

Manos robóticas para la manipulación de objetos punzo-cortantes

Alejandro Hernández De la Torre
Ing. en Robótica y Sistemas Digitales
Instituto Tecnológico y de Estudios
Superiores de Monterrey
Monterrey, México
A01651516@tec.mx

Abstract—This paper researches how could a prosthetic hand be implemented with an XArm robot in order to make a dexterous robot that could handle dangerous object like but not limited to knives and sharp objects to facilitate the everyday tasks that humans may find dangerous or tedious.

Palabras clave—Mano, Motor, Control, Humana, Ros, Deztresa, Ubuntu, ARM

I. INTRODUCCIÓN

Hoy en día la robótica esta ganando popularidad en varios sectores de la sociedad impactando incluso en la cultura, esta popularidad aumento más con la pandemia del *COVID -19* ya que el mundo vio que los robots podían hacer tareas sin exponer la integridad del ser humano [1]. La industria cada día pide más avances para aumentar su productividad así como sus ganancias y también reducir costos. Un sector el cual empieza a evolucionar es la robótica de servicio, esta rama se dedica a buscar un uso para los robots en un entorno casero, ya sea para facilitar su vida o hacer tareas peligrosas por el humano. Este es el punto el cual este reporte de investigación se va a enfocar.

II. CONTEXTO GENERAL Y JUSTIFICACIÓN

Muchas veces se cree que los accidentes en casa deben ser los menos usuales cuando es exactamente al revés, las personas se descuidan más en sus propios hogares. La herramienta casera que produce más heridas deshabilitantes es el cuchillo, esto fue comprobado por el investigador Gary A Smith. El cual comprobó con un 95% de exactitud que más de 8 millones de personas fueran tratadas en urgencias desde 1990 hasta 2008 por heridas con cuchillos [2]. Eso ha generado una necesidad de manejar de una manera más segura y eficiente este tipo de objetos. Aprovechando el aumento en la popularidad de los robots y su eficiencia de poder hacer tareas que el humano pudiera encontrar austeras o peligrosas surgió la problemática a resolver de esta investigación

III. DELIMITACIÓN DEL OBJETO DE ESTUDIO

En la planeación del proyecto se espera que el robot pueda en primera estancia, mapear el espacio de trabajo para que pueda identificar los objetos de interés como el cuchillos o las herramientas capaces de poder hacer algún daño a un ser humano. Despues de la identificación de estos se planea que un brazo robótico pueda acercarse y con una mano parecida a la estructura osea de una mano humana pueda manipular la herramienta de una forma apropiada (cortar algo con un cuchillo), o en su defecto

alejarlo de una persona cuando se encuentre muy cerca, con la intención de salvaguardar la integridad de la persona la cual es la tarea principal del robot.

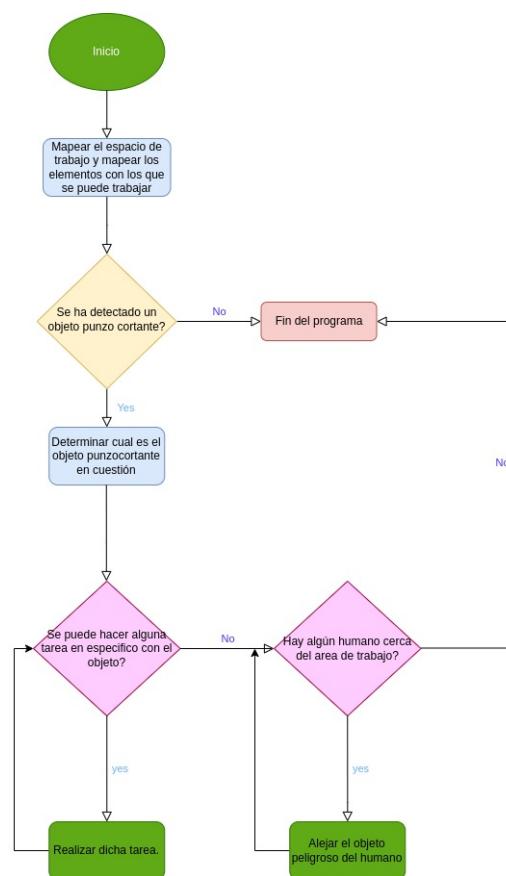


Fig. 1. Diagrama de flujo del funcionamiento del "robot"

IV. PLANTEAMIENTO DEL PROBLEMA

Se plantea que en esta situación algún individuo pueda estar expuesto a un objeto punzo cortante en una superficie, en este caso el robot detectara que el susodicho objeto esta al alcance de dicho individuo por lo cual se espera que el robot contenga el objeto punzo cortante en cuestión y lo aleje, esto con la intención de prevenir algún siniestro. Un ejemplo claro sería como en una mesa se encuentra un cuchillo al alcance de un infante y este pudiera tomarlo para empezar a realizar actividades que pudieran llevar a un siniestro.

V. MARCO TEÓRICO

A. Estructura osea de la mano humana

La mano humana posee 54 huesos, entre estos los que nos importan son aquellos que están en los dedos de la mano humana, la estructura osea de los dedos se repite en 4 de estos, los cuales son: *meñique* (5), *anular* (4), *dedo medio* (3) e *índice* (2), estos dedos se pueden agrupar como dígitos, esta estructura varía en el dedo *pulgar* (1) [3].

1) *Estructura del dedo humano (2-5)*: Yendo en orden desde las puntas de los dedos hasta el origen de estos los dedos se distribuyen de esta manera:

- **Falange distal**
- **Articulación interfalángica distal (DIP)**
- **Falange proximal**
- **Articulación interfalángica proximal (PIP)**
- **Metacarpio**
- **Articulación metacarpofalángica (MCP)**

2) *Estructura del pulgar (1)*: Yendo en orden desde las puntas del pulgar hasta el origen se distribuye de esta manera:

- **Falange distal**
- **Falange proximal**
- **Articulación interfalángica proximal (PIP)**
- **Metacarpio**
- **Articulación metacarpofalángica (MCP)**

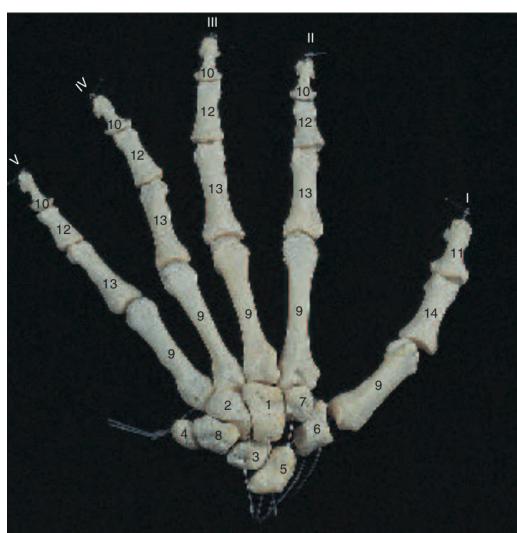


Fig. 2. Esquema de la estructura osea de la mano [3]

B. Definición de objetos punzo-cortantes

1) *Cuchillos*: La definición de cuchillos puede variar dependiendo del contexto. La RAE define estos utensilios como un instrumento para cortar formado por una hoja de metal de un corte y con mango [4]. En el sector de leyes, específicamente en Canadá, en la cláusula No.6872 define los cuchillos como: *cualquier daga, herramienta de mano con una hoja de metal o cualquier objeto diseñado, re-diseñado o tratado con la capacidad de propiciar heridas serias o la muerte de cualquier persona mediante el corte o*

empuñalamiento, estas herramientas deben de poseer una hoja de metal mínimamente de 7cm para ser consideradas como cuchillos [5].

2) *Herramientas y objetos punzo-cortantes*: En el ámbito médico la organización *SafeNeedleDisposal* define a los objetos punzo-cortantes como: *dispositivos con puntas o bordes afilados que pueden usarse en el hogar, en el trabajo o en viajes para controlar los trastornos médicos de las personas o de sus mascotas* [6]

3) *Consideración de utensilios punzo-cortantes para este trabajo*: Después de estudiar y pensar en que manera sería eficiente considerar una herramienta como peligrosa o punzo-cortante. He llegado a la conclusión que cualquier objeto con una hoja metálica mayor a los 5 cm con la capacidad de propiciar heridas o cortes profundos a la piel humana como un objeto a tomar cuenta para el algoritmo necesario para el robot a diseñar.



Fig. 3. Cuchillo de 11 cm

En la Fig. 3 se puede apreciar el cuchillo que se usará en la mayoría de la experimentación de este proyecto.

VI. OBJETIVOS

El funcionamiento óptimo mentalizado para este robot, estima que en una mesa la cual sera el espacio de trabajo. Una cámara estéreo en este caso la *Zed* en conjunto con la cámara del robot realice un mapeo de la zona de trabajo y encontrar los objetos necesarios con ayuda de *YOLO* y *Point Cloud Library*. Después de tener el mapa listo, con ayuda del *X-ARM* con la *Phoenix Hand V3* integrada tomara el objeto de interés y lo sostendrá de una forma específica y si detecta alguna persona a los alrededores, buscara alejar dicho objeto de esta persona.

VII. HIPÓTESIS

Si el robot es capaz de llevar el objeto punzocortante a otro lado sin provocar daños a un tercero y conteniéndolo como es debido, este proyecto de investigación esta rumbo a ser exitoso.

VIII. METODOLOGÍA Y PROPUESTA METODOLÓGICA A UTILIZAR

Para poder medir los avances obtenidos a lo largo de la investigación se favoreció un diagrama de ramas que dividiera las diferentes áreas del proyecto las cuales son:

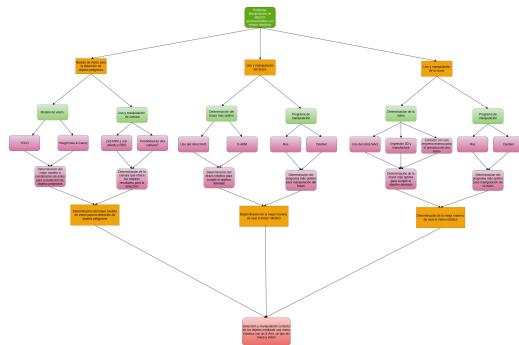


Fig. 4. Metodología

- Visión y percepción
- Manipulación del brazo
- Manipulación de la mano

De ahora en adelante siempre se hablará específicamente de cada área del proyecto para facilitar la documentación de este.

IX. TÉCNICAS Y HERRAMIENTAS DE INGENIERÍA EMPLEADAS

Para poder llevar acabo este proyecto se utilizaron las siguientes técnicas :

- Inteligencia artificial
- Aprendizaje automatizado
- Programación Orientada a objetos
- Visión computacional
- Impresión 3D
- Diseño de mecanismos
- Sistemas embebidos
- Diseño de circuitos
- Nube de puntos
- Mapeo de entornos

X. INFRAESTRUCTURA

En la siguiente imagen se puede apreciar como se comporta toda nuestra arquitectura y como se mandan mensajes entre los diferentes nodos de nuestro programa.

- El nodo que inicia con *image changer* es el encargado de analizar las imágenes para detectar si se trata de un objeto peligroso para después realizar un análisis con *OpenCV*
- Posteriormente el nodo que marca *zed2* es el de la cámara con el mismo nombre, esta publica la imagen y todo lo relacionado a visión como puede ser la nube de puntos.
- Más abajo se puede llegar a visualizar el nodo *xarm* que se encarga del movimiento del brazo. Finalmente

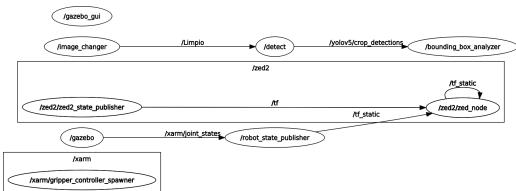


Fig. 5. Infraestructura del proyecto en *graph*

XI. RECURSOS UTILIZADOS

En esta sección se definirán cuales son los componentes planeados para utilizar en el proyecto:

A. Componentes Fisicos:

1) *Zed Camera 2*: La *Zed* es una cámara estéreo manufacturada por *Stereo Labs* con la intención de ser utilizada en percepción de espacios y profundidades. Incluye una *IMU*, *Barómetro*, *Acelerómetro*, *Giroscopio* y *Magnetómetro*. Esto le permite tener datos como la elevación, inercia y posición, sumamente útiles en el mapeo de los datos. Posee un ángulo de captura horizontal de 120°. La cámara mide 175 mm de largo, 30 mm de alto y 33 mm de ancho. Pesa 166 gramos, se alimenta con 380mA y 5V los cuales le son proporcionados mediante un cable USB [7].



Fig. 6. Camara Zed 2 [7]

2) *Brazo X-Arm 5*: El *X-Arm* es un brazo desarrollado por *UFactory*. Este brazo puede cargar hasta 5 kgs. Posee 6 grados de libertad, tiene un alcance de 700 mm y pesa 12.2 KGs. Tiene una *SDK open source* el cual se utilizará ampliamente en el proyecto. Se alimenta con 24V DC y 16.5 A. Esta hecho con aluminio y fibra de carbono. Tiene una base en la ultima extensión de su brazo lo cual le permite adaptarse con *grippers* y demás sensores, en este caso se utilizará este componente principalmente para la adaptación de la mano [8]



Fig. 7. X-ARM 5 [8]

3) *Phoenix Hand v3*: Esta mano creada por la empresa *e-NABLE* y *Team Unlimbited* es *open-source* y se usa principalmente para la rehabilitación de personas que perdieron o nacieron sin alguna de sus extremidades. Esta mano debe ser impresionada y se instala en el brazo utilizando un sistema de fijación.

da acceso a los *STL* y *CAD FILES* [9] necesarios para este proceso. Funciona mediante hilos los cuales hacen que con un movimiento simple de fuerza se pueda abrir y cerrar. También para su funcionamiento correcto se necesita ligas dentales [10]. El tamaño de la mano puede variar dependiendo de la escala que se le otorgue cuando se le imprima. Para este proyecto se planea que se asimile al tamaño de una mano adulta.



Fig. 8. Phoenix Hand v3 [10]

B. Software

1) *Ubuntu 18.04 LTS (Bionic Beaver)*: El sistema operativo a utilizar para el desarrollo de este proyecto sera *Ubuntu 18.04*, ya que su naturaleza *open source* [11] facilita la implementación de los diferentes software y *SDK* necesarias para la implementación del proyecto y sus diferentes componentes.



Fig. 9. Logo Ubuntu

2) *ROS Melodic*: Se optó por utilizar *Robotic Operating System* en su versión *Melodic Morenia* como la versión a utilizar ya que era sumamente compatible con las librerías necesarias para poder ejecutar todos los componentes necesarios en el trabajo y ya que esta versión del software fue diseñada en mente para la versión de Ubuntu que se describió anteriormente. Otro factor para su elección fue la compatibilidad con la arquitectura *arm64* [12]. También las opciones que nos permite utilizar como *Rviz*, *rqt_imageview*, *Gazebo*.



Fig. 10. ROS Melodic [12]

3) *Visual Studio Code*: Se utilizará VSC como el editor principal de código, ya que su capacidad para adaptarse a diferentes lenguajes de programación como puede ser *Python*, *C++*, etc. Será indispensable para lograr este objetivo en mente, también facilitará el almacenamiento de archivos mediante la plataforma *Github*. También sus extensiones hacen más cómodo el debuggeo y afinamiento de problemas [13].

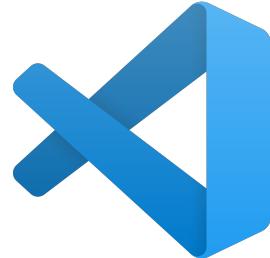


Fig. 11. Visual Studio Code [13]

C. Librerías y compiladores

NOTA: CABE DESTACAR QUE ESTAS LIBRERÍAS PUEDEN CAMBIAR DEPENDIENDO DEL PROGRESO DEL PROYECTO Y EL DESARROLLO DE ESTE PERO SE PLANEÁ UTILIZAR TODAS LAS LIBRERÍAS PLANTEADAS AQUÍ:

- **YOLO V5 "v6.2"**

Esta librería es una familia de arquitecturas de detección de objetos y modelos pre-entrenado con la librería de datos COCO. Es una investigación hecha por la empresa Ultralytics [14]. Se planea utilizar esta librería para la detección de los elementos punzocortantes y personas.

- **SDK - Stereo Labs**

Librería necesaria para utilizar los componentes de la cámara ZED 2 con Python y ROS [7].

- **Open CV 2.4**

Es una librería open source que incluye miles de algoritmos de visión computarizada para poder procesar imágenes y por ende videos para la realización de este proyecto [15].

- **CUDA Toolkit 11.7**

Técnicamente siendo un compilador CUDA, es un compilador desarrollado por la empresa Nvidia para

el desarrollo, optimización de aplicaciones en sistemas que cuenten con una tarjeta gráfica de esta marca. Indispensable para la cantidad de procesamiento de imagen y algoritmos a realizar. [16]

- **Rviz, Move it, Gazebo**

Librerías utilizadas para la visualización de imágenes, modelos 3D y manipulación de estos, vienen incluidas en las instalación de ROS [12], ya que son herramientas indispensables de este software

- **SDK - X-Arm**

Librerías utilizadas para el correcto funcionamiento del X-ARM en simulación e implementación real, altamente utilizable en Rviz y Moveit [8]

- **Point Cloud Library** El objetivo de esta librería es utilizar y manipular las Point-Cloud que podemos extraer de la ZED, aparte esta librería cuenta con detección de características como SIFT o diferentes algoritmos como detección de personas, objetos y mapeo de entornos así como de objetos. [17]

XII. RESULTADOS

A. Visión y Percepción

Como primer avance de este rubro se tuvo que instalar las dependencias para que funcionara correctamente la cámara ZED2 en Ubuntu18.04. Para poder lograr este objetivo se tuvo que instalar el Compilador CUDA 11.7, este avance fue logrado en medio día de trabajo. Al finalizar se pudo visualizar los diferentes nodos y propiedades en rqt image view y RViz.

Como segundo avance se buscó implementar las librerías de YOLOV3, con un resultado mediocre en el rendimiento del algoritmo por lo cual se empezó a optar por un algoritmo más refinado o que tuviera un mejor rendimiento el cual pudiera obtener mejores resultados en la detección de imágenes y rendimiento de la imagen, hablando claramente de los FPS. Por esto mismo se optó por el algoritmo descrito en el apartado III, el cual fue YOLOV5. Este algoritmo obtuvo resultados aceptables tanto en la cámara local de la computadora como con la cámara ZED 2. Los cuales se mostraran a continuación, cabe destacar que todos los resultados son desde la ZED 2 y corriendo en video.



Fig. 12. Resultados de detección de personas con YoloV5

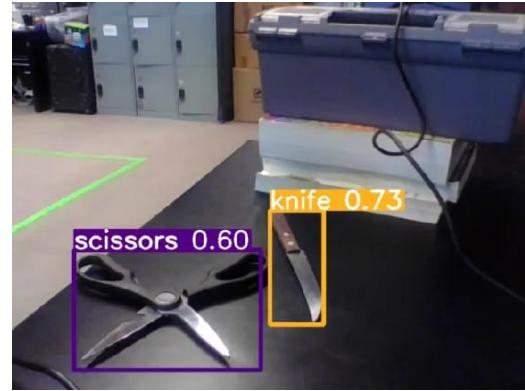


Fig. 13. Detección de objetos punzocortantes con YoloV5

Por otro lado con la misma cámara Zed 2 se implemento un algoritmo para poder encontrar los bordes de los objetos, esto con ayuda de Python y la librería Open CV. Se uso en específico las funciones Canny y Hough Lines. Los resultados en este caso fueron satisfactorios hasta cierto grado.



Fig. 14. Detección de orillas de un cuchillo

Después en otro intento para detectar el área metálica del cuchillo en comparación del área del mango o el área de manipulación, se intento detección de colores con resultados poco satisfactorios, por lo cual se descarto esta opción y se optará por un método de votación de lineas en la imagen el cual sera discutido más adelante

Se pudo empezar a encontrar las siluetas de humanos con la Camara ZED 2, esto usando sus propiedades de Cloud Detection las cuales con algoritmos nos dan esta filtrada con opciones específicas dependiendo del usuario, en este caso se usarón las siguientes opciones en el programa Rviz para lograr los resultados mostrados a continuación, estos resultados fueron obtenidos a 2 metros de la cámara.

- Style: Flat Squares
- Color Transformer : Axis Color
- Queue Size : 10

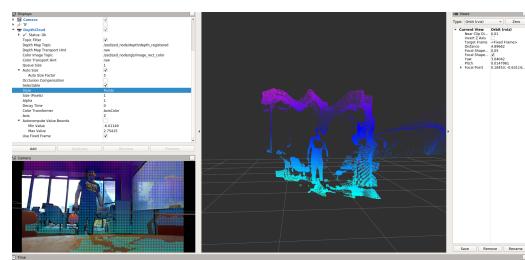


Fig. 15. Detección de contornos de un humano

También se logró la detección de profundidades, este resultado fue visto desde la Zed 2 en rqt image view

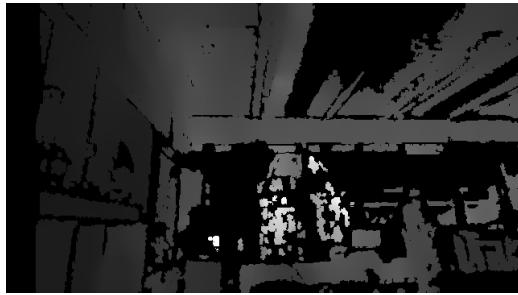


Fig. 16. Detección de la profundidad

Se empezó a mapear en 3d algunos entornos, con las opciones antes descritas, variando la configuración de Color Transformer a RGB8, también se tuvo que subir el Decay Time de los PointCloud para poder visualizar mejor el mapa en sí. También se empezó a implementar la IMU de la cámara para poder ver el trayecto que realizó la cámara, este se puede apreciar de color rosa. Para poder grabar estos datos se uso una Rosbag guardando los topics necesarios

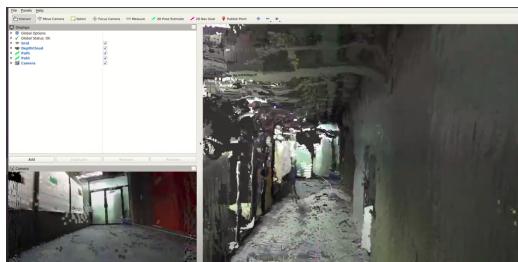


Fig. 17. Mapeo de pasillo en 3D

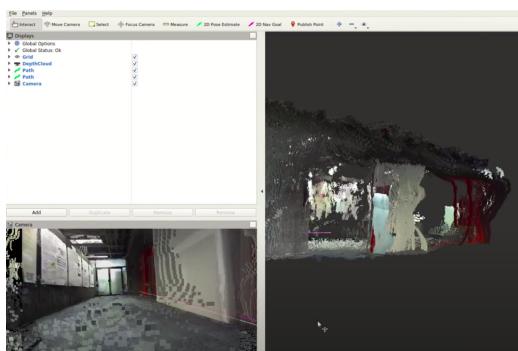


Fig. 18. Mapeo de pasillo en 3D visto desde afuera

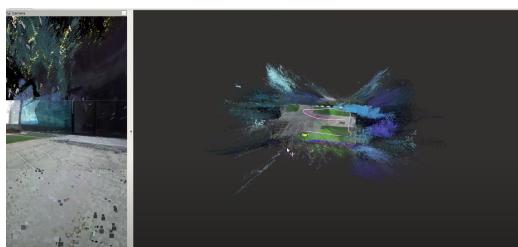


Fig. 19. Mapeo de jardín con pelotas de fondo

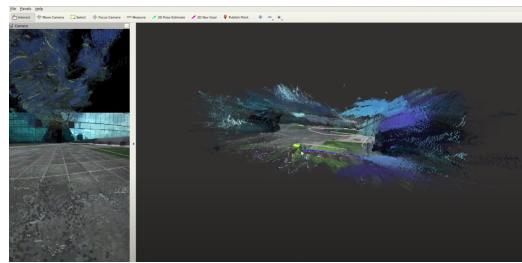


Fig. 20. Mapeo de jardín visto horizontalmente

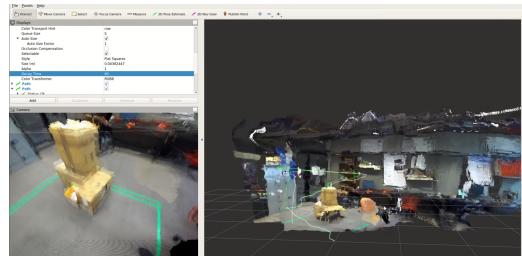


Fig. 21. Mapeo de laboratorio con objetos alrededor

Estos resultados muestran un buen camino de como en el futuro se podrían empezar a combinar estos algoritmos para lograr la detección de objetos en un mapa.

En el rubro de visión fue necesario empezar a conseguir maneras de obtener las características para el cuchillo por lo cual se empezó a obtener resultados de diferentes algoritmos como:

- ORB
- SIFT
- Shi-Tomashi
- Fast

Con resultados variados los cuales se muestran a continuación:

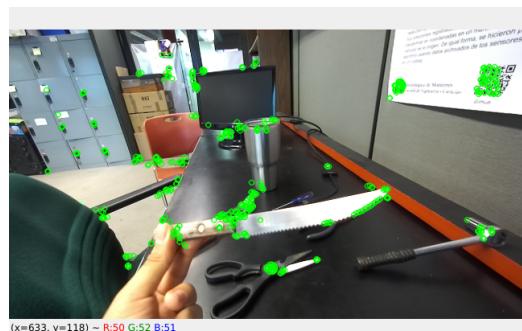


Fig. 22. Cuchillo con ORB

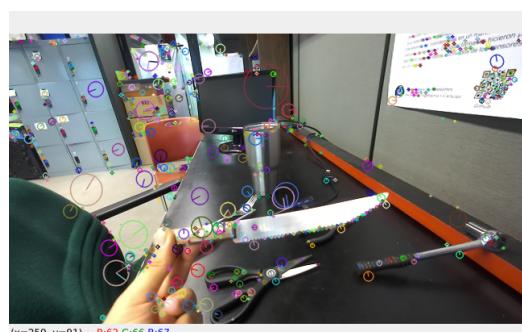


Fig. 23. Cuchillo con SIFT

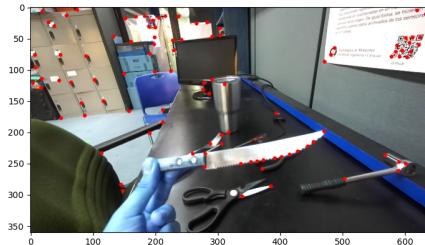


Fig. 24. Cuchillo con Shi-Tomashi



Fig. 25. Cuchillo con Fast

En las imágenes podemos ver que ORB, detecta relativamente bien las orillas del cuchillo pero no de una manera completa. SIFT las detecta de una manera idónea pero tristemente su uso esta patentado por lo cual se debe de pagar para usarlo. Shi-Tomashi podría funcionar pero tiene algunas separaciones entre los puntos detectados. En ultimo lugar Fast detecta de una manera idónea el cuchillo como tal y su uso esta especializado a ser un algoritmo ligero para aplicaciones de SLAM. Por lo cual este sera el algoritmo que se usara para identificación de la hoja del cuchillo. Independientemente se seguirá haciendo pruebas con todos los algoritmos menos SIFT, para verdaderamente comprobar en el modelo cual funciona mejor pero se le dará preferencia a FAST.

En el rubro de percepción, fue obvio que los datos que se obtuvieron con el tópico Point Cloud 2 son sumamente útiles para el proyecto. Por lo cual se empezó a buscar maneras de utilizar este dato. Después de entender como funcionan los Point Cloud, se determino que la librería Point Cloud Library sería el paso siguiente para el curso de la investigación. Investigando se encontró un github el cual tiene muchos ejemplos de percepción llamado ros-perception [18] y tambien otro llamado pcl-ros-tutorial [19]. Se ha logrado compilar todos los ejemplos hasta el apartado 5 obteniendo mejores resultados en el mapeo de puntos, entre ellos a destacar las funciones que dieron mejores resultados es el filtro Voxel Grid y el mapeo con Octomap, ya que la Point Cloud mostrada en Rviz genero resultados más claros. Se necesita indagar más en esos procedimientos pero se comprobó que funcionan correctamente en simulación. Tambien se busca implementar el

algoritmo SIFT para nuestros datos ya que viene incluida en PCL. Tambien este librería cuenta con percepción de objetos y mapeo de estos lo cual podría servir de manera grandiosa en el futuro.

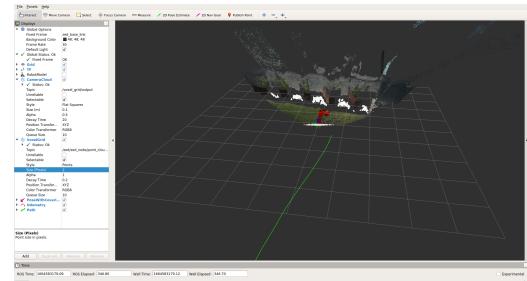


Fig. 26. 1er intento de mapeo con VoxelGrid

Se logro la identificación de las superficies metálicas para poder identificar donde es el lugar en el que una herramienta posee una hoja la cual puede llegar a ser peligrosa para un ser humano. Para lograr este resultado se realizaron estas transformaciones de la imagen con el software OpenCV:

- Se transforma la imagen de formato de color BGR a HSV
- Se genera una capa para que la imagen solo detecte el rango de color en el que se estima se encuentra la hoja de un cuchillo o alguna herramienta el cual tiene unos rangos en el límite inferior del rango HSV (0,0,180) y superior de (255,60,255).
- Despues se realiza una ecualización de nuestra imagen utilizando CLAHE
- Posteriormente se invierte la imagen, para que la parte interna del cuchillo sea resaltada de una manera precisa.
- Se dilata y erosiona la imagen para evitar falsos verdaderos.
- Se genera un límite binario en nuestra imagen para facilitar la identificación del área deseada con la función threshold
- Se encuentra el contorno del cuchillo y se dibuja en rojo.



Fig. 27. Detección de areas peligrosas en cuchillos

Finalmente para encontrar la mano se realizaron los siguientes puntos:

- Se saturo la imagen original
- Se realizo un difuminado con la herramienta Gaussian Blur

- Posteriormente se implemento límite binario y se dibujaron los contornos de este en color azul

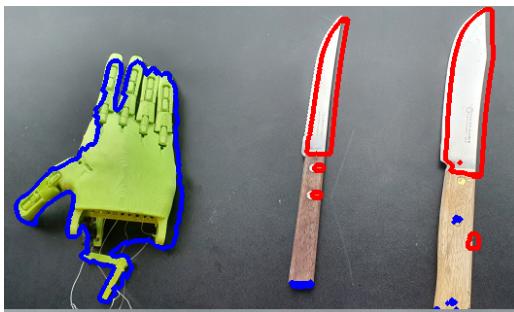


Fig. 28. Resultado final con detección de mano y cuchillos

B. Manipulación de la mano

En este rubro se intentaron varias alternativas antes de encontrar la mejor, entre las ideas descartadas podemos encontrar:

- Manipulación del brazo y mano del robot NAO
- Manipulación de una mano comercial manufacturada por una empresa externa
- Gripper de 3 dedos

Al final se decidió desechar estas 3 ideas ya que no fueron las más óptimas por su poca compatibilidad con nodos de ROS, en el caso del robot Nao, la poca cooperación de las empresas en el caso de la segunda opción, y la 3era opción fue descartada por su limitada operación.

Finalmente se decidió por el uso del Phoenix Hand V3 por su simple pero amplio sistema de hilos el cual permite la robotización de esta mano. Se empezó a experimentar con una mano prototipo para empezar a probar su funcionalidad para sostener objetos, mostrando resultados óptimos ya que la susodicha es capaz de sostener objetos punzocortantes.

Se empezó a realizar el prototipo con motores a pasos la capacidad de esta mano para ser automatizada, lo cual mostró resultados favorables, por el momento solo se ha logrado automatizar un dedo, el cual se muestra a continuación



Fig. 29. Prototipo de motorización de la mano

Viendo el resultado favorable que se obtuvo con esta mano, se imprimirá una de mayor tamaño capaz de sostener objetos más grandes, la cual también sera una versión actualizada de esta misma para obtener mejores resultados. También se mejorara el material con el que se imprimirá.

Se imprimió la mano de un mejor material y de un tamaño un poco superior, esta nueva mano mostro una mejora en la fuerza de agarre. Pero falta implementarle algo en las yemas de los dedos para tener un mejor agarre se esta explorando la posibilidad de usar cepillo de dientes para bebe o gomas de pasa-libros. Por otro lado ya se logro la conexión del Arduino MEGA 2560 con ROS para poder manipular los dedos desde ROS. El paso siguiente es empezar a diseñar la base 3d que iría al brazo X-ARM esto con la intención de poder lograr un brazo que pudiera agarrar eficientemente con una mano asimilada a la humana.



Fig. 30. Phoenix Hand V3 funcionando

Finalmente la mano pudo ser usada como debía, al final se optó por mecanizar la mano con 3 motores los cuales corresponden cada uno a diferentes partes de la mano los cuales son :

- Pulgar
- Índice y medio
- Angular y meñique

Estos motores se controlan con dos puentes H y un microcontrolador, por falta de recursos se tuvo que optar por la utilización de un Arduino MEGA 2560. Para la base se imprimió en PCL una caja con medidas de 17.5 cm de alto, 12 cm de ancho y 6.5 cm de profundo. A continuación se muestra la renderización en la plataforma SolidWorks

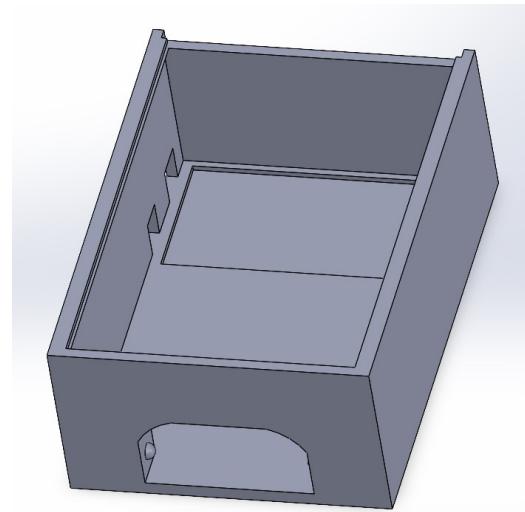


Fig. 31. Caja de la mano en SolidWorks

Para los circuitos eléctricos de la mano se utilizaron los siguientes componentes:

- 2 - Controlador de Motores L298n
 - 3 - Motorreductor recto
 - 1 - Arduino Mega 2560
 - 1 - Pila de 9v recargable
 - Jumpers M-H y M-M

En el siguiente esquema se muestra el conectado en la plataforma Proteus, cabe destacar que el archivo correspondiente se incluirá en el anexo

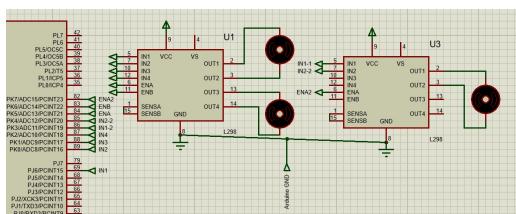


Fig. 32. Cableado de mano en el software Proteus

En la siguiente figura se muestra la caja y el cableado completos, también la antes mencionada acoplada con la mano.



Fig. 33. Producto Final de la caja con la mano

Se realizaron algunas pruebas de la fuerza y la capacidad en la que la mano podría tomar algunos objetos, así como su manera de gesticular.

NOTA: ESTAS PRUEBAS FUERON REALIZADAS EN UN ENTORNO SEGURO, CON LA MANO INMOVILIZADA DONDE NADIE PUDIERA SALIR POTENCIALMENTE DAÑADO. AUNQUE EL OBJETO NO FUE NEUTRALIZADO CORRECTAMENTE COMO LO MARCA EL ESTÁNDAR ANSI/RIA R15.06- [20], ESTAS PRUEBAS FUERON MERAMENTE DEMOSTRATIVAS DE LA FUERZA DE LA MANO, POSTERIORMENTE SE DEMOSTRARÁ COMO SE DEBERÍA DE TRATAR CORRECTAMENTE CON ESTOS OBJETOS.

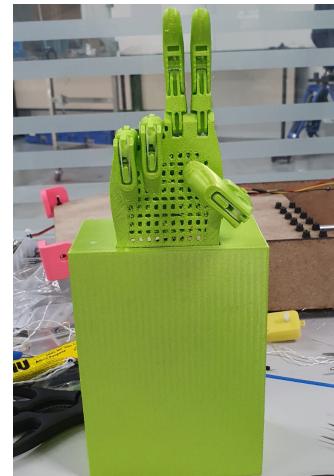


Fig. 34. Mano gesticulando

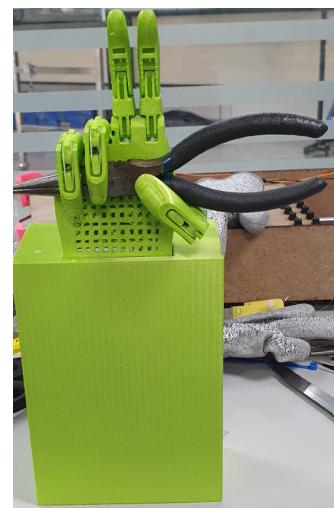


Fig. 35. Mano tomando unas pinzas

C. Manipulación del brazo

En este rubro se logró instalar las librerías necesarias para lograr simular el robot y manipularlo en RViz así como de manera real. En el siguiente reporte se podrá reportar el funcionamiento en físico de este robot. No fue posible este parcial por la falta de autorización de uso en el laboratorio

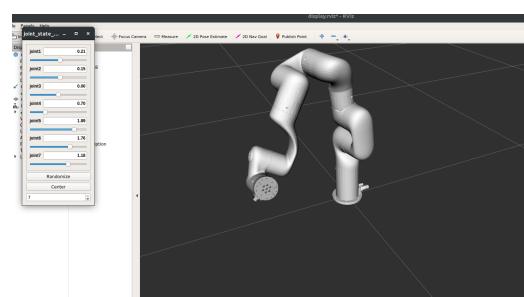


Fig. 36. X-ARM en simulación

Fue posible empezar a programar rutinas con el brazo en Rviz esto con la finalidad de empezar a declarar rutinas necesarias para el correcto movimiento de este. También se logró empezar a programar rutinas específicas con MoveIt!

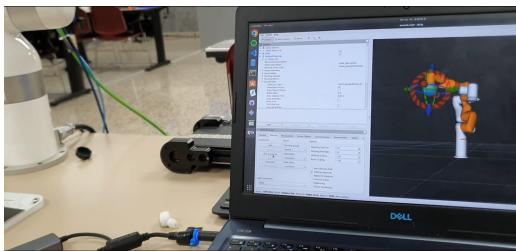


Fig. 37. Rviz funcionando con el Robot



Fig. 40. X-ARM sosteniendo una pinza



Fig. 38. X-ARM en la posición indicada

Se combino la caja que contiene la mano con el brazo para poder generar una pinza. Para poder probar esto, se tuvo que perforar 4 hoyos en la caja para poder ajustar tornillos y evitar dañar el prototipo. En la figura siguiente se puede observar como quedo este prototipo

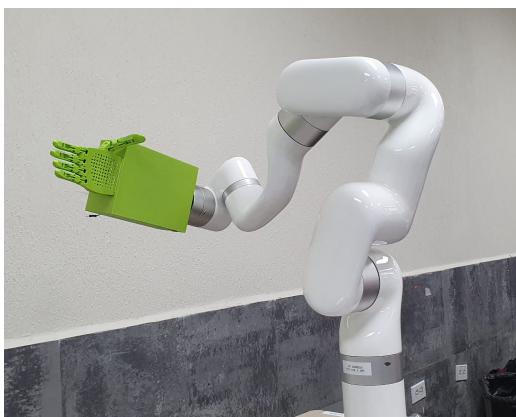


Fig. 39. X-ARM con mano robótica

Después se empezó a probar el funcionamiento de la mano con el brazo sosteniendo diferentes objetos peligrosos.

NOTA: ESTAS PRUEBAS FUERON REALIZADAS EN UN ENTORNO SEGURO, CON LA VELOCIDAD DEL X-ARM LIMITADA AL MÍNIMO DONDE NADIE PUDIERA SALIR POTENCIALMENTE DAÑADO. AUNQUE EL OBJETO NO FUE NEUTRALIZADO CORRECTAMENTE COMO LO MARCA EL ESTÁNDAR ANSI/RIA R15.06-2012 [20], ESTAS PRUEBAS FUERON MERAMENTE DEMOSTRATIVAS DE LA FUERZA DE LA MANO, POSTERIORMENTE SE DEMOSTRARÁ COMO SE DEBERÍA DE TRATAR CORRECTAMENTE CON ESTOS OBJETOS.

XIII. CONCLUSIONES

El proyecto se encuentra en sus etapas iniciales de desarrollo, a futuro se busca lograr resultados más robustos entre los cuales se encuentran pero no están limitados a:

- Lograr un mejor reconocimiento de los objetos peligrosos
- Mapear con técnicas más precisas en el software PCL
- Lograr encontrar una manera específica de manipular los objetos
- Mecanizar cada dedo específicamente de la mano
- Lograr un diseño más ergonómico de la mano
- Implementar la mano en otros modelos de brazos
- Implementar autonomía en la mano del brazo para usarse en diferentes contextos y aplicaciones
- Utilizar un microcontrolador más especializado como alguno de la familia SMT-32 o algún otro como un ATMEGA-16
- Encontrar una manera específica de manipular los objetos peligrosos sin violar los términos de seguridad expresadas en algunos documentos como ANSI/RIA

Este proyecto con algunas de estas características a futuro se plantea utilizar en un entorno donde la interacción humano-robot sea más cotidiana y normalizada. Esto para salvaguardar la vida de las personas que pudieran encontrarse en algún peligro.

XIV. AGRADECIMIENTOS

Se agradece la supervisión y apoyo del Dr. Luis Alberto Muñoz Ubando. Por otro lado se agradece al Ing. Juan de Dios Oseas Martínez por el apoyo con las impresiones 3D, al grupo estudiantil Vantec y al Instituto Tecnológico y de Estudios Superiores de Monterrey por proveer con los materiales e instrumentación necesaria para llevar a cabo la investigación.

REFERENCIAS

- [1] *The Rise of Service Robots in the Hospitality Industry: Some Actionable Insights* | Boston Hospitality Review. (2021, October 4). © 2022 Boston University. Retrieved September 9, 2022, from <https://www.bu.edu/bhr/2021/10/04/the-rise-of-service-robots-in-the-hospitality-industry-some-actionable-insights/>
- [2] Smith G. A. (2013). Knife-related injuries treated in United States emergency departments, 1990-2008. *The Journal of emergency medicine*, 45(3), 315–323. <https://doi.org/10.1016/j.jemermed.2012.11.092>
- [3] Eder, D. J. (1997, August 18). *Laboratory Atlas of Anatomy and Physiology* (2nd ed.). William C Brown Pub.
- [4] Real Academia Española. (2022). *Cuchillo*. In *Real Academia Española*. Retrieved September 9, 2022, from <https://dle.rae.es/cuchillo>

- [5] Police Department Brandon City Canada. (2007). BY-LAW NO. 6872 – KNIFE BY-LAW. In Brandon City Canada. Retrieved September 9, 2022, from: https://brandon.ca/images/pdf/CouncilAgendas/20070820_Agenda/By-law20No.206872.pdf
- [6] ¿Qué son objetos punzocortantes? - Desecho seguro de agujas - Tipos de objetos punzocortantes. (2019, October 25). Safe Needle Disposal. Retrieved September 9, 2022, from: <https://safeneedledisposal.org/es/manejo-de-objetos-punzocortantes/que-son-objetos-punzocortantes/>
- [7] Stereo Labs. (2022). Datasheet ZED2 Nov 2019 rev6 [Dataset; Pdf] from: <https://www.sterolabs.com/assets/datasheets/zed2-camera-datasheet.pdf>
- [8] xArm Collaborative Robot. (n.d.). UFACtORY. Retrieved September 9, 2022, from: <https://www.ufactory.cc/xarm-collaborative-robot>
- [9] Thingiverse.com. (n.d.). e-NABLE Phoenix Hand v3 by e-NABLE. Thingiverse. Retrieved September 9, 2022, from: <https://www.thingiverse.com/thin>
- [10] Phoenix Hand v3. (2021, September 20). e-NABLE. Retrieved September 9, 2022, from: <https://hub.e-nable.org/p/devices?p=e-NABLE+Phoenix+Hand+v3>
- [11] BionicBeaver/ReleaseNotes - Ubuntu Wiki. (n.d.). Retrieved September 9, 2022, from: <https://wiki.ubuntu.com/BionicBeaver/ReleaseNotes>
- [12] melodic - ROS Wiki. (n.d.). Retrieved September 9, 2022, from: <http://wiki.ros.org/melodic>
- [13] Documentation for Visual Studio Code. (2021, November 3). Retrieved September 9, 2022, from: <https://code.visualstudio.com/docs>
- [14] Ultralytics. (2022). GitHub - ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite. GitHub. Retrieved September 9, 2022, from: <https://github.com/ultralytics/yolov5>
- [15] OpenCV: Introduction. (n.d.). Retrieved September 9, 2022, from: <https://docs.opencv.org/4.6.0/d1/dfb/intro.html>
- [16] Release Notes: CUDA Toolkit Documentation. (n.d.). (C) Copyright 2005. Retrieved September 9, 2022, from: <https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html#cuda-general-new-features>
- [17] Point Cloud Library (PCL): PCL API Documentation. (n.d.). Retrieved September 30, 2022, from: <https://pointclouds.org/documentation/index.html>
- [18] GitHub rosperception/perceptionpcl: PCL (Point Cloud Library) ROS interface stack (By ROS Perception). (n.d.). GitHub. Retrieved September 30, 2022, from: https://github.com/ros-perception/perception_pcl
- [19] methylDragon. (n.d.). GitHub - methylDragon/pcl-ros-tutorial: A fairly in-depth tutorial for the Point Cloud Library (with ROS integration notes!). GitHub. Retrieved September 30, 2022, from: <https://github.com/methylDragon/pcl-ros-tutorial>
- [20] ANSI/RIA R15.06-2012 - Industrial Robots and Robot Systems - Safety Requirements (CONTAINS CORRIGENDUM). (n.d.).from: <https://webstore.ansi.org/Standards/RIA/ansiriar15062012>