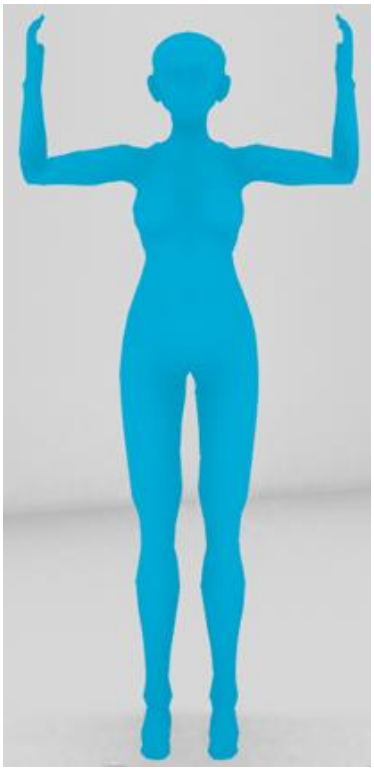

Figura 21 Posició Psi

La llista de les articulacions que proporciona és la següent:

- /head
- /neck
- /torso
- /left_shoulder
- /left_elbow

[2] Aquest node ha estat desenvolupat per la fundació Ascamm. S'expliquen a grans trets els procediments utilitzats.
[3] Node disponible a l'espai web de ROS: http://wiki.ros.org/openni_tracker

- /left_hand
- /right_shoulder
- /right_elbow
- /right_hand
- /left_hip
- /left_knee
- /left_foot
- /right_hip
- /right_knee
- /right_foot

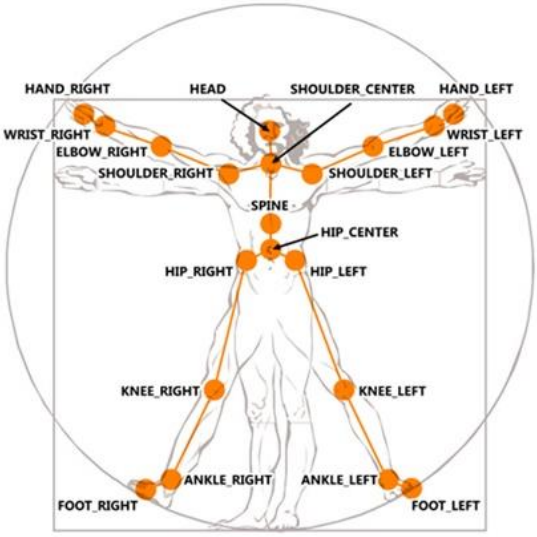

Figura 22 Articulacions detectades per la Kinect

Pel projecte únicament són necessàries la posició del colze dret i la mà dreta. Però per tenir tota la informació referenciada respecte el mateix origen s'ha de fer la transformació de les posicions del colze i la mà, així queden referenciades respecte el món i, posteriorment, es publiquen en els tòpics /elbow_pose i /hand_pose respectivament.

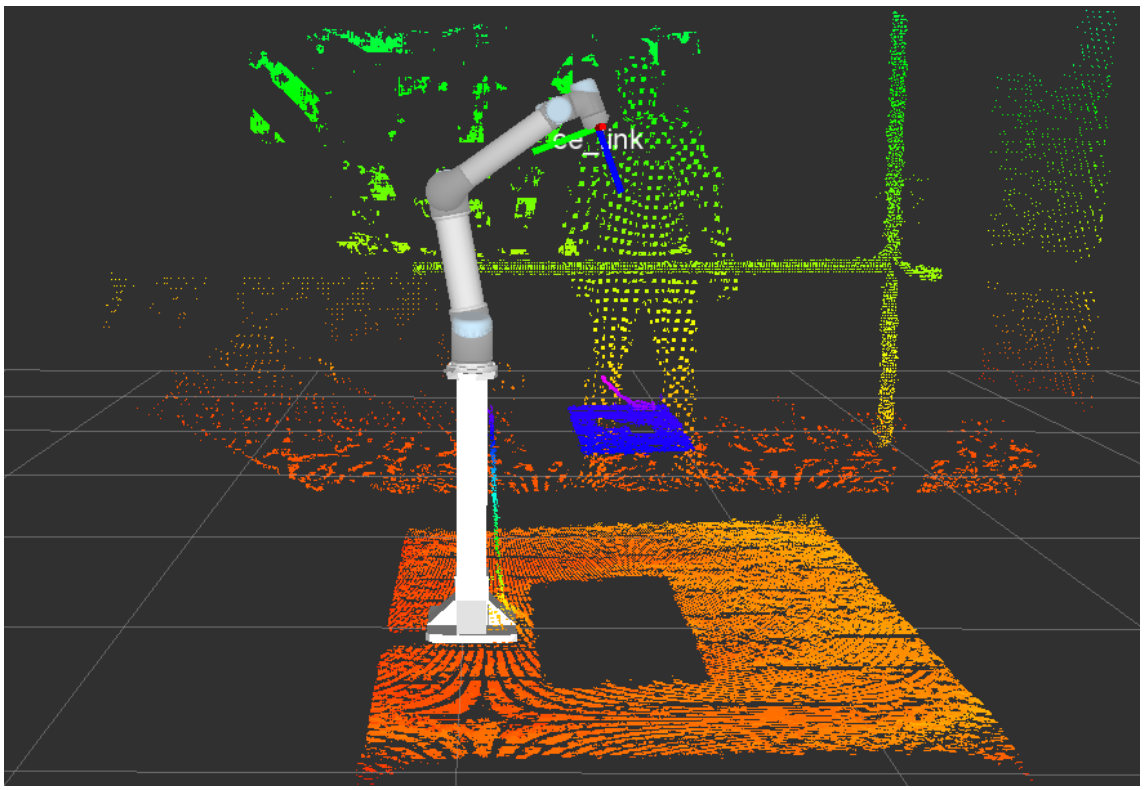

Figura 23 Usuari detectat però sense seguiment de les articulacions

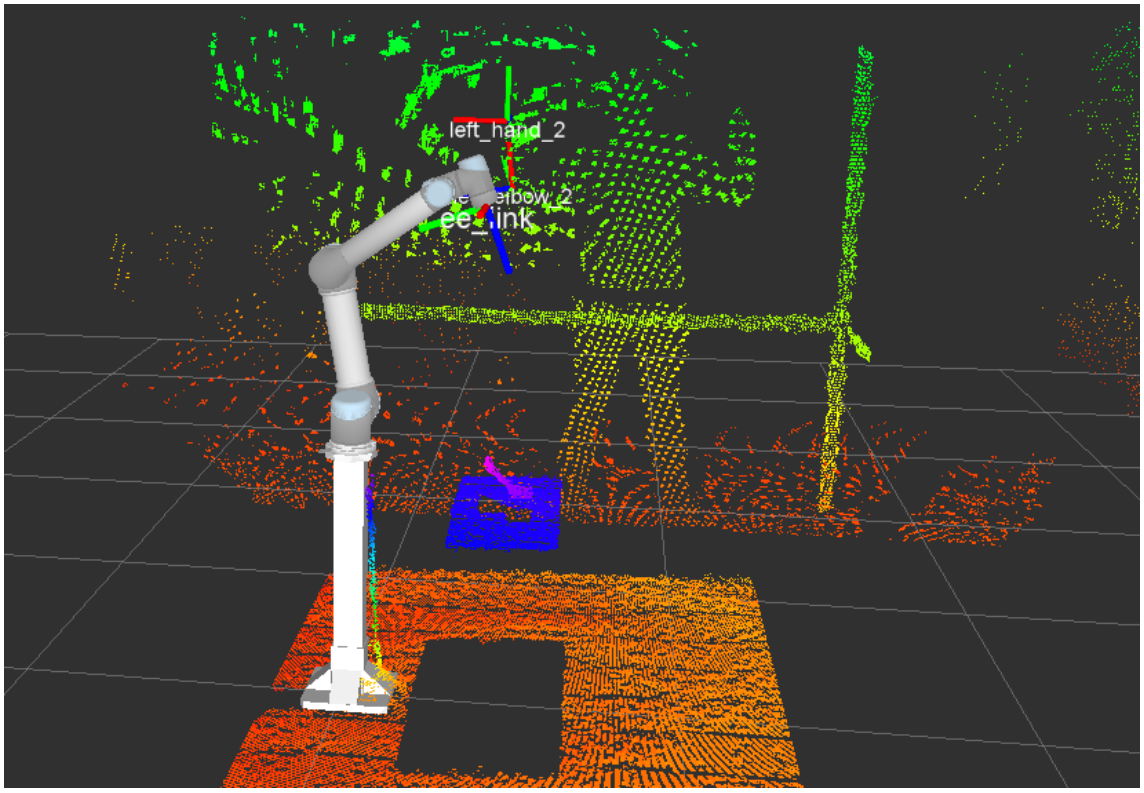

Figura 24 Usuari en posició Psi

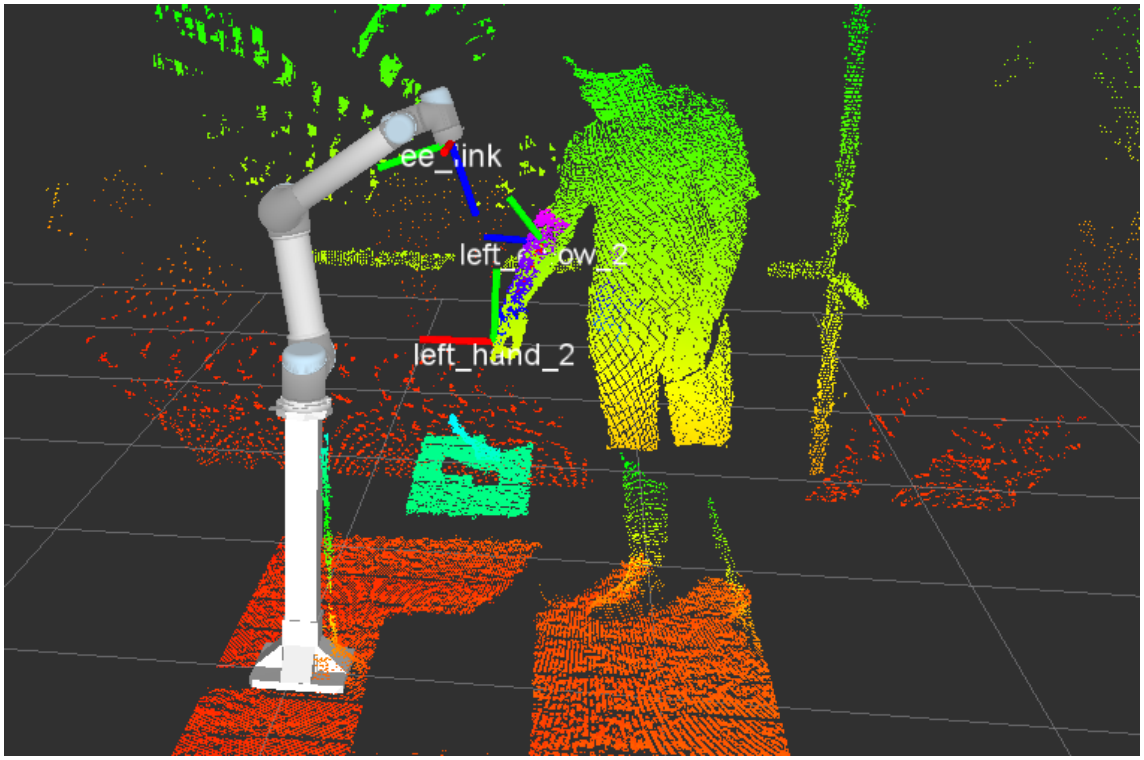

Figura 25 Usuari assenyalant la peça

5.3. Node Matlab

Aquest node és l'encarregat de rebre la informació publicada pels tòpics /workpiece_pose, /hand_pose i /elbow_pose, determinar si l'operari està assenyalant la peça i, si es dóna el cas afirmatiu, publicar en el tòpic /intersection la posició on s'ha assenyalat i l'orientació amb la que s'hi ha d'aproximar el robot.

Per això s'ha fet un algorisme principal que es subscriu als tòpics /workpiece_pose, /elbow_pose i /hand_pose i executa les funcions ReadPose.m, ReadElbow.m i ReadHandProcess.m respectivament quan rep noves dades del tòpic en qüestió.

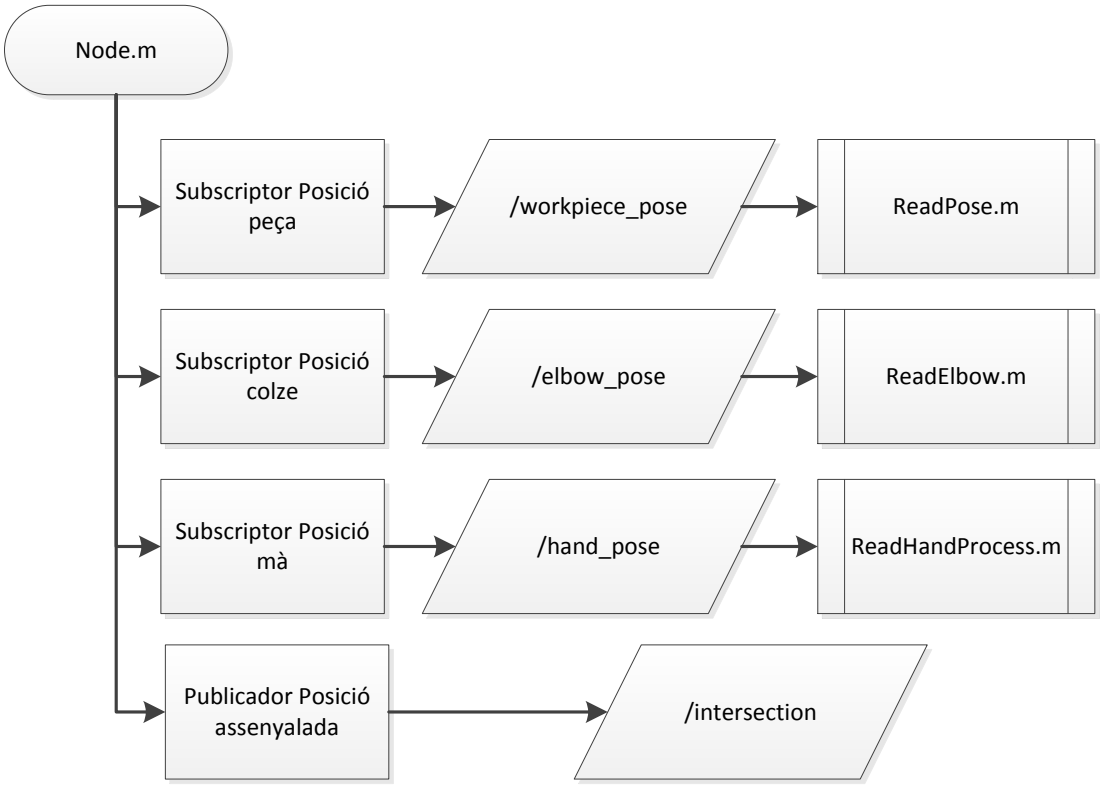

Figura 26 Diagrama de flux Node.m

5.3.1. Funció ReadPose Matlab

Aquesta funció s'executa quan rep noves dades del tòpic /workpiece_pose el qual, com ja s'ha comentat anteriorment, proporciona la posició i l'orientació en què està situada la peça.

Tenint en compte que la geometria de la peça és coneguda, es realitzen les rotacions i translacions pertinents per tal de poder situar cadascun dels punts de la peça en l'espai. Tot i així, abans s'ha de realitzar la transformació del quaternió rebut pel tòpic a una matriu de rotació homogènia per tal de facilitar els càlculs posteriors. Aquesta transformació es realitza amb una funció que aplica l'equació (35) i retorna una matriu de transformació homogènia.

Un cop coneguda la situació de tots els punts de la peça en l'espai es fa el càlcul de l'equació del pla de cadascuna de les cares de la peça estudiada. L'equació s'obté aplicant les equacions (16), (17) i (19) per a tres punts donats del pla obtenint un resultat com:

$$Ax + By + Cz + D = 0 \qquad (36)$$

Un cop obtinguts cadascun dels punts la funció fa una representació de la peça a l'espai:

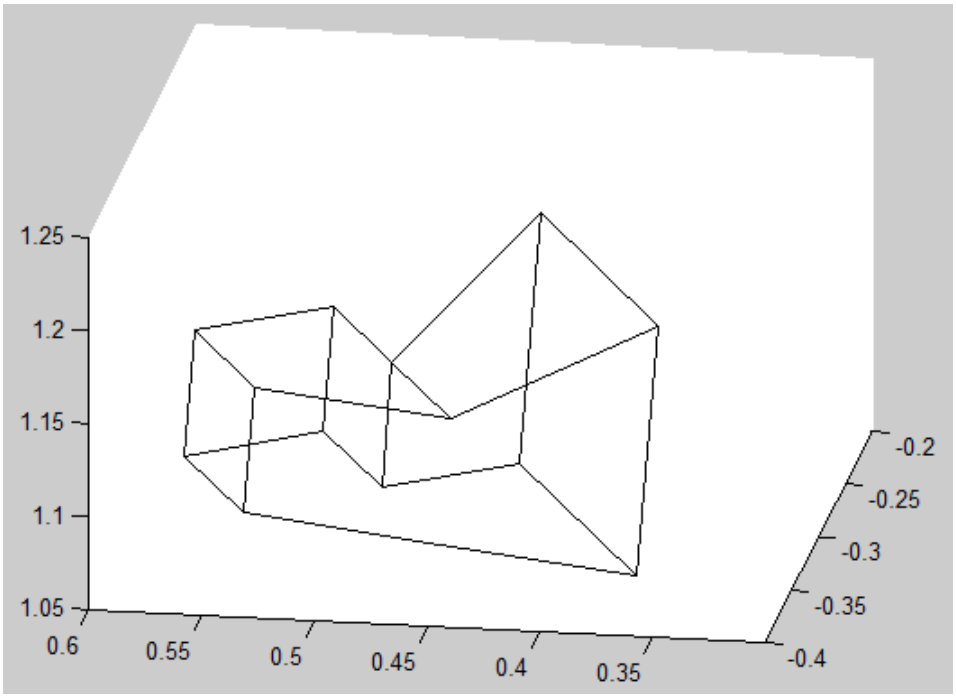

Figura 27 Representació de la peça en l'espai

Per altra banda, també s'ha fet la divisió per regions de cadascun dels plans per tal de, posteriorment, poder comprovar quina de les regions està assenyalant l'operari. La divisió ha estat la següent:

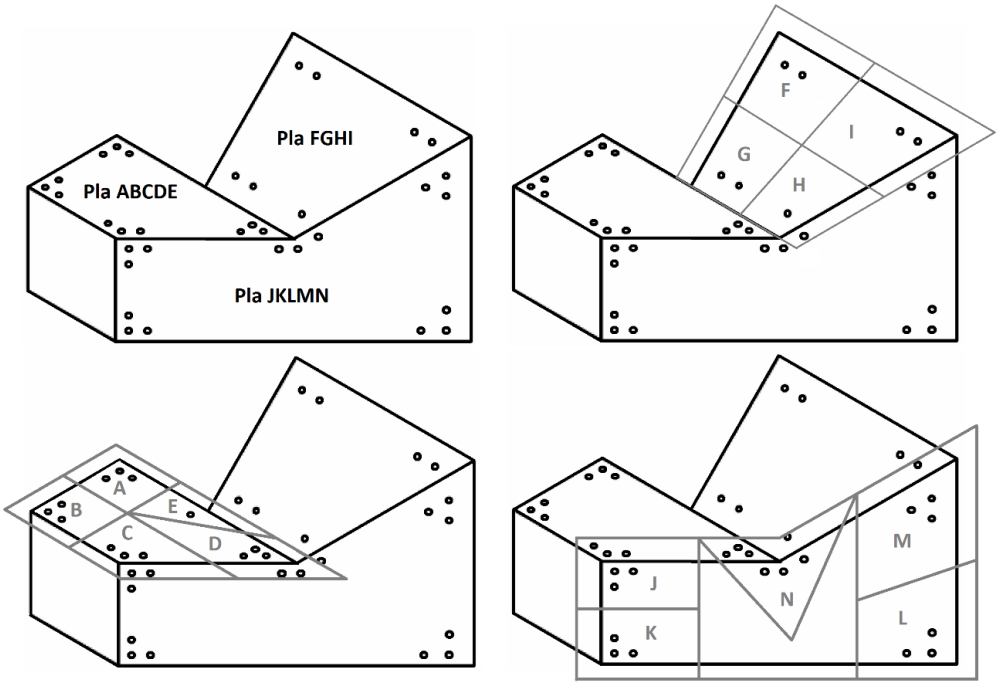

Figura 28 Regions dels plans de la peça

Altrament, també s'ha aprofitat aquesta funció per obtenir els punts als quals s'haurà d'aproximar el robot quan es detecti que una de les regions ha estat assenyalada i l'orientació amb què ho haurà de fer.

Tenint en compte que l'eix que defineix la direcció de l'eina del robot és l'eix Z s'han fet els càlculs previs de les rotacions i translacions:

Per al pla ABCDE la posició i orientació d'aproximació s'ha definit com es mostra a la Figura 29. A una distància del pla de 10 centímetres i amb l'eix Z perpendicular al pla.

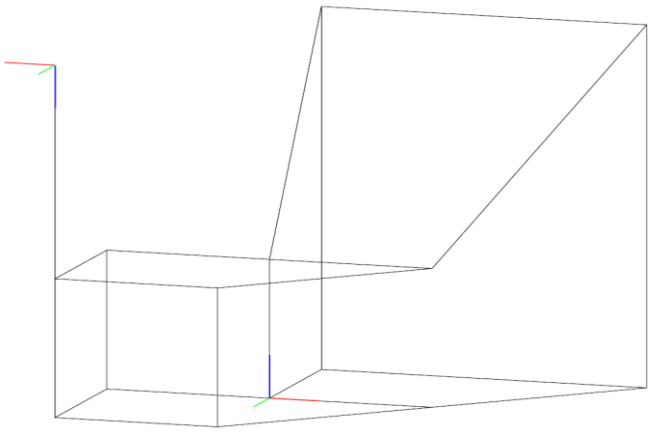

Figura 29 Posició i orientació d'aproximació per a un dels punts del pla ABCDE

Pel que fa a la posició, s'ha de fer una translació de 10 centímetres en direcció a la normal del pla aresta en qüestió. Per altra banda, pel que fa a l'orientació, s'ha de realitzar una rotació sobre l'eix Y de 180º respecte la base de la peça.

Per fer la translació s'ha obtingut el vector unitari normal al pla a partir de l'equació del pla (20) i s'ha dividit per 10 per tal de que la seva longitud sigui de 10 centímetres.

$$Trans_{Norm} = \begin{bmatrix} \frac{-A}{10 \cdot \sqrt{A^2 + B^2 + C^2}} \\ \frac{-B}{10 \cdot \sqrt{A^2 + B^2 + C^2}} \\ \frac{-C}{10 \cdot \sqrt{A^2 + B^2 + C^2}} \end{bmatrix} \quad (37)$$

$$Rot(y,180) = \begin{bmatrix} \cos(180) & 0 & \sin(180) \\ 0 & 1 & 0 \\ -\sin(180) & 0 & \cos(180) \end{bmatrix} \quad (38)$$

S'ha de tenir en compte, però, que a aquests càlculs se'ls ha d'afegir la rotació i la translació pertinents a la posició de la peça real obtinguts pel tòpic /workpiece_pose.

Per tant, la rotació final per a la posició d'aproximació dels punts del pla ABCDE serà:

$$Rot_{ABCDE} = Rot(y,180) \cdot Rot_{peça} \quad (39)$$

Sent $Rot_{peça}$ la matriu de rotació obtinguda fent la transformació del quaternió rebut pel tòpic /workpiece_pose.

La translació final per a la posició d'aproximació dels punts del pla ABCDE serà:

$$Trans_{ABCDE} = Trans_{Norm} + Trans_{aresta} + Trans_{peça} \quad (40)$$

Sent $Trans_{peça}$ la posició obtinguda pel tòpic /workpiece_pose i $Trans_{aresta}$ la posició de l'aresta assenyalada respecte l'origen de la peça.

Per al pla FGHI la posició i orientació d'aproximació s'ha definit com es mostra a la Figura 30. A una distància del pla de 10 centímetres i amb l'eix Z perpendicular al pla.

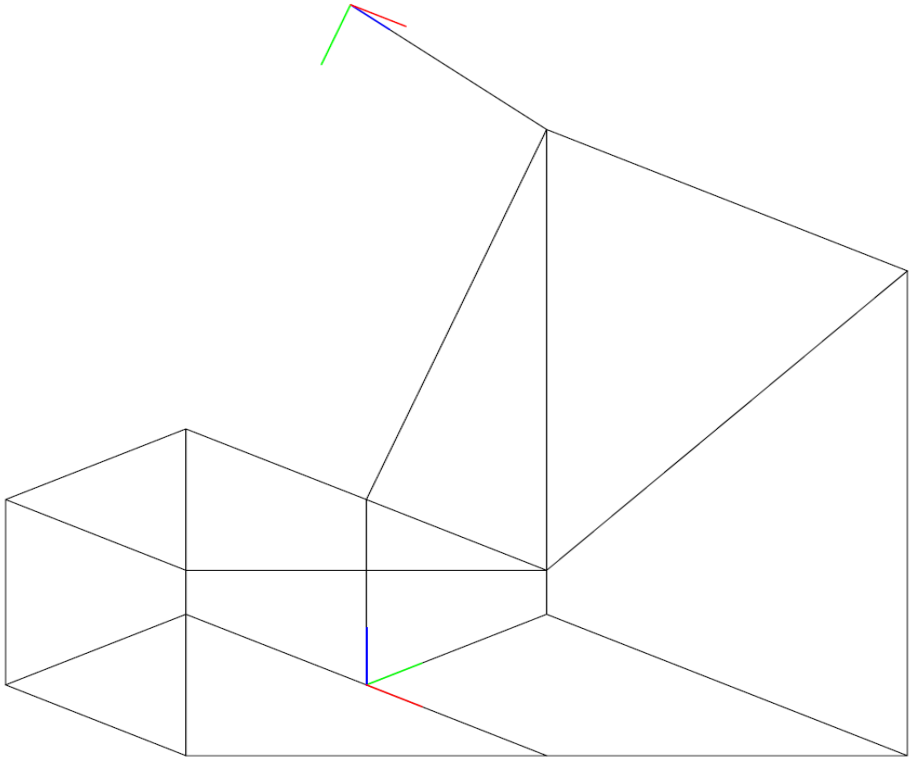

Figura 30 Posició i orientació d'aproximació per a un dels punts del pla FGHI

Pel que fa a la posició, s'ha de fer una translació de 10 centímetres en direcció a la normal del pla aresta en qüestió (Equació (37)). En canvi, pel que fa a l'orientació s'ha de realitzar una rotació sobre l'eix X de -135º respecte la base de la peça.

$$Rot(x,-135)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(-135) & -\sin(-135)\\ 0 & \sin(-135) & \cos(-135)\end{bmatrix} \qquad (41)$$

S'ha de tenir en compte, però, que a aquests càlculs se'ls ha d'afegir la rotació i la translació pertinents a la posició de la peça real obtinguts pel tòpic /workpiece_pose.

Per tant, la rotació final per a la posició d'aproximació dels punts del pla FGHI serà:

$$Rot_{FGHI} = Rot(x,-135)\cdot Rot_{peça} \qquad (42)$$

Sent $Rot_{peça}$ la matriu de rotació obtinguda fent la transformació del quaternió rebut pel tòpic /workpiece_pose.

La translació final per a la posició d'aproximació dels punts del pla FGHI serà:

$$Trans_{FGHI} = Trans_{Norm} + Trans_{aresta} + Trans_{peça} \qquad (43)$$

Sent $Trans_{peça}$ la posició obtinguda pel tòpic /workpiece_pose i $Trans_{aresta}$ la posició de l'aresta assenyalada respecte l'origen de la peça.

Per al pla JKLMN la posició i orientació d'aproximació s'ha definit com es mostra a la Figura 31. A una distància del pla de 10 centímetres i amb l'eix Z perpendicular al pla.

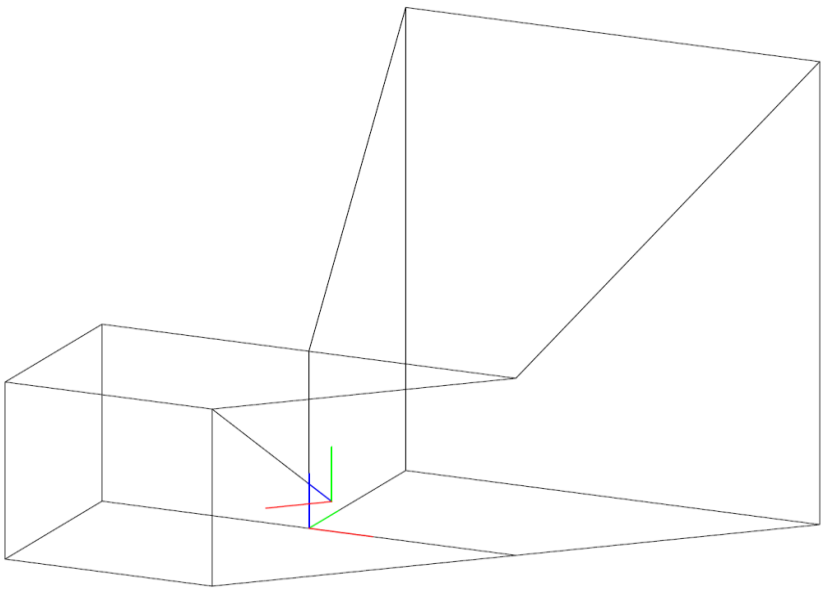

Figura 31 Posició i orientació d'aproximació per a un dels punts del pla JKLMN

Pel que fa a la posició, s'ha de fer una translació de 10 centímetres en direcció a la normal del pla aresta en qüestió (Equació (37)). Per altra banda, pel que fa a l'orientació, respecte la base de la peça s'ha de realitzar una rotació sobre l'eix Y de -135º i posteriorment una rotació sobre l'eix X de 90º.

$$Rot(y,-135)=\begin{bmatrix}\cos(-135) & 0 & \sin(-135)\\ 0 & 1 & 0\\ -\sin(-135) & 0 & \cos(-135)\end{bmatrix} \quad (44)$$

$$Rot(x,90)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(90) & -\sin(90)\\ 0 & \sin(90) & \cos(90)\end{bmatrix} \quad (45)$$

S'ha de tenir en compte, però, que a aquests càlculs se'ls ha d'afegir la rotació i la translació pertinents a la posició de la peça real obtinguts pel tòpic /workpiece_pose.

Per tant, la rotació final per a la posició d'aproximació dels punts del pla JKLMN serà:

$$Rot_{JKLMN} = Rot(x,90)\cdot Rot(y,-135)\cdot Rot_{peça} \quad (46)$$

Sent $Rot_{peça}$ la matriu de rotació obtinguda fent la transformació del quaternió rebut pel tòpic /workpiece_pose.

La translació final per a la posició d'aproximació dels punts del pla FGHI serà:

$$Trans_{JKLMN} = Trans_{Norm} + Trans_{aresta} + Trans_{peça} \quad (47)$$

Sent $Trans_{peça}$ la posició obtinguda pel tòpic /workpiece_pose i $Trans_{aresta}$ la posició de l'aresta assenyalada respecte l'origen de la peça.

Un cop obtingudes les posicions d'aproximació per a totes les arestes dels plans, s'ha de fer la transformació de les matrius de rotació a quaternions. Per fer-ho, s'ha implementat una funció que obté les components del quaternió fent el desenvolupament invers de l'equació (35).

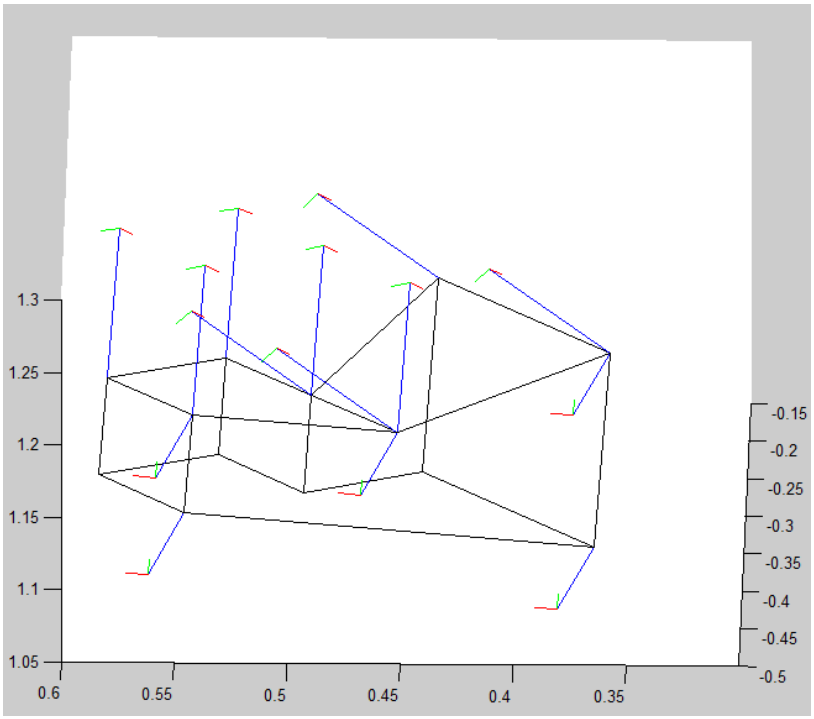

Figura 32 Representació de les posicions d'aproximació de cadascuna de les arestes

5.3.2. Funció ReadElbow Matlab

Aquesta funció s'executa quan rep noves dades del tòpic /right_elbow el qual, com ja s'ha comentat anteriorment, proporciona la posició del colze dret de l'operari.

Aquesta funció únicament emmagatzema la posició xyz del colze un una variable global per tal de poder-la utilitzar a la funció ReadHandProcess.

5.3.3. Funció ReadHandProcess Matlab

Aquesta funció s'executa quan rep noves dades del tòpic /right_hand el qual, com ja s'ha comentat anteriorment, proporciona la posició de la mà dreta de l'operari.

Un cop obtinguda la posició de la mà, s'obtenen les equacions de la recta que passa pels punts donats per la posició del colze i la posició de la mà de l'operari aplicant l'equació (6):

$$\begin{cases} A'x + B'y + C'z + D' = 0 \\ A''x + B''y + C''z + D'' = 0 \end{cases} \quad (48)$$

D'aquesta manera es pot determinar el punt d'intersecció resolent el sistema d'equacions:

$$\begin{cases} Ax + By + Cz + D = 0 \\ A'x + B'y + C'z + D' = 0 \\ A''x + B''y + C''z + D'' = 0 \end{cases} \quad (49)$$

Un cop s'ha determinat el punt d'intersecció per a tots els plans, s'ha de comprovar si aquest punt està dins de la regió del pla que correspon a la peça. Per fer això, s'han utilitzat les regions definides anteriorment (vegeu Figura 28) .

Per determinar si el punt està dins d'alguna de les regions s'ha fet servir una funció que determina l'orientació dels triangles (vegeu apartat 4.1.3) que formen cada regió i, si el punt per on creua la recta que formen el colze i la mà està dins d'aquesta, els triangles formats per aquest punt i els punts que delimiten la regió tindran la mateixa orientació.

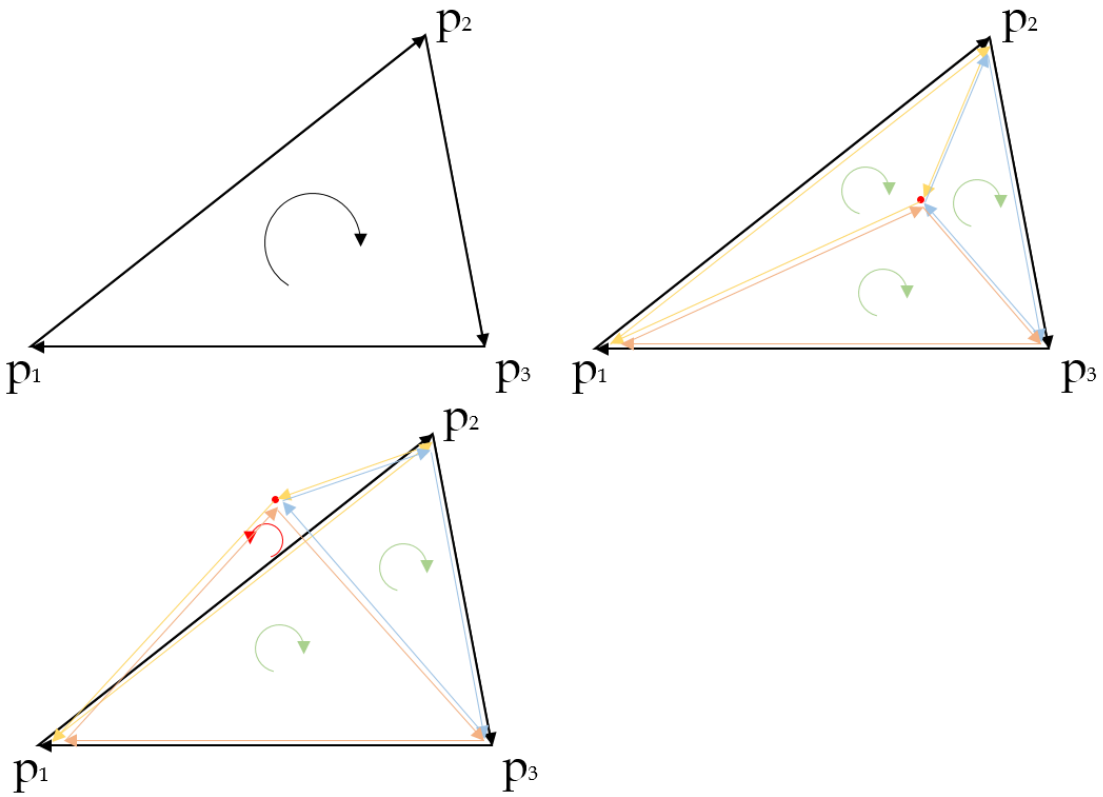

Figura 33 Sentit de gir dels triangles per determinar si està dins la regió

Per tal d'assegurar que l'operari realment està assenyalant la regió detectada, s'enviarà la posició a través del tòpic /intersection sempre que s'hagi detectat 10 vegades seguides que la regió assenyalada és la mateixa.

Per altra banda, cada cop que es publica una posició al tòpic /intersection també es representa de forma gràfica:

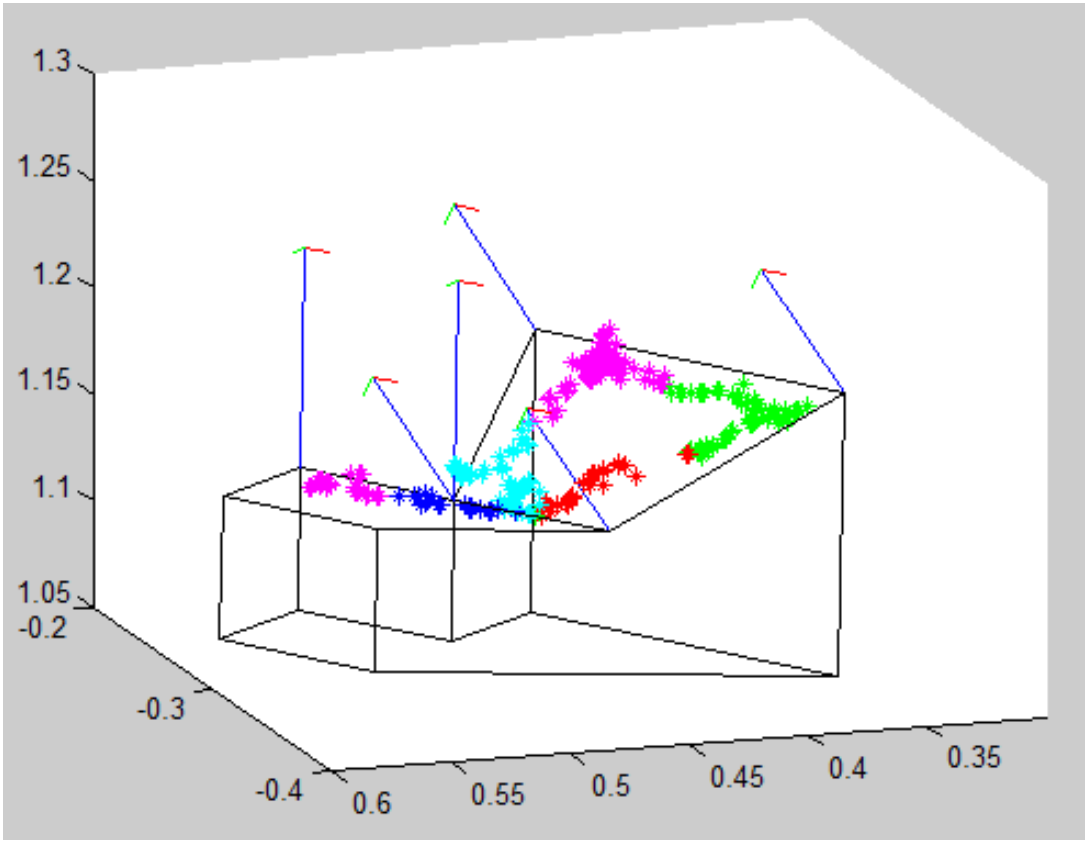

Figura 34 Exemple de representació gràfica dels punts assenyalats a la peça

5.4. Node Posicionament robot[4]

Aquest node és l'encarregat de moure el robot per posicionar l'eina en la posició i orientació rebudes a través del tòpic /intersection.

Per poder moure el robot utilitzant ROS-Industrial s'ha fet servir un paquet desenvolupat per al robot UR5 (aquest paquet està actualment en fase experimental).

[4] Aquest node ha estat desenvolupat per la fundació Ascamm. S'explica a grans trets els procediments utilitzats.

Aquest node utilitza un software anomenat *MoveIt!* que proporciona una plataforma fàcil d'utilitzar per al desenvolupament d'aplicacions de robòtica. Com ja s'ha comentat, aquest software per al robot UR5 encara està en fase experimental i accions com ara la planificació de les trajectòries encara no les realitza d'una manera òptima.

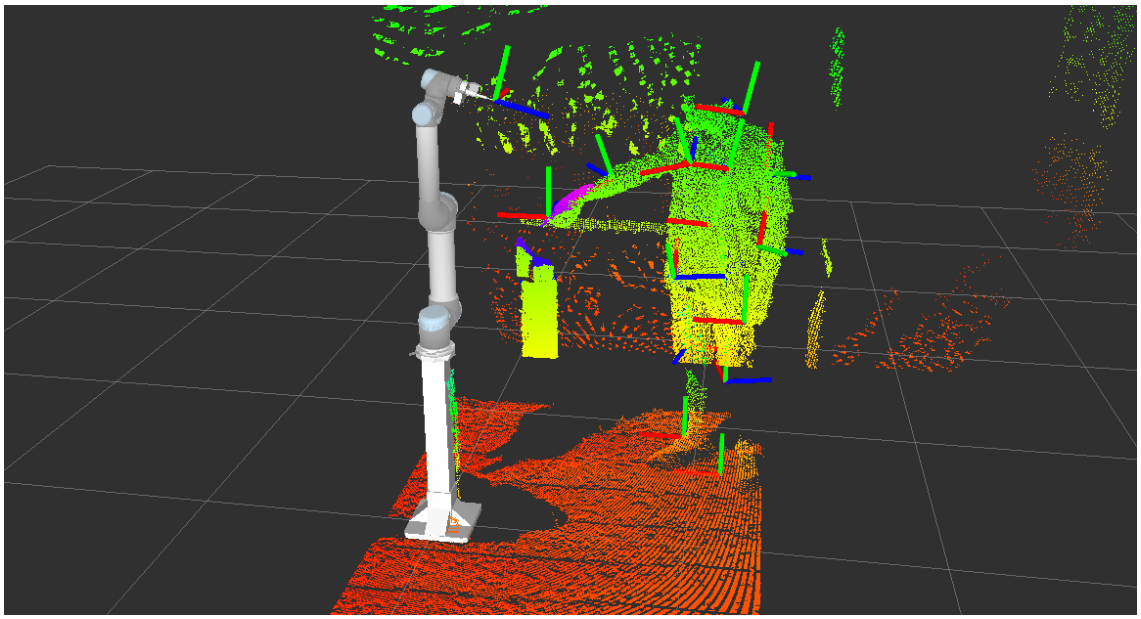

Figura 35 Operari assenyalant un punt de la peça

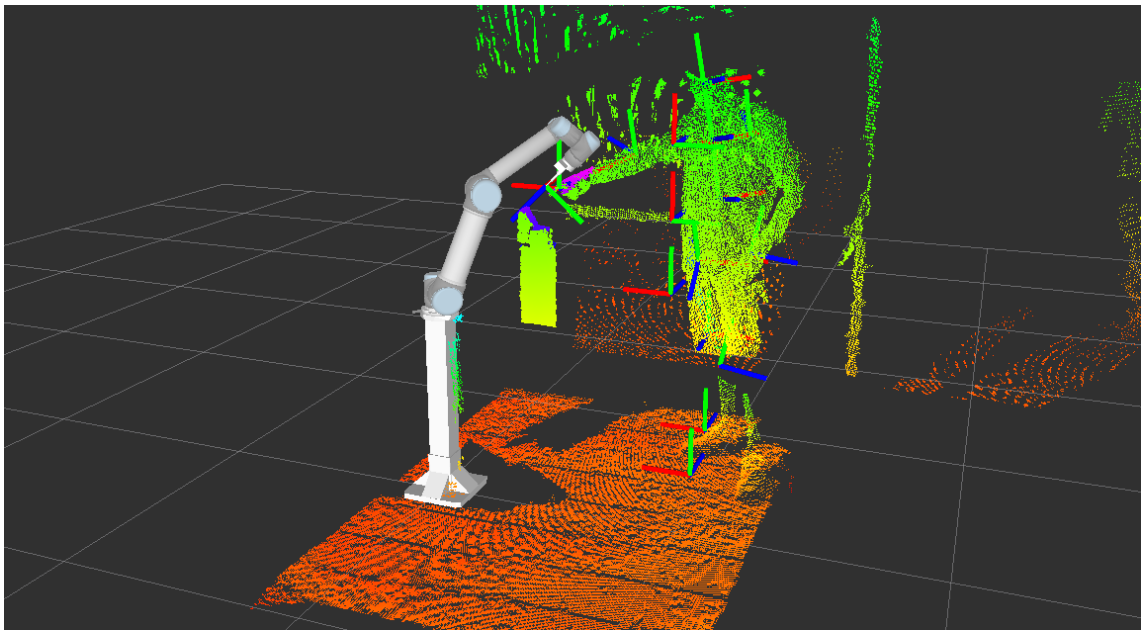

Figura 36 Robot posicionat al punt de la peça assenyalat

6. Conclusions

6.1. Resultats i conclusions

Fent referència als objectius plantejats a la introducció, que havien motivat el plantejament i desenvolupament d'aquest Treball final de Grau, es pot afirmar que, en línies generals, aquests s'han assolit amb uns resultats satisfactoris.

L'objectiu principal era desenvolupar un sistema de control mitjançant ROS i visió 3D per tal que, quan un operari assenyalés un punt d'una peça situada dins l'espai de treball del robot, aquest s'hi aproximés per tal de realitzar-hi una operació. L'objectiu s'ha assolit de tal manera que el robot es situa a la posició d'aproximació de l'aresta de la regió assenyalada orientant l'eina perpendicularment al pla on s'ha de realitzar l'operació.

Per assolir aquest objectiu s'han hagut d'anar assolit diferents fites les quals es poden agrupar en tres grans blocs: Formació ROS, ROS en Matlab i Implementació en l'entorn real.

6.1.1. Formació ROS

Una de les fases que ha portat més temps i que ha estat la base pel desenvolupament d'aquest treball ha estat la formació dins de l'entorn ROS.

ROS és un entorn en el qual no s'ha treballat dins els estudis de grau precedents a aquest treball, per això la fase de formació ha estat tant important.

En aquesta fase s'ha après com funciona l'entorn, què són els nodes, per a què serveixen els tòpics, en què es diferencien dels serveis, etc. (vegeu apartat 2.2). Amb aquests coneixements de base, també s'ha après com funciona l'entorn de ROS Industrial, en especial el paquet MoveIt!

Amb aquests coneixements previs es van desenvolupar els primers nodes en Ubuntu.

Un cop ja es tenien els coneixements bàsics es va posar en marxa l'entorn de simulació EuRoC on es van trobar molts problemes sobretot per la capacitat dels ordinadors dels quals es disposava.

Amb l'entorn de simulació en marxa es van poder desenvolupar els primers algorismes en C++ utilitzant llibreries d'OpenCV i PointClouds. D'aquesta manera es va poder col·laborar amb el desenvolupament dels nodes per detectar la peça i reconèixer l'operari.

6.1.2. ROS Matlab

Amb els coneixements adquirits a la fase de formació ROS es va poder donar peu a la següent fase, la qual ha estat el cos d'aquest treball: la implementació de ROS en l'entorn Matlab.

En aquesta fase s'ha après a implementar nodes ROS en Matlab, a realitzar una comunicació mitjançant ROS entre dos hosts (un d'ells programat amb Matlab sobre Windows i un altre programat en C++ sobre Ubuntu) i a crear els nodes i les funcions necessàries per tal d'obtenir el punt on està assenyalant l'operari.

Aquesta fase també ha presentat forces dificultats, sobretot pel fet de l'escassesa d'informació existent en la implementació de ROS en Matlab.

6.1.3. Implementació en l'entorn real

Aquesta fase quan es va desenvolupar la proposta era una fase què no es sabia si el temps permetria desenvolupar-la però, afortunadament, s'ha pogut desenvolupar.

En aquesta fase s'han implementat els algorismes desenvolupats en un entorn real.

Per assolir-ho es va haver de fixar el sistema de comunicació i quin tipus de missatges s'utilitzarien.

Per altra banda també es van presentar diverses dificultats com haver de configurar els controladors del robot per que funcionin amb ROS, fixar les posicions de les Kinect per fer correctament els canvis de base o tenir

en compte que les dades dels núvols de punts reals no són tant precises com les de l'entorn simulat.

Veient tot això es pot concloure que s'ha realitzat un sistema funcional per tal d'aproximar un robot a una zona d'una peça on ha assenyalat un operari. Tot i així, s'ha de tenir en compte que aquest treball forma part d'un projecte de recerca que actualment encara està en desenvolupament i que, per tant, pot tenir continuïtat amb tasques futures.

6.2. Tasques futures

Amb l'objectiu assolit en aquest treball, el robot es situa a una posició d'aproximació a la zona on ha de realitzar l'operació. Partint del posicionament del robot, podria ampliar aquest treball la següent tasca que proposa el repte EuRoC: afinar la posició del robot per tal de realitzar una operació en aquella zona de la peça.

Per realitzar aquesta tasca es proposa utilitzar una eina pel robot que disposi d'una càmera estereoscòpica i una rebladora (veieu Figura 37).

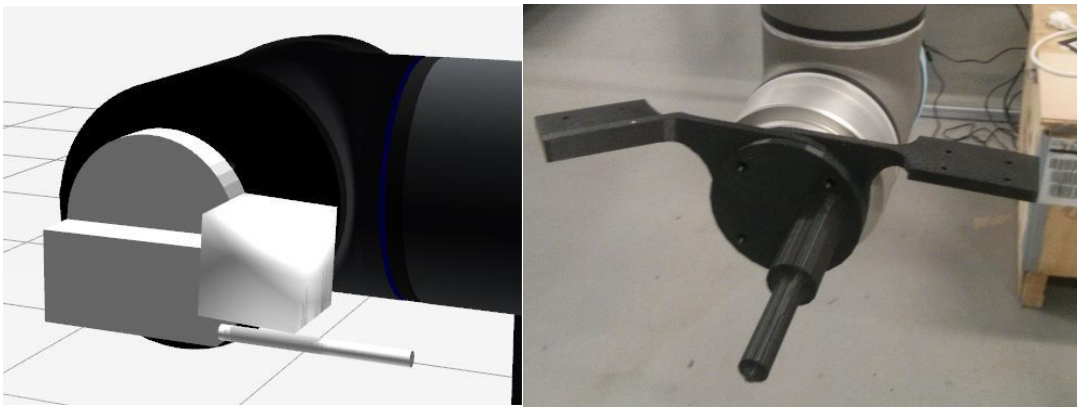

Figura 37 Rebladora amb càmera estereoscòpica unida al robot (entorn virtual i entorn real)

6.3. Valoració Personal

Per últim, cal dir que, personalment, la valoració d'aquest projecte és molt positiva. Realitzant aquest projecte he pogut consolidar alguns coneixements previs adquirits durant el grau, tot i que, per altra banda, també he pogut entrar en un món que per a mi era totalment desconegut.

Tot i no saber-ne res, però, he pogut afrontar-lo amb èxit gràcies a les eines i coneixements adquirits aquests anys en el grau; i és que ser enginyer bàsicament és fer el què he estat fent durant aquest projecte: amb el suport d'uns fonaments d'enginyeria sòlids, es pot arribar a qualsevol fi, tot i que en un inici et trobis amb un món desconegut.

7. Bibliografia

A. Barrientos, L.P Peñin, C.Balaguer, R.Aracil (2007). *Fundamentos de Robótica* (2n ed.) Madrid: Mc.GrawHill.

J. M. O'Kane (2013) *A Gentle Introduction to ROS*. Consultat 15 octubre 2014 des de http://www.cse.sc.edu/~jokane/agitr/

W. R. Hamilton (1899) *Elements of Quaternions* (1a ed.) Cambridge: Cambridge University Press.

G. Masferrer i Caralt, J. Ordeix i Rigo, M. Serra i Serra (2013). *Mòdul 2. Sistemes Robotitzats: Fonaments Matemàtics*. Apunts procedents de UVic-UCCMoodle.

Adreces d'Internet:

Robot Operating System. (2014). Consultat el 15 setembre 2014, des de http://www.wiki.ros.org/hydro

MathWorks. (2014). *Robot Operating System*. Consultat 08 desembre 2014, des de http://es.mathworks.com/help/robotics/robot-operating-system-ros.html?s_tid=srchtitle

MathWorks. (2014). *Use Matlab_ros_io_package to interact with the turlebot*. Consultat 08 desembre 2014, des de http://www.mathworks.com/matlabcentral/fileexchange/44853-use-matlab-ros-i-o-package-to-interact-with-the-turtlebot-simulator-in-gazebo

Vitutor. (2015). *Equaciones de la recta en el espacio.* Consultat 10 gener 2015 des de http://www.vitutor.com/analitica/recta/ecuaciones_recta.html

Vitutor. (2015). *Equaciones del plano en el espacio.* Consultat 14 gener 2015 des de http://www.vitutor.com/analitica/recta/ecuaciones_plano.html

TecDigital. (2015). *Punto interior de un triangulo.* Consultat 02 febrer 2015 des de http://tecdigital.tec.ac.cr/revistamatematica/Contribucionesv3n2002/WMoraMatProg/pag2.html

Universidad de Sonora. (2015). *Rotaciones y Translaciones.* Consultat 25 març 2015 des de http://tesis.uson.mx/digital/tesis/docs/21070/Capitulo4.pdf

XIX Congreso Internacional de Ingeniería Mecánica. (2015). *Parámetros redundantes para Rrotación y Translación en Cinemática.* Consultat 04 abril 2015 des de http://www.xixcnim.uji.es/CDActas/Documentos/ComunicacionesOrales/02-01.pdf

Universidad de León. (2015). *OpenNI tutorial 2: Cloud processing (basic) Normal estimation.* Consultat 10 maig 2015 des de http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_2:_Cloud_processing_(basic)#Normal_estimation

Universidad de León. (2015). *OpenNI tutorial 4: 3D Object recognition.* Consultat 10 maig 2015 des de http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_4:_3D_object_recognition_(descriptors)#FPFH

Universidad de León. (2015). *OpenNI tutorial 3: Cloud processing (advanced) Model fitting.* Consultat 10 maig 2015 des de http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_3:_Cloud_processing_(advanced)#Model_fitting_.28RANSAC.29

Universidad de León. (2015). *OpenNI tutorial 3: Cloud processing (advanced).* Consultat 10 maig 2015 des de http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_3:_Cloud_processing_(advanced)#RANSAC

Universidad de León. (2015). *OpenNI tutorial 3: Cloud processing (advanced) Iterative Closest Point.* Consultat 10 maig 2015 des de http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_3:_Cloud_processing_(advanced)#Iterative_Closest_Point_.28ICP.29

Fotografies:

Figura 5 Robot UR5 [fotografia]. (n.d.). Consultat des de http://www.globalrobots.com/products/DF072994B6D049DD92BD311E37036C6F.jpg

Figura 6 Sensor Kinect [fotografia]. (n.d.). Consultat des de http://blog.robotiq.com/bid/40428/Using-The-Kinect-For-Robotic-Manipulation

Abans d'aplicar el filtre [fotografia]. (n.d.). Consultat des de http://pointclouds.org/documentation/tutorials/passthrough.php#passthrough

Després d'aplicar el filtre [fotografia]. (n.d.). Consultat des de http://pointclouds.org/documentation/tutorials/passthrough.php#passthrough

Després d'aplicar el filtre Voxel gird [fotografia]. (n.d.). Consultat des de http://pointclouds.org/documentation/tutorials/voxel_grid.php#voxelgrid

Abans d'aplicar el filtre Voxel gird [fotografia]. (n.d.). Consultat des de http://pointclouds.org/documentation/tutorials/voxel_grid.php#voxelgrid

Representació de les normals obtingudes del núvol de punts [fotografia]. (n.d.). Consultat des de__http://pointclouds.org/documentation/tutorials/voxel_grid.php#voxelgrid

Posició Psi [fotografia]. (n.d.). Consultat des de https://slm-assets0.secondlife.com/assets/10194789/lightbox/psipose.jpg?1408925843

Figura 22 Articulacions detectades per la Kinect [fotografia]. (n.d.). Consultat des de http://blogs.msdn.com/cfs-filesy stemfile.ashx/_key/communityserver-blogs-components-weblogfiles/00-00-01-40-66-metablogapi/1884.image_5F00_01FB8F38.png

8. Annexos

A continuació es mostra el codi desenvolupat per al node Matlab.

Node.m

```
clc
close all
clear all
global ColzeDret MaDreta pg pose q a1 a2 a3 a4 b1 b2 b3 b4 c1 c2 c3 c4
d1 d2 d3 d4 e1 e2 e3 e4 Aa Ba Ca Da f1 f2 f3 f4 g1 g2 g3 g4 h1 h2 h3
h4 i1 i2 i3 i4 Af Bf Cf Df Aj Bj Cj Dj contA contB contC contD contE
contF contG contH contI contJ contK contL contM contN pas intersection
n msg publisher;
pg=0;contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;c
ontI=0;contJ=0;contK=0;contL=0;contM=0;contN=0;pas=0;

% node1 = rosmatlab.node('ReadPose', 'http://192.168.1.138:11311/');
%Casa
% node1 = rosmatlab.node('ReadPose', 'http://10.9.145.147:11311/');
%UVIC
node1 = rosmatlab.node('ReadPose', 'http://192.168.1.10:11311/');
%Ascamm

publisher = node1.addPublisher('/intersection', 'geometry_msgs/Pose');
subscriberPOSE = rosmatlab.subscriber('/workpiece_pose',
'geometry_msgs/Pose', 25, node1);
subscriberHAND = rosmatlab.subscriber('/hand_pose',
'geometry_msgs/Pose', 25, node1);
subscriberELBOW = rosmatlab.subscriber('/elbow_pose',
'geometry_msgs/Pose', 25, node1);
subscriberPOSE.setOnNewMessageListeners({@ReadPose});
subscriberELBOW.setOnNewMessageListeners({@ReadElbow});
subscriberHAND.setOnNewMessageListeners({@ReadHandProcess});
msg = rosmatlab.message('geometry_msgs/Pose', node1);

str = input('Enter any key to terminate','s');
subscriberPOSE.delete();
subscriberELBOW.delete();
subscriberHAND.delete();
publisher.delete();
node1.delete();
```

ReadPose.m

```
function ReadPose(message)

    global pose q a1 a2 a3 a4 b1 b2 b3 b4 c1 c2 c3 c4 d1 d2 d3 d4 e1
e2 e3 e4 Aa Ba Ca Da f1 f2 f3 f4 g1 g2 g3 g4 h1 h2 h3 h4 i1 i2 i3 i4
Af Bf Cf Df j1 j2 j3 j4 k1 k2 k3 k4 l1 l2 l3 l4 m1 m2 m3 m4 n1 n2 n3
n4 Aj Bj Cj Dj nJ qJ nF nA pas qF qA inta intb intc intd inte intf
intg inth inti intj intk intl intm intn

    if (pas==0)
        poseX = message.getPosition().getX();
```

```
poseY = message.getPosition().getY();
poseZ = message.getPosition().getZ();
qX = message.getOrientation().getX();
qY = message.getOrientation().getY();
qZ = message.getOrientation().getZ();
qW = message.getOrientation().getW();

pose=[poseX poseY poseZ];
q=[qW qX qY qZ];


A1=[-0.075 0.01 0.065];
A2=[-0.075 -0.0325 0.065];
A3=[-0.01625 -0.0325 0.065];
A4=[-0.0325 0.01 0.065];

B1=[-0.075 -0.0325 0.065];
B2=[-0.075 -0.075 0.065];
B3=[-0.0325 -0.075 0.065];
B4=[-0.01625 -0.0325 0.065];

C1=[-0.01625 -0.0325 0.065];
C2=[-0.0325 -0.075 0.065];
C3=[0 -0.075 0.065];
C4=[0.0425 -0.0325 0.065];

D1=[0.0325 0 0.065];
D2=[-0.01625 -0.0325 0.065];
D3=[0.0425 -0.0325 0.065];
D4=[0.075 0 0.065];

E1=[-0.0325 0 0.065];
E2=[-0.01625 -0.0325 0.065];
E3=[0.0325 0 0.065];
E4=[0 0 0.065];

a1=RotTrans(A1,pose,q);
a2=RotTrans(A2,pose,q);
a3=RotTrans(A3,pose,q);
a4=RotTrans(A4,pose,q);

b1=RotTrans(B1,pose,q);
b2=RotTrans(B2,pose,q);
b3=RotTrans(B3,pose,q);
b4=RotTrans(B4,pose,q);

c1=RotTrans(C1,pose,q);
c2=RotTrans(C2,pose,q);
c3=RotTrans(C3,pose,q);
c4=RotTrans(C4,pose,q);

d1=RotTrans(D1,pose,q);
d2=RotTrans(D2,pose,q);
d3=RotTrans(D3,pose,q);
d4=RotTrans(D4,pose,q);

e1=RotTrans(E1,pose,q);
e2=RotTrans(E2,pose,q);
```

```
e3=RotTrans(E3,pose,q);
e4=RotTrans(E4,pose,q);

%Equació del pla ABCDE (agafant els punts a1 a2 a3)
Aa=det([a2(2)-a1(2) a3(2)-a1(2);a2(3)-a1(3) a3(3)-a1(3)]);
Ba=-det([a2(1)-a1(1) a3(1)-a1(1);a2(3)-a1(3) a3(3)-a1(3)]);
Ca=det([a2(1)-a1(1) a3(1)-a1(1);a2(2)-a1(2) a3(2)-a1(2)]);
Da=-Aa*a1(1)-Ba*a1(2)-Ca*a1(3);
%Normal del pla ABCDE
nA(1)=-Aa/sqrt(Aa^2+Ba^2+Ca^2);
nA(2)=-Ba/sqrt(Aa^2+Ba^2+Ca^2);
nA(3)=-Ca/sqrt(Aa^2+Ba^2+Ca^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
qpA=rot2quat(rotx(180));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

F1=[-0.01 0.07207 0.1370711];
F2=[-0.01 0.0325 0.0975];
F3=[0.04875 0.0325 0.0975];
F4=[0.067854 0.07009 0.1370711];

G1=[-0.01 0.0325 0.0975];
G2=[-0.01 0 0.065];
G3=[0.0325 0 0.065];
G4=[0.04875 0.0325 0.0975];

H1=[0.04875 0.0325 0.0975];
H2=[0.0325 0 0.065];
H3=[0.075 0 0.065];
H4=[0.1075 0.0325 0.0975];

I1=[0.06854 0.07207 0.1370711];
I2=[0.04875 0.0325 0.0975];
I3=[0.1075 0.0325 0.0975];
I4=[0.14707 0.07207 0.1370711];

f1=RotTrans(F1,pose,q);
f2=RotTrans(F2,pose,q);
f3=RotTrans(F3,pose,q);
f4=RotTrans(F4,pose,q);

g1=RotTrans(G1,pose,q);
g2=RotTrans(G2,pose,q);
g3=RotTrans(G3,pose,q);
g4=RotTrans(G4,pose,q);

h1=RotTrans(H1,pose,q);
h2=RotTrans(H2,pose,q);
h3=RotTrans(H3,pose,q);
h4=RotTrans(H4,pose,q);

i1=RotTrans(I1,pose,q);
i2=RotTrans(I2,pose,q);
i3=RotTrans(I3,pose,q);
i4=RotTrans(I4,pose,q);

%Equació del pla FGHI (agafant els punts f1 f2 f3)
Af=det([f2(2)-f1(2) f3(2)-f1(2);f2(3)-f1(3) f3(3)-f1(3)]);
```

```
Bf=-det([f2(1)-f1(1) f3(1)-f1(1);f2(3)-f1(3) f3(3)-f1(3)]);
Cf=det([f2(1)-f1(1) f3(1)-f1(1);f2(2)-f1(2) f3(2)-f1(2)]);
Df=-Af*f1(1)-Bf*f1(2)-Cf*f1(3);
%Normal del pla FGHI
nF(1)=-Af/sqrt(Af^2+Bf^2+Cf^2);
nF(2)=-Bf/sqrt(Af^2+Bf^2+Cf^2);
nF(3)=-Cf/sqrt(Af^2+Bf^2+Cf^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
qpF=rot2quat(rotx(-135));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


J1=[-0.0070711 -0.0720711 0.075];
J2=[-0.0070711 -0.0720711 0.0325];
J3=[0.0289645 -0.0360355 0.0325];
J4=[0.0289645 -0.0360355 0.075];

K1=[-0.0070711 -0.0720711 0.0325];
K2=[-0.0070711 -0.0720711 -0.01];
K3=[0.0289645 -0.0360355 -0.01];
K4=[0.0289645 -0.0360355 0.0325];

L1=[0.0830178 0.0180178 0.0825];
L2=[0.0830178 0.0180178 -0.01];
L3=[0.1370711 0.0720711 -0.01];
L4=[0.1370711 0.0720711 0.0825];

M1=[0.1010355 0.0360355 0.1325];
M2=[0.0830178 0.0180178 0.0825];
M3=[0.1370711 0.0720711 0.0825];
M4=[0.1370711 0.0720711 0.19];

N1=[0.0289645 -0.0360355 0.075];
N2=[0.065 0 0.0325];
N3=[0.1010355 0.0360355 0.1325];
N4=[0.065 0 0.075];

j1=RotTrans(J1,pose,q);
j2=RotTrans(J2,pose,q);
j3=RotTrans(J3,pose,q);
j4=RotTrans(J4,pose,q);

k1=RotTrans(K1,pose,q);
k2=RotTrans(K2,pose,q);
k3=RotTrans(K3,pose,q);
k4=RotTrans(K4,pose,q);

l1=RotTrans(L1,pose,q);
l2=RotTrans(L2,pose,q);
l3=RotTrans(L3,pose,q);
l4=RotTrans(L4,pose,q);

m1=RotTrans(M1,pose,q);
m2=RotTrans(M2,pose,q);
m3=RotTrans(M3,pose,q);
m4=RotTrans(M4,pose,q);

n1=RotTrans(N1,pose,q);
```

```
n2=RotTrans(N2,pose,q);
n3=RotTrans(N3,pose,q);
n4=RotTrans(N4,pose,q);

%Equació del pla JKLMN (agafant els punts j1 j2 j3)
Aj=det([j2(2)-j1(2) j3(2)-j1(2);j2(3)-j1(3) j3(3)-j1(3)]);
Bj=-det([j2(1)-j1(1) j3(1)-j1(1);j2(3)-j1(3) j3(3)-j1(3)]);
Cj=det([j2(1)-j1(1) j3(1)-j1(1);j2(2)-j1(2) j3(2)-j1(2)]);
Dj=-Aj*j1(1)-Bj*j1(2)-Cj*j1(3);

%Normal del pla JKLMN
nJ(1)=-Aj/sqrt(Aj^2+Bj^2+Cj^2);
nJ(2)=-Bj/sqrt(Aj^2+Bj^2+Cj^2);
nJ(3)=-Cj/sqrt(Aj^2+Bj^2+Cj^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RX=rotx(90);
RY=roty(-135);
RotJ=RX*RY;
qpJ=rot2quat(RotJ);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%Representació gràfica%%%
A1=[-0.065 -0.065 0.065];
A2=[-0.065 0 0.065];
A3=[0 -0.065 0.065];
A4=[0 0 0.065];
A5=[0.065 0 0.065];

B1=[0 0.065 0.13];
B3=[0.13 0.065 0.13];

C1=[-0.065 -0.065 0];
C2=[-0.065 0 0];
C3=[0 -0.065 0];
C4=[0 0 0];
C5=[0.065 0 0];

D1=[0 0.065 0];
D3=[0.13 0.065 0];

A1=RotTrans(A1,pose,q);
A2=RotTrans(A2,pose,q);
A3=RotTrans(A3,pose,q);
A4=RotTrans(A4,pose,q);
A5=RotTrans(A5,pose,q);

B1=RotTrans(B1,pose,q);
B3=RotTrans(B3,pose,q);

C1=RotTrans(C1,pose,q);
C2=RotTrans(C2,pose,q);
C3=RotTrans(C3,pose,q);
C4=RotTrans(C4,pose,q);
C5=RotTrans(C5,pose,q);

D1=RotTrans(D1,pose,q);
D3=RotTrans(D3,pose,q);
```

```
        x=[A4(1) A5(1) B3(1) B1(1) A4(1) A2(1) A1(1) A3(1) A5(1)];
        y=[A4(2) A5(2) B3(2) B1(2) A4(2) A2(2) A1(2) A3(2) A5(2)];
        z=[A4(3) A5(3) B3(3) B1(3) A4(3) A2(3) A1(3) A3(3) A5(3)];
        line(x,y,z,'Color','k')
        x=[C5(1) D3(1) D1(1) C4(1) C2(1) C1(1) C3(1) C5(1)];
        y=[C5(2) D3(2) D1(2) C4(2) C2(2) C1(2) C3(2) C5(2)];
        z=[C5(3) D3(3) D1(3) C4(3) C2(3) C1(3) C3(3) C5(3)];
        line(x,y,z,'Color','k')
        x=[A1(1) C1(1)];
        y=[A1(2) C1(2)];
        z=[A1(3) C1(3)];
        line(x,y,z,'Color','k')
        x=[A2(1) C2(1)];
        y=[A2(2) C2(2)];
        z=[A2(3) C2(3)];
        line(x,y,z,'Color','k')
        x=[A3(1) C3(1)];
        y=[A3(2) C3(2)];
        z=[A3(3) C3(3)];
        line(x,y,z,'Color','k')
        x=[A4(1) C4(1)];
        y=[A4(2) C4(2)];
        z=[A4(3) C4(3)];
        line(x,y,z,'Color','k')
        x=[B1(1) D1(1)];
        y=[B1(2) D1(2)];
        z=[B1(3) D1(3)];
        line(x,y,z,'Color','k')
        x=[B3(1) D3(1)];
        y=[B3(2) D3(2)];
        z=[B3(3) D3(3)];
        line(x,y,z,'Color','k')
        hold all
        %%%%%%%%%Fi representació gràfica%%%%%%%%%%%%%%%%

        %%%%%%INTERSECTIONS%%%%%%%%%%%%%
        nA10=-nA./10;
        inta=A2+nA10; intb=A1+nA10; intc=A3+nA10; intd=A5+nA10;
inte=A4+nA10;
        nF10=-nF./10;
        intf=B1+nF10; intg=A4+nF10; inth=A5+nF10; inti=B3+nF10;
        nJ10=-nJ./10;
        intj=A3+nJ10; intk=C3+nJ10; intl=D3+nJ10; intm=B3+nJ10;
intn=A5+nJ10;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %%%%%%%%%%QUATERNIONS%%%%%%%%%%
        qA=RotTransNormPla(q,qpA);
        qF=RotTransNormPla(q,qpF);
        qJ=RotTransNormPla(q,qpJ);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        pause(1);

        pas=1
    end
end
```

ReadElbow.m

```
function ReadElbow(message)
  global ColzeDret
  poseX = message.getPosition().getX();
  poseY = message.getPosition().getY();
  poseZ = message.getPosition().getZ();

  ColzeDret=[poseX poseY poseZ];

end
```

ReadHandProcess.m

```
function ReadHandProcess(message)
    global publisher msg ColzeDret MaDreta a1 a2 a3 a4 b1 b2 b3 b4 c1
c2 c3 c4 d1 d2 d3 d4 e1 e2 e3 e4 Aa Ba Ca Da f1 f2 f3 f4 g1 g2 g3 g4
h1 h2 h3 h4 i1 i2 i3 i4 Af Bf Cf Df contA contB contC contD contE
contF contG contH contI contJ contK contL contM contN inta intb intc
intd inte intf intg inth inti intj intk intl intm intn qA qF
intersection n nF nA j1 j2 j3 j4 k1 k2 k3 k4 l1 l2 l3 l4 m1 m2 m3 m4
n1 n2 n3 n4 Aj Bj Cj Dj nJ qJ


sA=0;sB=0;sC=0;sD=0;sE=0;sF=0;sG=0;sH=0;sI=0;sJ=0;sK=0;sL=0;sM=0;sN=0;

    poseX = message.getPosition().getX();
    poseY = message.getPosition().getY();
    poseZ = message.getPosition().getZ();
    MaDreta=[poseX poseY poseZ];

    %Recta ColzeMa
    ux=-(MaDreta(1,1)-ColzeDret(1,1));
    uy=-(MaDreta(1,2)-ColzeDret(1,2));
    uz=-(MaDreta(1,3)-ColzeDret(1,3));

    Ar1=uy;
    Br1=-ux;
    Cr1=0;
    Dr1=-MaDreta(1,2)*Br1-MaDreta(1,1)*Ar1;

    Ar2=uz;
    Br2=0;
    Cr2=-ux;
    Dr2=-MaDreta(1,1)*Ar2-MaDreta(1,3)*Cr2;

    M=[Ar1 Br1 Cr1;
       Ar2 Br2 Cr2;
       Aa Ba Ca];
   d=[-Dr1; -Dr2; -Da];

   if(det(M)~=0)
        e=M\d;
        sA=PuntEnPla(a1,a2,a3,a4,e);
        if (~sA)
            sB=PuntEnPla(b1,b2,b3,b4,e);
```

```
        end
        if(~sA && ~sB)
            sC=PuntEnPla(c1,c2,c3,c4,e);
        end
        if(~sA && ~sB && ~sC)
            sD=PuntEnPla(d1,d2,d3,d4,e);
        end
        if(~sA && ~sB && ~sC && ~sD)
            sE=PuntEnPla(e1,e2,e3,e4,e);
        end
        ABCDE=sA+sB+sC+sD+sE;
    end
    M=[Ar1 Br1 Cr1;
        Ar2 Br2 Cr2;
        Af Bf Cf];
    d=[-Dr1; -Dr2; -Df];
    if(det(M)~=0 && ABCDE==0)
        ef=M\d;
        sF=PuntEnPla(f1,f2,f3,f4,ef);
        if (~sF)
            sG=PuntEnPla(g1,g2,g3,g4,ef);
        end
        if(~sF && ~sG)
            sH=PuntEnPla(h1,h2,h3,h4,ef);
        end
        if(~sF && ~sG && ~sH)
            sI=PuntEnPla(i1,i2,i3,i4,ef);
        end
        FGHI=sF+sG+sH+sI;
    end
    M=[Ar1 Br1 Cr1;
        Ar2 Br2 Cr2;
        Aj Bj Cj];
    d=[-Dr1; -Dr2; -Dj];
    if(det(M)~=0 && ABCDE==0 && FGHI==0)
        ej=M\d;
        sJ=PuntEnPla2(j1,j2,j3,j4,ej);
        if (~sJ)
            sK=PuntEnPla2(k1,k2,k3,k4,ej);
        end
        if(~sJ && ~sK)
            sL=PuntEnPla(l1,l2,l3,l4,ej);
        end
        if(~sJ && ~sK && ~sL)
            sM=PuntEnPla2(m1,m2,m3,m4,ej);
        end
        if(~sJ && ~sK && ~sL && ~sM)
            sN=PuntEnPla2(n1,n2,n3,n4,ej);
        end
        JKLMN=sJ+sK+sL+sM+sN;
    end

    if(sA==1)

contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contA=contA+1; Md1=e;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','m');contJ=0
;contK=0;contL=0;contM=0;contN=0;
```

```
    end
    if(sB==1)

contA=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contB=contB+1;Md1=e;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','r');
    end
    if(sC==1)

contA=0;contB=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contC=contC+1;Md1=e;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','c');
    end
    if(sD==1)

contA=0;contB=0;contC=0;contE=0;contF=0;contG=0;contH=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contD=contD+1;Md1=e;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','g');
    end
    if(sE==1)

contA=0;contB=0;contC=0;contD=0;contF=0;contG=0;contH=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contE=contE+1;Md1=e;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','b');
    end
    if(sF==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contG=0;contH=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contF=contF+1;Md1=ef;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','m');
    end
    if(sG==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contH=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contG=contG+1;Md1=ef;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','c');
    end
    if(sH==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contI=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contH=contH+1;Md1=ef;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','r');
    end
    if(sI==1)
```

```
contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contJ=
0;contK=0;contL=0;contM=0;contN=0;
        contI=contI+1;Md1=ef;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','g');
   end
   if(sJ==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=
0;contK=0;contL=0;contM=0;contN=0;
        contJ=contJ+1;Md1=ej;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','m');
   end
   if(sK==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=
0;contJ=0;contL=0;contM=0;contN=0;
        contK=contK+1;Md1=ej;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','r');
   end
   if(sL==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=
0;contJ=0;contK=0;contM=0;contN=0;
        contL=contL+1;Md1=ej;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','g');
   end
   if(sM==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=
0;contJ=0;contK=0;contL=0;contN=0;
        contM=contM+1;Md1=ej;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','c');
   end
   if(sN==1)

contA=0;contB=0;contC=0;contD=0;contE=0;contF=0;contG=0;contH=0;contI=
0;contJ=0;contK=0;contL=0;contM=0;
        contN=contN+1;Md1=ej;

plot3(Md1(1),Md1(2),Md1(3),'marker','*','markeredgecolor','b');
   end

%    if(ABCDE==1 || FGHI==1 || JKLMN==1)
%    else
%         plot3(e(1),e(2),e(3),'marker','.','markeredgecolor','y');
%         plot3(ef(1),ef(2),ef(3),'marker','.','markeredgecolor','g');
%    end

   Punts=10; Pint=0;
   if(contA>=Punts)
        contA=0
        Pint=1;
        intersection=inta; n=qA;
   end
```

```
if(contB>=Punts)
     contB=0
     Pint=1;
     intersection=intb; n=qA;
end
if(contC>=Punts)
     contC=0; Pint=1;
     intersection=intc; n=qA;
end
if(contD>=Punts)
     contD=0
     Pint=1;
     intersection=intd; n=qA;
end
if(contE>=Punts)
     contE=0
     Pint=1;
     intersection=inte; n=qA;
end
if(contF>=Punts)
     contF=0
     Pint=1;
     intersection=intf; n=qF;
end
if(contG>=Punts)
     contG=0
     Pint=1;
     intersection=intg; n=qF;
end
if(contH>=Punts)
     contH=0
     Pint=1;
     intersection=inth; n=qF;
end
if(contI>=Punts)
     contI=0
     Pint=1;
     intersection=inti; n=qF;
end
if(contJ>=Punts)
     contJ=0
     Pint=1;
     intersection=intj; n=qJ;
end
if(contK>=Punts)
     contK=0
     Pint=1;
     intersection=intk; n=qJ;
end
if(contL>=Punts)
     contL=0
     Pint=1;
     intersection=intl; n=qJ;
end
if(contM>=Punts)
     contM=0
     Pint=1;
     intersection=intm; n=qJ;
end
if(contN>=Punts)
```

```
        contN=0
        Pint=1;
        intersection=intn; n=qJ;
    end

    if (Pint==1)
     RepNorm(intersection,n);
     msg.getPosition().setX(intersection(1));
     msg.getPosition().setY(intersection(2));
     msg.getPosition().setZ(intersection(3));
     msg.getOrientation().setX(n(2));
     msg.getOrientation().setY(n(3));
     msg.getOrientation().setZ(n(4));
     msg.getOrientation().setW(n(1));
     publisher.publish(msg);
    end

end
```

Funció RotTransNormPla.m

```
function [ qPla ] = RotTransNormPla(q,q2)
    Rot=q2tr(q);
    Rot2=q2tr(q2);
    Punt=[1 0 0 0;
          0 1 0 0;
          0 0 1 0;
          0 0 0 1];
      Punt=Rot2*Punt;
      Punt=Rot*Punt;
      qPla=rot2quat(Punt);
end
```

Funció RotTrans.m

```
function NouPunt = RotTrans(Punt,pose,q)
    Rot=q2tr(q);
    Punt=[1 0 0 Punt(1,1);
          0 1 0 Punt(1,2);
          0 0 1 Punt(1,3);
          0 0 0 1];
    Trans=[0 0 0 pose(1,1);
          0 0 0 pose(1,2);
          0 0 0 pose(1,3);
          0 0 0 0];
      Punt=Rot*Punt;
      Punt=Punt+Trans;
      NouPunt=[Punt(1,4) Punt(2,4) Punt(3,4)];
end
```

Funció Rot2quat.m

```
function [b]=rot2quat(R)
threshold = 0.1;
% first, normalized the matrix
```

```
[U,S,V]=svd(R);
R = U*V';

% select the best of the four solution methods
T(1) = 1+R(1,1)+R(2,2)+R(3,3);
T(2) = 1+R(1,1)-R(2,2)-R(3,3);
T(3) = 1-R(1,1)+R(2,2)-R(3,3);
T(4) = 1-R(1,1)-R(2,2)+R(3,3);
[T,ind]=max(T);
switch ind
case  1,  % b1 solution
    b1(1,1)    = 0.5*sqrt(T);
    b1(2,1)    = (R(3,2)-R(2,3))/4/b1(1);          % checked
    b1(3,1)    = (R(1,3)-R(3,1))/4/b1(1);          % checked
    b1(4,1)    = (R(2,1)-R(1,2))/4/b1(1);          % checked
    b       = b1/norm(b1);    % renormalize
case  2,  % b2 solution
    b(2,1)    = 0.5*sqrt(T);
    b(1,1)    = (R(3,2)-R(2,3))/4/b(2);          % checked
    b(3,1)    = (R(1,2)+R(2,1))/4/b(2);          % checked
    b(4,1)    = (R(3,1)+R(1,3))/4/b(2);          % checked
    b       = b/norm(b);    % renormalize
    %R2 = quat2R(b)
case  3,  % b3 solution
    b(3,1)    = 0.5*sqrt(T);
    b(1,1)    = (R(1,3)-R(3,1))/4/b(3);          % checked
    b(2,1)    = (R(1,2)+R(2,1))/4/b(3);          % checked
    b(4,1)    = (R(2,3)+R(3,2))/4/b(3);          % checked
    b      = b/norm(b);    % renormalize
case  4,  % b4 solution
    b(4,1)    = 0.5*sqrt(T);
    b(1,1)    = (R(2,1)-R(1,2))/4/b(4);          % checked
    b(2,1)    = (R(1,3)+R(3,1))/4/b(4);          % checked
    b(3,1)    = (R(2,3)+R(3,2))/4/b(4);          % checked
    b       = b/norm(b);    % renormalize
otherwise
    error('Error in Rot 2 Quat')
end
```

Funció RepNorm.m

```
function RepNorm(intb,RotA )
v1=[0.01 0 0];
v2=[0 0.01 0];
v3=[0 0 0.1];
v1=RotTrans(v1,intb,RotA);
v2=RotTrans(v2,intb,RotA);
v3=RotTrans(v3,intb,RotA);
line([v1(1) intb(1)],[v1(2) intb(2)],[v1(3) intb(3)],'Color','r');
line([v2(1) intb(1)],[v2(2) intb(2)],[v2(3) intb(3)],'Color','g');
line([v3(1) intb(1)],[v3(2) intb(2)],[v3(3) intb(3)],'Color','b');
end
```

Funció q2tr.m

```
function t = q2tr(q)

    q = double(q);
    s = q(1);
    x = q(2);
    y = q(3);
    z = q(4);

    r = [1-2*(y^2+z^2) 2*(x*y-s*z) 2*(x*z+s*y);
        2*(x*y+s*z) 1-2*(x^2+z^2) 2*(y*z-s*x);
        2*(x*z-s*y) 2*(y*z+s*x) 1-2*(x^2+y^2)];
    t = eye(4,4);
    t(1:3,1:3) = r;
    t(4,4) = 1;
end
```

Funció PuntEnPla.m

```
function s=PuntEnPla(P1,P2,P3,P4,e)

        Sentit1=(P2(1)-P1(1))*(P4(2)-P1(2))-(P2(2)-P1(2))*(P4(1)-
P1(1));
        Sentit23e=(P2(1)-e(1))*(P4(2)-e(2))-(P2(2)-e(2))*(P4(1)-e(1));
        Sentit31e=(P4(1)-e(1))*(P1(2)-e(2))-(P4(2)-e(2))*(P1(1)-e(1));
        Sentit12e=(P1(1)-e(1))*(P2(2)-e(2))-(P1(2)-e(2))*(P2(1)-e(1));

        Sentit2=(P2(1)-P3(1))*(P4(2)-P3(2))-(P2(2)-P3(2))*(P4(1)-
P3(1));
        Sentit25e=(P2(1)-e(1))*(P4(2)-e(2))-(P2(2)-e(2))*(P4(1)-e(1));
        Sentit53e=(P4(1)-e(1))*(P3(2)-e(2))-(P4(2)-e(2))*(P3(1)-e(1));
        Sentit32e=(P3(1)-e(1))*(P2(2)-e(2))-(P3(2)-e(2))*(P2(1)-e(1));

        if((Sentit1<=0 && Sentit23e<=0 && Sentit31e<=0 &&
Sentit12e<=0)||(Sentit1>=0 && Sentit23e>=0 && Sentit31e>=0 &&
Sentit12e>=0)||(Sentit2<=0 && Sentit25e<=0 && Sentit53e<=0 &&
Sentit32e<=0)||(Sentit2>=0 && Sentit25e>=0 && Sentit53e>=0 &&
Sentit32e>=0))
           s=1;
        else
            s=0;
        end
end
```