Comandos de Movimiento de un Robot Delta por medio de Gestos Manuales

Movement Commands of a Delta Robot with Manual Gesture

Hansel Bonifacio Trujillo[1], Eugenio Yime Rodríguez[2], Javier Roldán McKinley[3]

[1]Ingeniero Mecánico, Universidad del Atlántico, Grupo DIMER, Barranquilla, Colombia.
[2]PhD. Automatización y Robótica. Universidad del Atlántico, Grupo DIMER. Barranquilla, Colombia.
[3]PhD. Ingeniería Mecánica. Universidad del Atlántico, Grupo DIMER. Barranquilla, Colombia.
eugenio.yime@mail.uniatlantico.edu.co

Cite this article as: H. Bonifacio, E. Yime, Javier Roldán, "Comandos de Movimiento de un Robot Delta por medio de Gestos Manuales", Prospectiva, Vol. 21 N° 1, 2023.

Recibido: 27/05/2022 / Aceptado: 10/04/2023
http:://doi.org/10.15665/rp.v21i1.2979

RESUMEN

Este artículo presenta la implementación de un sistema de visión artificial en un robot Delta, el cual logra comandar el robot en cada uno de sus seis movimientos posibles en el espacio cartesiano: derecha, izquierda, arriba, abajo, atrás y adelante. Para cada dirección de movimiento se asignó un comando gesticular, dando lugar a un lenguaje de seis gestos, todos realizados con la mano derecha. La detección de los gestos fue posible gracias al diseño de un software basado en las características geométricas de la mano. El código se desarrolló en el lenguaje Python, en conjunto con la librería OpenCV. La validez y efectividad en la detección de los gestos se demostró experimentalmente a través de pruebas donde se identificaron los mismos en diferentes condiciones de iluminación y con varias personas. Los resultados indican una alta efectividad en el reconocimiento de los gestos, logrando detectar todos y cada uno de ellos, con una confiabilidad superior al 96%. Además, el tiempo de detección de los gestos es corto, con un 90% de probabilidades de lograr la detección antes de los dos segundos. Si bien el software se desarrolló para utilizarse de forma conjunta con un robot paralelo tipo Delta; el mismo puede ser adaptado para interactuar con otros tipos de sistemas robóticos con tres grados de libertad traslacionales.

Palabras claves: *Robot Delta, comandos gesticulares, programación visual de robots, control por visión.*

ABSTRACT

This article presents the implementation of a computer vision system for commanding a Delta robot in academic labs. The software is able to command the robot in each of its six directions in cartesian space, i.e., right, left, up, down, back and forward. A gestural command was assigned for each direction of movement, giving rise to a language of six gestures, all performed with the right hand. Each gesture was detected thanks to the design of a software based on the geometric characteristics of the hand. The code was programmed in Python language using the OpenCV library. The validity and effectiveness of the gestures' detections was experimentally demonstrated through live tests using several people with different hand dimensions. The results show a high effectiveness in the recognition of gestures, managing to detect each and every one of them, with a reliability greater than 96%. In addition, the detection time of gestures is short, with a 90% of probability of achieving detection in two seconds. Although the software was developed to be used in conjunction with a Delta-type parallel robot; it can be adapted to interact with other types of robotic systems with three translational degrees of freedom.

Keywords: *Delta Robot, gesture commands, visual programming of robots, vision control.*

1. INTRODUCCIÓN

Los robots paralelos son dispositivos robóticos ampliamente utilizados en la industria por sus ventajas sobre los robots seriales como mayor velocidad, menor error de posicionamiento y mayor relación carga / peso, [1]. Uno de los robots paralelos de mayor influencia a nivel industrial es el robot Delta el cual posee tres grados de libertad traslacionales y se emplea ampliamente para aplicaciones de pick-and-place, [2], [3], [4].

Respecto a las desventajas de los robots paralelos respecto a los robots seriales, se tiene un modelo cinemático más complejo, un mayor esfuerzo al evaluar el modelo dinámico, mayor presencia de singularidades dentro del espacio de trabajo, entre otros, [5], [6], [7]. La mayor complejidad del modelo dinámico de los Robots paralelos implica un mayor esfuerzo computacional, que se transforma en un mayor tiempo de procesamiento al momento de realizar el control nolineal del sistema, [8], [9].

Si bien existen diversas formas de controlar un robot paralelo, como por ejemplo el control por par calculado [10], control basado en el modelo dinámico [11], [12], control por modelo dinámico inverso [13], [14], control en el espacio de la tarea [15]; el control cinemático inverso es la técnica más comúnmente empleada cuando se desea controlar de forma rápida y eficaz un robot paralelo como el robot Delta, [16]. [17], [18]. Una vez se tenga el control de bajo nivel, es decir el control de las juntas y los motores, se debe dotar al sistema robótico de un control de alto nivel, esto es un lenguaje de programación que permita que el sistema pueda ser programado por los seres humanos.

Los robots en general, ya sean seriales o paralelos, se programan empleando lenguajes acordes con los empleados por los fabricantes de robots. Así por ejemplo se tiene el lenguaje de programación RAPID utilizado en los robots fabricados por ABB, [19]. En el caso de la empresa KUKA esta utiliza en sus robots el lenguaje KRL, acrónimo de KUKA Robot Language, [20]; para el caso de los robots EPSON existe el lenguaje EPSON SPEL+, [21]; y así se puede enumerar muchos lenguajes más.

Una alternativa a la programación de robots por lenguaje escrito es el comando por gestos, el cual posee la ventaja de ser un procedimiento interactivo con el robot, facilitando la programación del movimiento el mismo. Ejemplos de control de robots por gestos se tiene a [22] donde se controla con movimientos en la mano un robot móvil. En [23] se utiliza un DSP para programar el reconocimiento de los gestos manuales. En [24] se utiliza el reconocimiento en tiempo real de gestos manuales para mover un robot móvil. En [25] se controla un robot AX-12 empleando movimientos en el brazo. En [26] se emplean los gestos para mover un robot de rehabilitación y mejorar el desempeño de la terapia. Otro ejemplo de control por gestos de robot se aprecia en [27].

El presente artículo trata sobre el control gesticular de un robot Delta construido en la Universidad del Atlántico, [28] se ha decidido dotar al robot de una etapa de control de bajo nivel empleando a cinemática inversa. Sobre dicha capa existirá el control de alto nivel basado en el control de gestos. El sistema de control estará constituido por un sistema de visión artificial el cual reconocerá los gestos del programador y procederá a mover el robot acorde con los mismos. En las siguientes secciones se explicará con detalle las diferentes etapas involucradas en el desarrollo del sistema de control basado en visión artificial.

2. METODOLOGÍA

La investigación se desarrolló en varias etapas, donde cada una de ellas representaba una fase u objetivo específico a conseguir como paso previo al objetivo final. Las siguientes son las fases involucradas durante el desarrollo de la investigación.

a) Diseño de una interfaz de comunicación hombre-máquina por medio de gestos manuales.

b) Desarrollo de un programa de visión artificial para el reconocimiento de los gestos manuales.

c) Realización de experimentos para la validación, y determinación del porcentaje de éxito, del software de reconocimiento de los gestos manuales.

d) Programación de movimiento del robot Delta a partir de los gestos manuales definidos como comandos.

La primera fase fue la definición de los comandos gesticulares que se emplearían como el lenguaje para mover el robot Delta. Esta etapa se explica con mayor detalle en la sección 2.1. La segunda fase es el desarrollo de un programa de visión que sea capaz de reconocer los gestos; para ello fue necesario caracterizar los gestos y programar un software capaz de reconocerlos. La sección 2.2 explica con mayor detalle los desarrollos logrados. La tercera fase implica la validación del software de reconocimiento a través de experimentos con el mismo. Esta fase se explica en la sección 3. La última fase es la programación como tal del controlador del robot Delta y validar que es posible mover le mismo a partir de gestos manuales. Esta fase se explica con detalle en la sección 4.

2.1. Definición de la Interfaz de Comunicación con Gestos Manuales

En la revisión del estado del arte sobre avances previos en el área de reconocimiento de gestos manuales, se encontró que Villa et al., [29], realizaron el desarrollo de un sistema de reconocimiento de gestos no móviles usando como principal herramienta de desarrollo el entorno de Matlab; su objetivo fue la contribución al aprendizaje del lenguaje de señas en Colombia. En García, [30], se encuentra una propuesta para una estrategia de detección de gestos utilizando cámaras web para interactuar con un robot. En [31] y [32] se presentan dos desarrollos de reconocimiento de gestos utilizando como plataforma de desarrollo la cámara Kinect de Microsoft. En [33] se ilustra un desarrollo de un software para el reconocimiento de gestos empleando un sensor gesticular producido por la compañía Leap Motion, y con interfaz a ROS. En [34], por otro lado, emplean un sensor Kinect para detectar la posición corporal, a partir de la cual se detectan los comandos a enviar a un robot.

Una vez revisada la literatura de gestos para comandar robots, se procedió a seleccionar y definir los gestos con la mano que se utilizarán para mover el robot paralelo Delta. Los requisitos para la selección y/o definición de los gestos es que los mismos fueran sencillos de realizar con la mano derecha, y que a su vez tuvieran un significado asociado con el movimiento deseado del robot. La figura 1 contiene los seis comandos que se definieron para mover el robot.

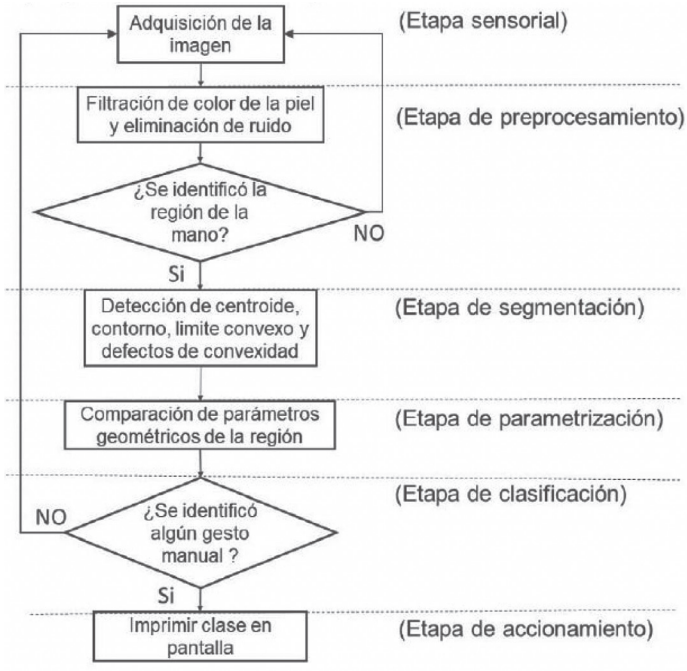
Figura 1. Lenguaje Gesticular Propuesto
Figura 1. Proposed Gesture Languaje

Los comandos gesticulares fueron seleccionados teniendo en cuenta que el robot Delta posee tres grados de libertad, los cuales están asociados al movimiento lineal en los ejes cartesianos X, Y y Z. Por lo tanto, se requieren de seis comandos, o gestos manuales, para poder mover el robot en las dos direcciones posibles de cada uno de los grados de libertad. Se escogió la mano derecha por cuanto la mayoría de la población es diestra.

2.2. Software de reconocimiento de Gestos Manuales

Una vez definido los gestos para controlar el robot Delta, se procede con el desarrollo del software para el reconocimiento de los mismos. El lenguaje seleccionado para el desarrollo del software fue Python ya que el controlador del robot Delta se ejecuta en un entorno Linux. En ese sentido, Python tiene la ventaja de ser multiplataforma, lo cual permite que el desarrollo del software pueda realizarse en Windows y posteriormente portarse a Linux, que es donde se ejecuta el control en tiempo real del robot Delta. La figura 2 ilustra el algoritmo del software desarrollado. Ahí se puede apreciar que el funcionamiento del software se ha dividido en seis etapas: adquisición, procesamiento, segmentación, parametrización, clasificación y accionamiento.

Figura 2. Algoritmo del software desarrollado para el reconocimiento de los gestos manuales.
Figura 2. Software algorithm used for gesture recognition.



2.1.1. Adquisición de la imagen

El proceso de adquisición de la imagen utilizando la librería OpenCV es bastante directo, e implica el llamado a funciones internas. El software toma solo una sección de dimensiones 300 x 300 píxeles para extraer la mano. La zona de detección se le indica al usuario por medio de un recuadro verde en la interfaz de usuario (GUI). En la sección 3.2 se explica con mayor detalle el funcionamiento de dicha interfaz.

2.1.2. Procesamiento

La detección de la mano en la zona de 300 x 300 píxeles se logra al hacer una conversión del espacio de color RGB al espacio HSV. Posteriormente, se procesa la imagen para generar un filtro por color y extraer la mano. La figura 3 muestra el resultado del proceso de reconocimiento de la mano. En el lado izquierdo se muestra la imagen en el espacio de color RGB, y del lado derecho la imagen binaria después de la extracción.

Figura 3. Comparación entre la imagen capturada y el resultado del procesamiento para obtener el gesto manual.
Figura 3. Comparison between the original captured image and the result after isolating the gesture.

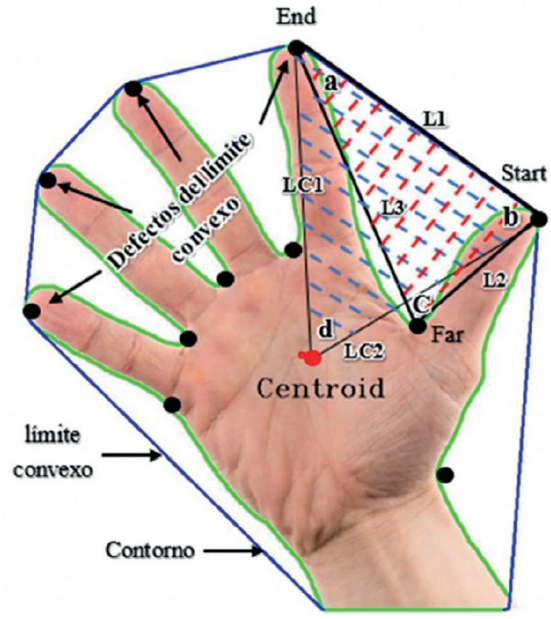
La imagen binaria extraída posee ruido de fondo debido a que algunos pixeles pueden coincidir con el criterio de comparación por color, pero no pertenecen a la mano. El ruido se elimina a través de un proceso de filtrado llamando a las funciones erode(), diltate(), y GaussianBlur() de OpenCV. La figura 4 ilustra el resultado del proceso de filtrado.

Figura 4. Imagen con ruido vs imagen filtrada.
Figura 4. Imaging with noise and after noise filtering.

2.1.3. Segmentación

Esta etapa tiene por objetivo determinar el contorno externo de la mano, es decir el límite entre la zona blanca y negra en la figura 4. Adicionalmente, se identifican una serie de puntos en el contorno con el objetivo de utilizarlos en la siguiente etapa de parametrización. En la figura 5 se muestra parte del resultado de la segmentación y la parametrización. La segmentación da por resultado el contorno de la mano, línea color verde, y los puntos negros de interés para la parametrización, dichos puntos son: centroid, far, start y end. La línea verde es el resultado del cálculo del contorno utilizando la función *findContours()*. El punto rojo, denominado "centroid", es el centroide del contorno y se obtiene promediando la imagen. La línea azul es el límite del convexo obtenido con la función *convexHull()*. Los puntos negros son los defectos de convexidad obtenidos con *convexityDefects()*.

Figura 5. Definición de los puntos de parametrización con una mano abierta
Figura 5. Parametrization points with an opened hand.



2.1.4. Parametrización y Clasificación

La parametrización se logra por medio la caracterización geométrica de dos triángulos formados con los puntos devueltos por la función *convexityDefects()*. En la figura 5 se han identificado dichos puntos con los nombres "Far", "Start" y "End". Estos tres puntos forman el primer triángulo empleado en la parametrización. El segundo triángulo es el formado por los puntos "Centroid", "Star" y "End". En la parametrización se obtienen las características geométricas de estos triángulos que ayudarán a identificar el comando. En el proceso de "Clasificación" se comparan las características geométricas detectadas con los parámetros establecidos para cada gesto.

Los parámetros de cada gesto se obtuvieron a partir de la toma de cuatro imágenes por comando en cinco personas diferentes. A partir de dichas imágenes se estableció un rango de valores para cada comando. Las tablas 1 hasta 6 contienen el resumen de los resultados de la caracterización geométrica de cada comando.

Tabla 1. Caracterización geométrica del comando mover hacia la izquierda.
Tabla 1. Geometrical characterization of move to left gesture.

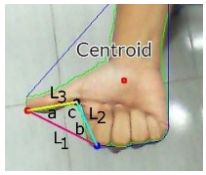
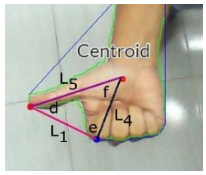
Imagen	Ángulo	Lado	Área
	a: [30-48] b:[25-40] c:[95-120]	 L1 > L3 L3 > L2	 1500 10000
	d:[35-90] e:[55-95] f:[25-65]	 L4 > L1 L4 > L2	 3000 18000

Tabla 2. Caracterización geométrica del comando mover hacia la derecha.
Tabla 2. Geometrical characterization of move to right gesture.

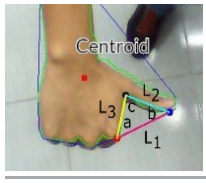
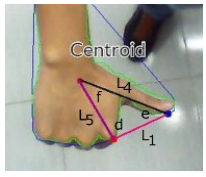
Imagen	Ángulo	Lado	Área
	a: [25-45] b:[33-50] c:[90-120]	 L1 > L2 L2 > L3	 3000 18000
	d:[35-85] e:[60-100] f:[25-65]	 L4 > L5 L5 > L1	 4000 6000

Tabla 3. Caracterización geométrica del comando mover hacia adelante.
Tabla 3. Geometrical characterization of move to forward gesture.

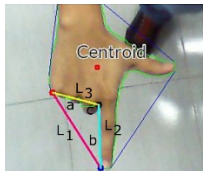
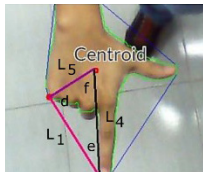
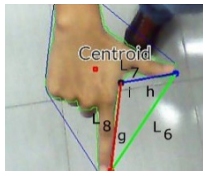
Imagen	Ángulo	Lado	Área
	a: [25-45] b:[30-50] c:[25-40]	 L1 > L2 L2 > L3	 3500 7000
	d:[70-120] e:[15-30] f:[35-75]	 L4 > L1 L1 > L5	 4000 10000
	g:[15-30] h:[45-60] i:[90-115]	 L6 > L8 L8 > L7	 4000 10000

Tabla 4. Caracterización geométrica del comando mover hacia atrás.
Tabla 4. Geometrical characterization of move to backward gesture.

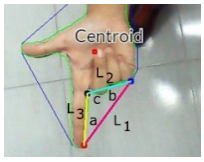
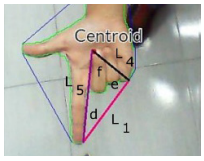
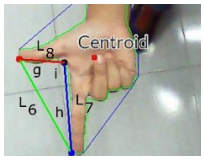
Imagen	Ángulo	Lado	Área
	a: [30-50] b:[25-35] c:[90-120]	L1 > L3 L3 > L2	3000 16000
	d:[70-110] e:[20-40] f:[40-70]	L1 > L4 L5 > L4	4000 12000
	g:[40-85] h:[15-45] i:[75-120]	L6 > L7 L7 > L8	3000 16000

Tabla 5. Geometrical characterization of move to up gesture.

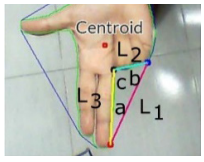
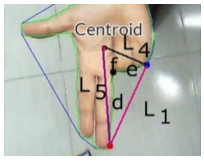
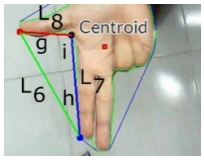
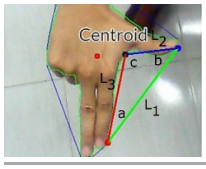
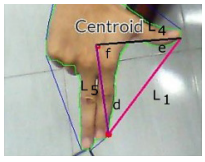
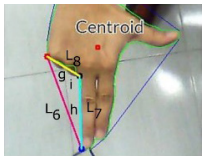
Imagen	Ángulo	Lado	Área
	a: [15-26] b:[35-50] c:[95-135]	L1 > L3 L3 > L2	1000 10000
	d:[20-40] e:[75-130] f:[45-70]	L5 > L4 L5 > L1	5000 10000
	g:[40-80] h:[15-45] i:[75-120]	L6 > L7 L7 > L8	5000 10000

Tabla 6. Geometrical characterization of move to down gesture.

Imagen	Ángulo	Lado	Área
	a: [15-25] b:[25-40] c:[115-140]	L1 > L2 L2 > L3	3000 6500
	d:[20-45] e:[50-90] f:[55-90]	L1 > L4 L1 > L5	4000 10000
	g:[20-35] h:[35-60] i:[90-120]	L6 > L8 L8 > L9	3000 6500

2.1.5. Validación de los parámetros y efectividad del reconocimiento gesticular

La siguiente prueba que se realizó consistió en validar la efectividad de la caracterización geométrica en el reconocimiento de los comandos gesticulares. Para ello se procedió a invitar a cinco personas diferentes a las que participaron en la obtención de los parámetros geométricos. A cada persona se le pidió que repitiera cada gesto cuatro veces, y se procedió a calcular las tasas de acierto y fallo de la caracterización. Los resultados se resumen en la tabla 7, en ella se puede apreciar una tasa de acierto superior al 90%.

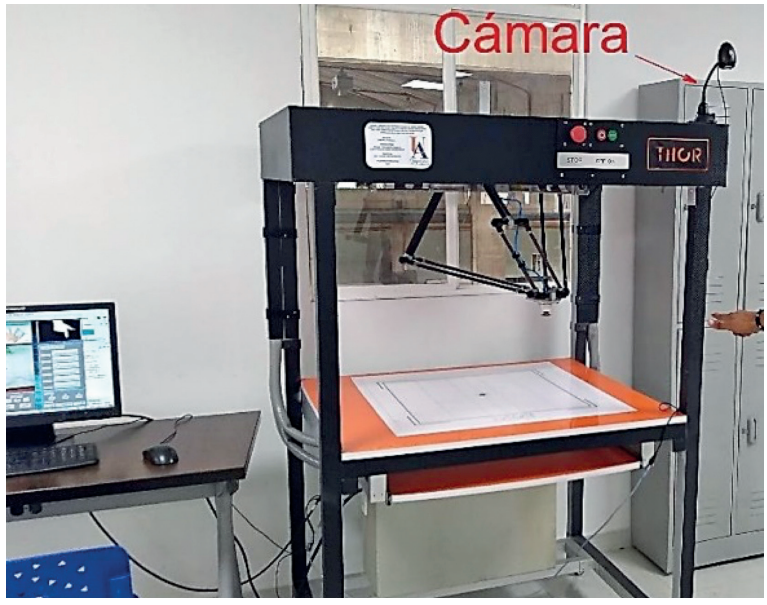
Tabla 7. Tasas de acierto durante el reconocimiento de los comandos gesticulares.
Tabla 7. Success rates for the recognition of gesture commands.

Gesto	Reconocidos	Fallados	Efectividad
Derecha	20	0	100%
Izquierda	19	1	95%
Arriba	20	0	100%
Abajo	19	1	95%
Adelante	18	2	90%
Atrás	19	1	95%

3. MATERIALES Y EQUIPOS

La figura 6 muestra el montaje final del sistema de visión artificial en conjunto con el robot Delta. En la parte izquierda se aprecia un computador, en donde se ejecuta el control de movimiento del robot y el software de reconocimiento visual de gestos. En la parte derecha se aprecia el robot Delta, encima del cual se encuentra la cámara para la captura y reconocimiento de los gestos manuales. En las siguientes secciones se explicará brevemente como es el funcionamiento del robot Delta y como fue el diseño de la interfaz de usuario.

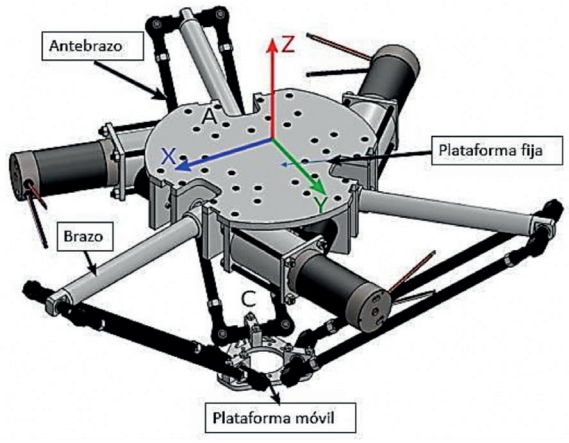
Figura 6. Conjunto sistema de visión y robot Delta.
Figura 6. Delta Robot and Vision System.



3.1. Robot Delta

La figura 7 ilustra el diseño CAD del robot Delta, donde se puede observar los tres motores sin escobillas que accionan a cada brazo, que a su vez están unidos por un sistema tipo mecanismo de cuatro barras a la plataforma móvil. En la parte superior, es decir la plataforma fija, se ha ubicado un sistema de referencia para definir los ejes a lo largo de los cuales se realizan los tres grados de libertad lineales de la plataforma móvil.

Figura 7. Diseño CAD del Robot Delta.
Figura 7. Delta Robot CAD Design.

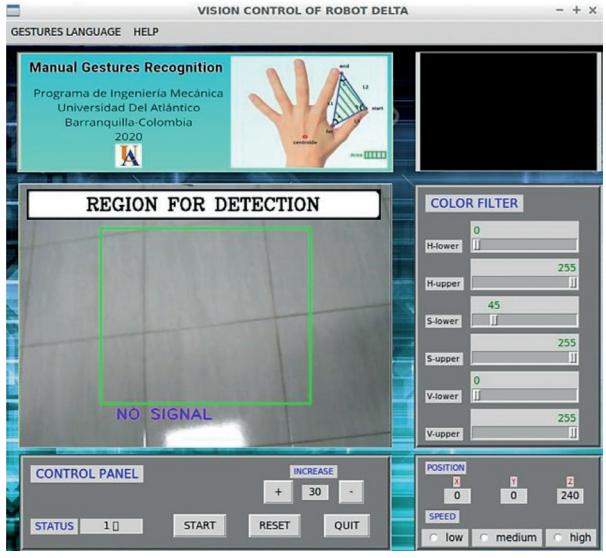


El control de movimiento del robot Delta es un control de posición PID basado en la cinemática inversa del robot. Este control se ejecuta en tiempo real en un computador PC104 que ejecuta Linux. En la misma PC104 se ha agregado el software de reconocimiento visual de gestos manuales. Entre el software de visión y el control PID existe una comunicación basada en comandos a través de la terminal o Shell del sistema operativo. Los comandos son: "DeltaHome", para realizar la posición inicial o cero cinemático del robot; "DeltaStatus", para conocer si el robot está disponible para realizar un movimiento o se encuentra en situación de error; "DeltaReset", para recuperar el control a partir de un error durante el movimiento; "DeltaMoveLinear", para mover el robot en línea recta en el espacio cartesiano entre dos puntos; "DeltaMoveCircular", para mover el robot describiendo un círculo en el espacio cartesiano; "DeltaQuit", para terminar el control del robot. El software de reconocimiento de gestos manuales fue desarrollado en Phyton, y se comunica con el control PID por medio de llamados a la función os.system() para poder acceder a la Shell de Linux, de esta manera se logra que los movimientos del robot Delta sean fluidos y transparentes al usuario.

3.2. Interfaz de Usuario

La figura 8 ilustra el diseño de la interfaz de usuario para el software de reconocimiento de los gestos manuales. La interfaz se ha dividido para efectos de explicación en siete secciones.

Figura 8. Interfaz de Usuario del Sistema de Visión
Figura 8. Vision system GUI.



La primera sección son los menús de la ventana, "Gestures Language" para la explicación de los gestos manuales donde se visualizan las imágenes de cada gesto, resumidas en la figura 1, y "Help" que abre una ventana con la información del código, creador, entidad y año. La segunda sección tiene por título "Manual Gestures Recognition" y es la información del software, similar a la visualizada en el menú "Help". La sección tercera está localizada al lado de "Manual Gestures Recognition" y su función es reproducir la imagen binaria de la mano después del filtrado, similar a la figura 4, lo cual sirve para que el usuario aprecie como está funcionando el filtro por color y la eliminación del ruido. La cuarta sección tiene el título de "Region for Detection" y es donde se visualiza la toma proveniente de la cámara; su objetivo es que el usuario coloque el gesto manual dentro de la ventana de color verde que tiene una dimensión de 300 x 300 píxeles. La quinta sección tiene el título "Color Filter" y son seis sliders para que el usuario pueda mejorar el filtrado por color de la mano en caso de que las condiciones de iluminación y ruido de fondo no contribuyan al reconocimiento del gesto. La sexta sección es el panel de control del robot delta, donde se encuentran tres botones: "start", "reset" y "end". El botón "start" cuando se presiona realiza el llamado a "DeltaHome" procediendo con la puesta a cero cinemático del robot. El botón "reset" es para recuperar el robot en caso de que ocurra un error. El botón "end" es para finalizar el software y hacer un llamado a "DeltaQuit". En esta sección también hay un indicador del estado del robot a través de la etiqueta "status" al lado del cual aparece el estado numérico del robot. Agrupados bajo la etiqueta "increase" se encuentran en la parte superior dos botones para cambiar el desplazamiento que realizará el robot cada vez que reconozca uno de los seis comandos gesticulares; estos desplazamientos están en milímetros y se agregará como parámetros a los llamados a función de mover en línea recta "DeltaMoveLinear". Los desplazamientos por cada gesto reconocido varían en el rango de 0 mm a 50 mm, según decisión del usario. Por último, la última subventana, contiene las etiquetas "Position" y "Speed" y se utilizan para visualizar la posición cartesiana actual del robot, y para seleccionar tres velocidades de movimiento lineal del robot Delta. Las velocidades son 80 mm/s para la velocidad baja, 120 mm/s para la velocidad media y 150 mm/s para la velocidad alta.

4. RESULTADOS

La prueba final, para validar la efectividad y eficiencia del software de reconocimiento gesticular y control del robot Delta, consistió en invitar a cincuenta personas diferentes para que movieran el robot utilizando los seis comandos gesticulares. Para cada comando se anotó si fue reconocido adecuadamente y el tiempo que demoró el reconocimiento. El gesto se da por reconocido si el robot se mueve en la dirección indicada por el gesto. Se da por no reconocido si el robot no se mueve dentro de los cinco segundos después de realizar el gesto dentro de la venta de color verde en la sección "Region for Detection". Los resultados de este experimento se encuentran resumidos en las tablas 8 y 9.

Figura 8. Interfaz de Usuario del Sistema de Visión
Figura 8. Vision system GUI.

Gesto	Reconocidos	Fallados
Derecha	49 (98%)	1 (2%)
Izquierda	48 (96%)	2 (4%)
Arriba	50 (100%)	0 (0%)
Abajo	50 (100%)	0 (0%)
Adelante	49 (98%)	1 (2%)
Atrás	100 (100%)	0 (0%)

Tabla 9. Tiempo requerido para la detección del comando gesticular.
Tabla 9. Gesture reconigtion proccesing time.

Gesto	Reconocidos	Fallados
Derecha	47 (94%)	3 (6%)
Izquierda	45 (90%)	5 (10%)
Arriba	49 (98%)	1 (2%)
Abajo	45 (90%)	5 (10%)
Adelante	47 (94%)	3 (6%)
Atrás	45 (90%)	5 (10%)

Un análisis de los resultados agrupados en la tabla 8 arroja que el software tiene un buen indicador de reconocimiento de los gestos, superior al 96 por ciento, lo que significa que es muy eficaz, es decir de uso práctico, ya que el usuario no se verá sometido a una alta tasa de errores o la sensación de que no es un software muy útil. En cuanto al tiempo de procesamiento, tabla 9, se puede deducir que el 90% de las veces el software logra reconocer el gesto dentro de los dos primeros segundos, es decir es eficiente. La tasa restante del 10% que demora más de 2 segundos,

puede tener como causante el procesamiento interno en Python. Sin embargo, se necesita de un análisis más detallado para identificar las verdaderas causas y poder disminuir el tiempo de procesamiento.

A modo de aclaración, hay que decir que una vez que el software reconoce un gesto, este gesto queda internamente retrasado por software para su posterior reconocimiento. El objetivo es evitar que se reconozca un solo gesto como varios comandos iguales ejecutados secuencialmente uno después del otro, generando así un movimiento prolongado e indeseable en el robot.

5. CONCLUSIONES

En el presente artículo de investigación se ha estudiado una forma de enviar comandos de movimiento, a un robot paralelo tipo Delta, utilizando gestos con la mano. El principal objetivo de emplear este enfoque es facilitar la operación de comandar un robot Delta . De esta forma, se pretende ampliar el uso del robot a personas con pocos conocimientos en programación. Dentro de los pasos realizados se tuvo que diseñar una interfaz gráfica que de forma intuitiva y simple ayuda a que las personas con pocos conocimientos en robótica puedan manipular y mover el robot Delta dentro de su espacio de trabajo. El diseño de la interfaz fue validado con diez personas diferentes, estudiantes de Ingeniería Mecánica de la Universidad del Atlántico, los cuales manifestaron como aspectos positivos la simplicidad del diseño gráfico y la facilidad de los comandos gesticulares.

La validez del software de reconocimiento de comandos gesticulares se realizó a través de un experimento, donde un grupo de usuarios probó si el software era capaz de reconocer el gesto realizado y mover el robot Delta. Los resultados obtenidos arrojaron una fiabilidad mayor al 96% en el reconocimiento de los gestos; además de una duración menor de 2 segundos, en el 90% de los casos, para lograr reconocer el gesto.

A partir de los buenos resultados obtenidos, se espera que el software pueda ser empleado a futuro como experimento en las sesiones de robótica, con el objeto de poder obtener un programa completo de movimiento del robot Delta asociado a una tarea más compleja, donde los puntos deseados de traslación del robot se programen de forma visual sin tener que interactuar con un teclado o un ratón.

BIBLIOGRAFÍA

[1] J. Merlet, Parallel Robots, 2nd ed. Springer, 2005.

[2] R. Clavel, (1990, Dic.) "*Conception d'un robot parallele rapide a 4 degres de liberte*", Tesis de Doctorado, Ecole polytechnique federale de Lausanne EPFL, 1990.

[3] R. Clavel, "Device for the movement and positioning of an element in space," Switzerland Patent 4.976.582, 1990.

[4] C. Boer, T. Molinari, and L. Smith, Parallel Kinematic Machines, 1st ed. Springer, 1999.

[5] S. Staicu, Dynamics of Parallel Robots, 1st ed. Springer, 2018.

[6] S. Briot and W. Khalil, Dynamics of Parallel Robots: From Rigid Bodies to Flexible Elements, 1st ed. Springer, 2015.

[7] W. Khalil and O. Ibrahim, "General solution for the dynamic modeling of parallel robots," Journal of Intelligent and Robotic Systems, 49, 19–37, 2007.

[8] H. D. Taghirad, Parallel Robots: Mechanics and Control, 1st ed. CRC Press, 2013.

[9] D. Zhang, Parallel Robotic Machine Tools, 1st ed. Springer, 2010.

[10] A. Codourey, "Dynamic modeling of parallel robots for computed torque control implementation," The International Journal of Robotics Research., 17 (12), 1325–1336, 1998.

[11] H. S. Kim, "Dynamics modeling and control of a delta high-speed parallel robot," Journal of the Korean Society of Manufacturing Process Engineers, 13 (5), 90–97, 2014.

[12] W. P. Feng, Z. L. Min, and Z. X. Man, "Dynamic modeling, simulation and experiment of the delta robot," En: Future Communication, Computing, Control and Management, ser. 141, Y. Zhang, ed. Springer, 2012, p. 149–156.

[13] K. Miller and B. S. Stevens, "Modeling of dynamics and model-based control of delta direct-drive parallel robot," Journal of robotics and mechatronics, 7 (4), 344–352, 1995.

[14] M. Rachedi, (2017, Feb.), "Model based control of 3 dof parallel delta robot using inverse dynamic model", Presentado en 2017 IEEE International Conference on Mechatronics and Automation (ICMA), 2017.

[15] P. Guglielmetti and R. Longchamp, (1992, Oct.), "Task space control of the delta parallel robot". Presentado en IFAC Workshop on Motion Control for Intelligent Automation, 1992.

[16] F. C. Can, M. Hepeyiler, and O. Baser, "A novel inverse kinematic approach for delta parallel robot," International Journal of Materials, Mechanics and Manufacturing, 6 (5), 321–326, 2018.

[17] D. Rivas, E. Galarza, D. Turbaco, and W. Quimbita, "Delta robot controlled by robotic operating system," ITECKNE, 12 (1), 54–59, 2015.

[18] V. Damic, M. Cohodar, and A. Voloder, (2018, Oct.), "Modelling and path planning of delta parallel robot in virtual environment". Presentado en 29th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2018.

[19] Technical reference manual RAPID Instructions, Functions and Data types, ABB Robotics, 2010.

[20] KUKA System Software 8.3, operating and Programming Instructions for End Users, KUKA Roboter GmbH, 2013.

[21] EPSON RC+ 7.0 (Ver.7.5) SPEL+ Language Reference Rev.1, SEIKO EPSON CORPORATION, 2020.

[22] K. Sekar, R. Thileeban, V. Rajkumar, and S. B. Sembian, "Hand gesture-controlled robot," International Journal of Engineering Research & Technology, 9 (11), 17–19, 2020.

[23] J. L. Raheja, G. A. Rajsekhar, and A. Chaudhary, (2016, Oct.), "Controlling a remotely located robot using hand gestures in real time: A DSP implementation," Presentado en 5th International Conference on Wireless Networks and Embedded Systems (WECON), 2016.

[24] A. A. M. Faudzi, M. H. K. Ali, M. A. Azman, and Z. H. Ismail, (2012, Sept.) "Real-time hand gestures system for mobile robots control," Presentado en International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[25] T. Grzejszczak, M. Mikulski, T. Szkodny, and K. Jedrasiak, "Gesture based robot control", En: Computer Vision and Graphics, Ed: L. Bolc, R. Tadeusiewicz, L. J. Chmielewski, K. Wojciechowski. Springer Berlin Heidelberg, p. 407–413.

[26] A. D. Segal, M. C. Lesak, A. K. Silverman, and A. J. Petruska, "A gesture-controlled rehabilitation robot to improve engagement and quantify movement performance," Sensors, 20 (15), 2020.

[27] V. S. Rao and C. Mahanta, "Gesture based robot control," (2006, Dic.), Presentado en 2006 Fourth International Conference on Intelligent Sensing and Information Processing, 2006.

[28] R. Castro and A. Manotas, (2018, Jun.), "*Diseño, construcción e implementación de un mecanismo paralelo tipo Delta con arquitectura de control abierta, para el uso como herramienta tecnológica en los procesos de enseñanza en la Robótica Educativa*", Tesis en Ingeniería Mecánica, Universidad del Atlántico, Barranquilla, Colombia, 2018.

[29] B. Villa, V. Valencia, and J. Berrio, "Diseño de un sistema de reconocimiento de gestos no móviles mediante el procesamiento digital de imágenes," Prospectiva, 16, 41–48, 2018.

[30] D. García, (2014, Mayo), "*Reconocimiento de gestos de manos como mecanismo de interacción humano-robot*", Tesis de Maestría, Universidad Nacional de Colombia, Bogotá, Colombia, 2014.

[31] J. Millán, (2015, Agosto), "*Reconocimiento gestual para interacción humano-robot basado en ROS*", Proyecto Final de Carrera, Departamento de Ingeniería de Sistemas y Automática. Escuela Técnica Superior de Ingeniería. Universidad de Sevilla, Sevilla, España, 2015.

[32] E. León, (2015, Abril), "*Reconocimiento de gestos del mano aplicado al desarrollo de una interfaz kinect para el museo regional de huajuapan*" Tesis de Maestría, Universidad Tecnológica de la Mixteca, Oaxaca, Mexico, 2015.

[33] P. Lázaro, (2017, Agosto) "*Módulo de reconocimiento gestual para control de robot en tareas de asistencia*", Proyecto Final de Carrera, Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III, Leganés, Madrid, España, 2017.

[34] J. Estefan, (2013, Dic.) "*Arquitectura de telecontrol de un robot mediante el uso de interfaces gestuales*", Tesis de Maestría, Universidad EAFIT, Medellín, Colombia, 2013