risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido/Submission: 16/09/2019
Aceitação/Acceptance: 13/01/2020

Algoritmo para controlar un brazo antropomórfico usando una transformación lineal

Yadira Quiñonez[1], Oscar Zataraín[1], Carmen Lizárraga[1], Juan Peraza[1], Juan Peraza[1], Jezreel Mejía[2]

yadiraqui@uas.edu.mx, 2016030617@upsin.edu.mx, carmen.lizarraga@uas.edu.mx, jfperaza@uas.edu.mx, restrada@uas.edu.mx ozatarain@uas.edu.mx, jmejia@cimat.mx

[1] Universidad Autónoma de Sinaloa, Facultad de Informática, Av. Leonismo Internacional S/N y Av. de los Deportes, Ciudad Universitaria, 82000, Mazatlán, Sinaloa, México.

[2] Centro de Investigación en Matemáticas, Unidad Zacatecas, Parque Quantum, Ciudad el Conocimiento Avenida Lassec, Andador Galileo Galilei, Manzana, 3 Lote 7, 98160. Zacatecas, México.

DOI: 10.17013/risti.36.65–81

Resumen: En este artículo se propone un algoritmo para manipular un brazo antropomórfico utilizando una pantalla táctil LCD y una transformación lineal para obtener la trayectoria del brazo robótico, cinemática inversa y directa. Con la finalidad de que las personas con discapacidades físicas que tienen algún problema para mover sus cuerpos o simplemente no tienen la fuerza para moverlo puedan controlar el brazo robótico en las actividades cotidianas de una forma natural y fácil. La implementación del algoritmo se ha desarrollado en MATLAB, a través de esta simulación se visualiza la trayectoria del brazo robótico y la representación con las tres articulaciones que son movidas por motores paso a paso. El artículo termina con una discusión crítica de los resultados experimentales.

Palabras-clave: Brazo Antropomórfico; Transformación Lineal; Pantalla Táctil LCD; Personas con Discapacidades Físicas; MATLAB.

Algorithm to control an anthropomorphic arm using a linear transformation

Abstract: In this paper proposes an algorithm to manipulate an anthropomorphic arm using an LCD touch screen and a linear transformation to obtain the trajectory robotic arm, inverse and direct kinematics. In order to people with physical disabilities who have an issue to move their bodies or simply do not have enough force to move it, can control the robotic arm in daily activities in a natural and easy way. The implementation of the algorithm has been developed in MATLAB, through this simulation the trajectory of the robotic arm is visualized and the representation with the three joints that are moved by stepper motors. The article ends with a critical discussion of the experimental results.

Keywords: Anthropomorphic Arm; Linear Transformation; LCD Touch Screen; People with Physical Disabilities; MATLAB.

1. Introducción

En las últimas dos décadas, la automatización y el control se han convertido en un tema de interés para investigadores en diferentes áreas de aplicación, donde lo más importante es la Experiencia de Usuario (UX) (Fernández-Ordoñez, 2019), esto con la finalidad de facilitar la interacción hombre-máquina tan natural como sea posible (Lara, 2019). Exite una gran variedad de aplicaciones de autimatización y control, sin embargo, el área médica has sido muy beneficiada, a continuación, se mecionan algunos trabajos relacionados con ésta área. En la robótica industrial (Grau, 2017, Yenorkar, 2018) y en los sistemas robotizados aplicados en el área médica, tales como la cirugía tele operada (Burgner-Kahrs, 2015), el corte de patrones quirúrgicos (Murali, 2015), según Shademan et al. (2016) la precisión de un robot industrial ofrece grandes ventajas en esta área. Actualmente, existen un sinfín desarrollos tecnológicos y diversas aplicaciones como prótesis (Allen, 2016), ortesis (Niyetkaliyev, 2017), exoesqueletos (Proietti, 2016, Rehmat 2018, Young, 2017) y dispositivos para la tele operación con el fin de mejorar las capacidades humanas (Makin, 2017, Beckerle, 2018, Jiang, 2012, Kruthika, 2016). De acuerdo con Chung et al. (2013) realizó una revisión de la literatura sobre los diferentes manipuladores de asistencia robótica desde 1970 hasta 2012, menciona que la confiabilidad, rentabilidad, apariencia, funcionalidad y facilidad de uso son factores determinantes para lograr una comercialización exitosa. Algunos trabajos relacionados para la rehabilitación de pacientes con capacidad de movimiento reducida han utilizado la realidad virtual para mejorar la manipulación de los movimientos e influir positivamente en un entorno virtual (Perez-Marcos, 2017, Levin, 2015, Kokkinara, 2015, Bovet, 2018). Recientemente, en un trabajo de Atre et al. (2018) han utilizado técnicas de visión por computadora para crear un brazo robótico controlado por gestos, mencionan la importancia de que un brazo robótico sea más eficiente en relación con la precisión y la velocidad, en otros trabajos, Badrinath et al. (2016) proponen el control de un brazo robótico utilizando una cámara Kinect 3-D para rastrear los movimientos de las personas mediante la técnica de visión por computador.

2. Descripción del algoritmo

En la actualidad, es común utilizar un joystick para controlar los movimientos de un brazo robótico, de hecho, el uso del joystick es implementado en muchos tipos de aplicaciones electrónicas y una de estas, en la que se enfoca este trabajo, es controlar un brazo robótico antropomórfico para dar asistencia a las personas con discapacidades físicas. El Joystick puede ser una gran herramienta de control, sin embargo, al momento de implementarlo en alguna problemática en particular como lo es el brazo robótico para personas con discapacidades físicas no cubre con las necesidades o expectativas para sacar el máximo aprovechamiento, debido a que se requiere de un esfuerzo constante y tiempo de espera para ejercer una fuerza, sin mencionar que se necesita coordinación al momento de ejecutarlo. Suponiendo que un paciente también sufre de diabetes (en el caso de hipoglucemia) o para las personas que sufren de lupus u otra enfermedad que impida tener una buena coordinación, usar un Joystick puede ser una alternativa complicada para las personas que sufren alguno de estos casos, ya que a veces se requieren movimientos muy largos para que el brazo robótico llegue al punto

deseado, y aunque el Joystick sea muy fácil de usar y programar, controlar un brazo robótico puede ser aún más fácil, sin tanto esfuerzo, sin aplicar una fuerza constante, de manera intuitiva y con un solo dedo poder mover el brazo robótico hasta el punto deseado. Esto es posible utilizando una pantalla LCD táctil, que ayuda a ejecutar el algoritmo usando una transformación lineal (también llamada función lineal, aplicación u operador lineales) la cual es una herramienta importantísima para la ejecución del algoritmo y diversos problemas, en el siguiente apartado se desarrolla este tema.

Al implementar este control, una persona con limitaciones físicas solo es necesario realizar un ligero movimiento con el dedo y tocar un punto en la superficie de la pantalla táctil, para hacer que el brazo robótico obtenga un punto en el espacio, realizar este proceso es similar a utilizar un teléfono celular, una tableta o cualquier otro aparato electrónico que cuente con una pantalla táctil y que se puedan realizar las actividades cotidianas que se realizan hoy en día, de esta forma, se proporciona al paciente más libertad para controlar un brazo robótico en las actividades cotidianas de una forma natural.

El algoritmo está pensado para que el brazo robótico antropomórfico llegue al punto deseado a más tardar en un segundo, es decir, todas las articulaciones se moverán para que el brazo robótico llegue al punto deseado en menos de un segundo si la posición es menor que a la posición máxima, en otro caso, las articulaciones se mueven para que el brazo robótico llegue en un segundo si la posición es la máxima. Por ejemplo, este algoritmo está pensado para que una articulación llegue como máximo en 1 segundo a 180 grados, por lo que los grados α serán menor o iguales a 180 grados ($\alpha \leq 180$) en sentido horario o antihorario dependiendo el punto que se haya tocado en la pantalla LCD. En este sentido, de la pantalla táctil LCD se puede obtener una coordenada (x, y), sin embargo, es necesario obtener otra dimensión para poder mover el brazo robótico en el espacio, entonces, de las coordenadas (x, y) se puede obtener un tercer dato, el cual es $\alpha = \text{tang}(\frac{y}{x})$, por lo tanto, se puede decir que x es diferente a cero y los números x, y y α serán representados en intervalos de entre cero y 1, entonces, y se encuentra en un conjunto cerrado $[0,1]$, x estará en el conjunto $[0,1]$, y α estará en el intervalo $[0,1]$. Otro punto importante es que las articulaciones son movidas por motores paso a paso, sin embargo, este método también se puede utilizar cuando se tiene otro tipo de motor que use un pulso PWM para el control de su torque, pero tiene mejor rendimiento cuando se utilizan motores paso a paso.

3. Transformación lineal y condiciones

Para representar una coordenada que se toca en la pantalla táctil LCD, se tienen cuatro condiciones, en las cuales los valores de x e y son caracterizados por ser mayor o igual que el otro, es decir, se puede tener el caso donde $x \geq y$ o $y \geq x$ de esta manera se identifican las cuatros condiciones, se construyen cuatro triángulos rectángulos que cubre la pantalla táctil LCD la cual se cumple que $x \geq y$ o $y \geq x$.

Primera condición: En esta condición se obtiene una coordenada (x, y) la cual se obtenga un $\alpha \leq 45$, entonces, construyendo un triángulo rectángulo se obtiene lo siguiente:

$$\alpha \leq 45° \Rightarrow tan^{-1}(\left(\frac{y}{x}\right)) \leq 45 \Rightarrow \frac{y}{x} \leq tan(45) \Rightarrow \frac{y}{x} \leq 1 \quad (1)$$

Para $|x| \geq |y|$, el brazo robótico se mueve en el área del triángulo rectángulo que se muestra en la figura 1, con un x en $(0,1]$, y un y en $[0,1]$, de tal forma que α esté en el intervalo $[0,0.25]$.

Segunda condición: Al obtener la coordenada (x,y), se tiene que $y \geq x$ y se obtiene un $\alpha \geq 45$, con un x en $(0,1]$, y un y en $[0,1]$ tal que α se encuentre en el intervalo $[0.25,0.50]$, esto sucede debido a que el movimiento del brazo robótico es realizado en el área del segundo triángulo rectángulo como se muestra en la figura 1, por lo tanto, para $45 \leq \alpha \leq 90$, y para que $\alpha = 90$, entonces, $\left(\frac{y}{x}\right)$ tiene que alcanzar un valor muy grande, por lo que $\alpha = 90$ cuando $y > 0$ y x se aproxima tanto a 0 tomando en cuenta que x debe ser $x > 0$.

$$\lim_{(x)\to 0^+} \alpha = \lim_{(x)\to 0^+} tan^{-1}\left(\frac{1}{x}\right) = \frac{\pi}{2} = 90 \quad (2)$$

Es decir, existe el límite para cuando x $x$se aproxima a cero, pero no existe $\alpha(0)$ porque se trata de una función divergente que tiene discontinuidad de salto.

Tercera condición: la coordenada (x, y) se posiciona en el área del triángulo rectángulo número 3, es fácil observar porque se obtiene $|y| \geq |x|$, entonces $90 \leq \alpha \leq 135$, por lo que,

$$\alpha = tan^{-1}\left(\frac{y}{x}\right) + 180 \quad (3)$$

Se tiene entonces que x se encuentra en $[-1,0)$, $y$en $[0,1]$, y α en $[0.50,0.75]$.

Cuarta condición: por último, la cuarta condición proporciona la coordenada (x,y) , se encuentra en el área del cuarto triangulo rectángulo, por lo que nuevamente se tiene que $|x| \geq |y|$ y α es definida igual como al caso de la tercera condición pero ahora $135 \leq \alpha \leq 180$ por ser $|x| \geq |y|$, por lo que se tiene que x se localiza en $[-1,0)$, y en $[0,1]$ y α en $[0.75,1]$. A continuación, en la Figura 1 se representan las cuatro condiciones.

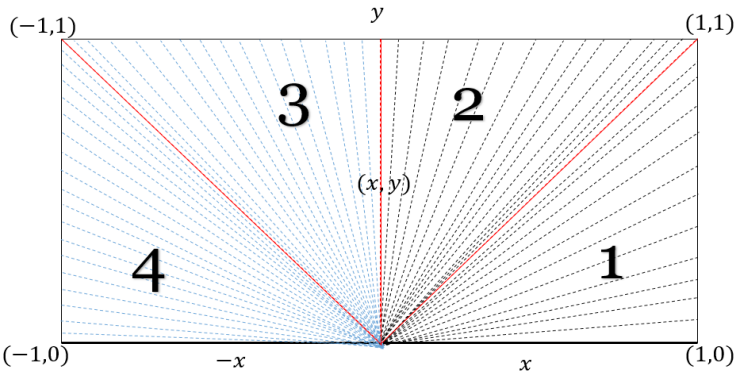

Figura 1 – Representación de las cuatro condiciones, donde el brazo robótico se coloca

dependiendo la ubicación de la coordenada.

Transformación Lineal: En realidad, la figura 1 es la representación de la transformación lineal que se comentó en el apartado de desarrollo del algoritmo. Se sabe que una pantalla táctil LCD es un espacio con dos dimensiones y se necesita mover el brazo robótico en un espacio con 3 dimensiones. Con los datos que hemos definido, tenemos entonces un vector (x, y, α), se puede definir la siguiente transformación, sea $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ y k es cualquier número real. Se define la siguiente transformación lineal para cuando α gira en un sentido antihorario.

$$T\begin{pmatrix} x \\ y \\ \alpha \end{pmatrix} = \begin{pmatrix} kx \\ ky \\ k\alpha \end{pmatrix} \tag{4}$$

Si α tiene que girar en el sentido horario, entonces se utiliza la siguiente transformación lineal.

$$T\begin{pmatrix} x \\ y \\ \alpha \end{pmatrix} = \begin{pmatrix} -kx \\ ky \\ k\alpha \end{pmatrix} \tag{5}$$

La transformación lineal (5) es utilizada cuando la garra del brazo robótico se encuentre en la tercera o cuarta condición y se necesite llegar a la primera o segunda condición (o si la garra está en la cuarta condición, y se necesite llegar a la primera, segunda o tercera condición de manera que α tenga que girar en un sentido horario), entonces α necesitaría rotar de forma en sentido inverso, es decir, las condiciones se intercambian de lugar, la primera condición por la cuarta y la segunda condición por la tercera, tal y como se muestra a continuación:

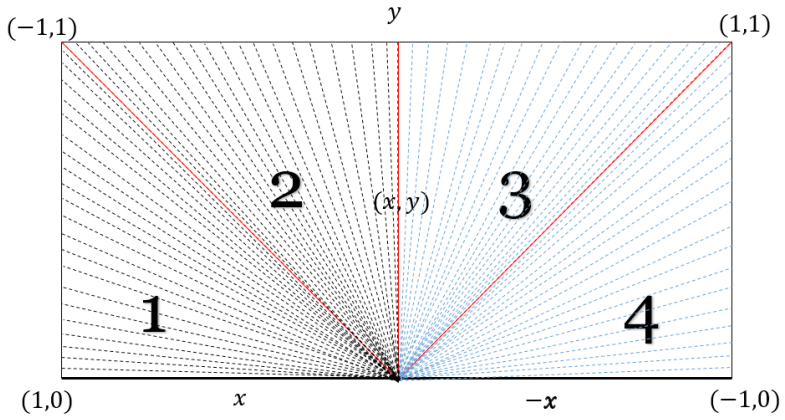

Figura 2 – Representación de las cuatro condiciones y uso de la segunda transformación lineal.

Por ejemplo, se tiene una coordenada inicial $(-x_0, y_0)$ y se encuentra en la cuarta condición de la figura 1, quiere decir que $-x_0 \geq y$, y si se quiere llegar a la tercera condición, entonces

$$T\begin{pmatrix} -x_0 \\ y_0 \\ \alpha \end{pmatrix} = \begin{pmatrix} -k(-x_0) \\ ky_0 \\ k\alpha \end{pmatrix} = \begin{pmatrix} k(x_0) \\ ky_0 \\ k\alpha \end{pmatrix} \tag{6}$$

Entonces $(kx_0, ky_0, k\alpha)$ y tambien $x_0 \leq -y$, luego $|x_0| \leq |-y| = x_0 \leq y_0$, por lo que efectivamente se encuentra el vector en la segunda condición de la segunda transformación lineal representada en la figura 2.

Después, si se tiene una coordenada inicial (x_0, y_0) y se encuentra en la tercera condición vista desde el plano de la figura 1 y se desea llegar a la segunda condición, entonces $y_0 \geq x_0$ y viendo el plano como en la figura 2, entonces se toma la transformación lineal (5)

$$T\begin{pmatrix} x_0 \\ y_0 \\ \alpha \end{pmatrix} = \begin{pmatrix} -kx_0 \\ ky_0 \\ k\alpha \end{pmatrix} = \begin{pmatrix} -kx_0 \\ ky_0 \\ k\alpha \end{pmatrix} \tag{7}$$

Se tiene que $(-kx_0, y_0, k\alpha)$ y se sabe que $y \geq x_0$, por lo que viendo al nuevo vector se tiene que $y \geq x_0 \geq -x_0$, luego $|y| \geq |x_0|$, teniendo entonces que $y_0 \geq x_0$, por lo que el vector se encuentra en la tercera condición de la figura 2 y en la segunda condición visto desde el plano de la Figura 1.

Es muy fácil demostrar que las transformaciones son lineales ya que cumple con las condiciones para que una transformación sea lineal, primero se seleccionan dos vectores:

$$v_1\begin{pmatrix} x_1 \\ y_1 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \\ \alpha_2 \end{pmatrix} \tag{8}$$

Entonces, se prueba la primera condición:

$$T(v_1 + v_2) = T\left(\begin{pmatrix} x_1 \\ y_1 \\ \alpha_1 \end{pmatrix} + \begin{pmatrix} x_2 \\ y_2 \\ \alpha_2 \end{pmatrix}\right) = \begin{pmatrix} k(x_1 + x_2) \\ k(y_1 + y_2) \\ k(\alpha_1 + \alpha_2) \end{pmatrix} = \begin{pmatrix} kx_1 \\ ky_1 \\ k\alpha_1 \end{pmatrix} + T\begin{pmatrix} kx_2 \\ ky_2 \\ k\alpha_2 \end{pmatrix} = T(v_1) + (v_2) \tag{9}$$

Probando la segunda condición, tomando a λ en el conjunto de los reales, entonces:

$$\lambda T(v_1) = \lambda T\begin{pmatrix} x_1 \\ y_1 \\ \alpha_1 \end{pmatrix} = \lambda \begin{pmatrix} kx_1 \\ ky_1 \\ k\alpha_1 \end{pmatrix} = \begin{pmatrix} \lambda kx_1 \\ \lambda ky_1 \\ \lambda k\alpha_1 \end{pmatrix} = T\left(\begin{pmatrix} \lambda x_1 \\ \lambda y_1 \\ \lambda \alpha_1 \end{pmatrix}\right) = T\left(\lambda \begin{pmatrix} x_1 \\ y_1 \\ \alpha_1 \end{pmatrix}\right) = T(\lambda v_1) \tag{10}$$

Por lo que *T* es una transformación lineal, de la misma manera se demuestra para la transformación lineal (5).

4. Desarrollo del Algoritmo

Utilizando la pantalla táctil LCD, se intenta mover tres motores paso a paso para llegar a las posiciones correspondientes dada por la transformación lineal, cada motor tiene una resolución de 1.8 grados por paso la cual se alcanza cuando se le aplica a la bobina del motor un pulso PWM de 10 milisegundos. Se nombra al motor α como el motor de la primera articulación, este motor hace girar al brazo robótico en el sentido horario y anti horario en el eje vertical paralelo al brazo robótico, después se define el motor x el cual se mueve a través del eje x de la figura 1, y por último el motor y se moverá a través del eje y. Para ver mejor la forma en la que se mueve el brazo robótico, se utilizó como ejemplo el brazo, el motor α representa a los movimientos que hace el hombro de manera vertical, el motor y representa los movimientos de reflexión y contracción del codo, por último, el motor x representa los movimientos del hombro de manera horizontal, posicionando nuestra mano en un punto en el espacio. Se sabe que la resolución del motor es de 0° grados y que se necesita un pulso PWM para mover un paso de 1.8 grados, entonces, los grados obtenidos son por el número de pasos de 10 milisegundos, por lo que se construye la siguiente función:

$$f(n) = 1.8n \tag{11}$$

donde n es el número de pulsos PWM con grosor de 10 milisegundos, quiere decir que, para alcanzar 180 grados, se necesitarán 100 pulsos, por lo tanto, se alcanzan en 1 segundo. La función discreta (11) funciona de una forma correcta, de hecho, es una función precisa, sin embargo, al momento de utilizar la pantalla LCD y mandar los pulsos PWM se debe hacer por lo menos una pausa de 5 milisegundos para poder mandar otro pulso de 10 milisegundos, de no ser así, puede hacer que las uniones del brazo robótico no lleguen a la posición deseada, pues se genera un tráfico en cada unión del brazo robótico. Para llegar 180 grados se necesita más de un segundo para llegar, entonces, se propone la siguiente función, la cual puede alcanzar una cantidad muy grande con un solo pulso.

$$f(s) = \sum_{n=0}^{s} 1.8n \tag{12}$$

Esta función incrementa la anchura del pulso dependiendo el valor de s, esta función es muy fácil de programar y aunque no es muy precisa, puede ser combinada con la función (11) para obtener un valor preciso. El límite superior de la sumatoria de la función (12) puede tomar los siguientes valores $s = \{1,2,3,4,5,6,7,8,9,10,11,12,13\}$, así que mientras n incrementa, el grosor del pulso incrementa, tal y como se muestra en la siguiente tabla.

$f(s)$	s	Tiempo
1.8	1	10 ms
5.4	2	20 ms
10.8	3	30 ms
18	4	40 ms
27	5	50 ms

$f(s)$	s	Tiempo
37.8	6	60 ms
50.4	7	70 ms
64.8	8	80 ms
81	9	90 ms
99	10	100 ms
118.8	11	110 ms
140.4	12	120 ms
163.8	13	130 ms

Tabla 1 – grosor de los pulsos usando la función $f(s)$.

4.1. Representación del Algoritmo

Hasta el momento, se han presentado las dos funciones discretas con las que el algoritmo trabaja, en resumen, el algoritmo combina las funciones discretas para alcanzar los grados deseados utilizando la menor cantidad de pulsos o grosor de los pulsos, luego, el número de combinaciones con la que se puede usar las funciones es 2^n formas, cada manera puede tener diferente tiempo y algunas veces algunas combinaciones pueden tener la misma cantidad de tiempo para alcanzar los grados deseados. Además, se debe saber que algunas combinaciones pueden ser más exactas que otras, pero más lentas en alcanzar los grados deseados. Sin embargo, la mejor combinación es encontrada por el algoritmo y puede anticipar la localización del tiempo de la mejor combinación.

$$a \geq \lim_{t_s \to b} \frac{2^n}{t_s} \geq a - b \tag{13}$$

El número a representa la cantidad de los grados deseados, el número b ayuda a tener el intervalo para encontrar los límites de tiempo (límite superior o inferior) y ayuda a que la inecuación se cumpla, el número 2^n es la cantidad de combinaciones en las que se encuentra nuestra combinación a buscar. Para desarrollar el algoritmo, es importante saber qué $a < 2^n$ y b puede tener valores menores o iguales a $(2^n/a)$, también, puede ser redondeado mientras la inecuación se cumpla. Entonces, cuando se encuentra el valor de b, se iguala a la división del tiempo límite superior entre el tiempo límite inferior, tal y como se muestra a continuación:

$$b = \left(\frac{t_p}{t_d}\right) \tag{14}$$

Donde t_p es el tiempo límite superior y t_d es el tiempo límite inferior. Cuando se sabe los límites de tiempo, se puede estimar el tiempo que le corresponde a la combinación que alcanza los grados deseados.

$$t_p \geq t \geq t_d \tag{15}$$

Encontrar el número b puede ser difícil cuando el número deseado es muy grande, sin embargo, se tiene que poner atención a los límites de tiempo del comportamiento de la inecuación para cuando el número n es repetitivo, esto quiere decir que el límite superior esta cera al tiempo medio o máximo del proceso (500 milisegundos o 1000 milisegundos). Por ejemplo, un ejemplo fácil y claro es para cuando se toma un $\alpha = 16.2$, entonces:

$$16.2 \geq \lim_{t_s \to b} \frac{2^n}{t_s} \geq 16.2 - b \tag{16}$$

Entonces n debe ser igual a 5, debido a que $16.2<2^5$ y $b \leq \left(\frac{2^5}{16.2}\right) = 1.9753 \ldots$, pero b puede tener el valor 2, ya que la inecuación se cumple, entonces:

$$16.2 \geq \lim_{t_s \to 2} \frac{2^5}{t_s} \geq 14.2 \tag{17}$$

Como $b = (\frac{t_p}{t_d})$ y $n = 5$, se puede decir que el tiempo limite inferior es 50 milisegundos, entonces:

$$2 = (\frac{t_p}{50ms}) \Rightarrow t_p = 100ms \tag{18}$$

Por lo que la combinación se encuentra para cuando $n = 5$ con 32 formas de combinar las dos funciones discretas en un tiempo entre 50 milisegundos y 100 milisegundos. En la Figura 3, se muestra como se construye la combinación:

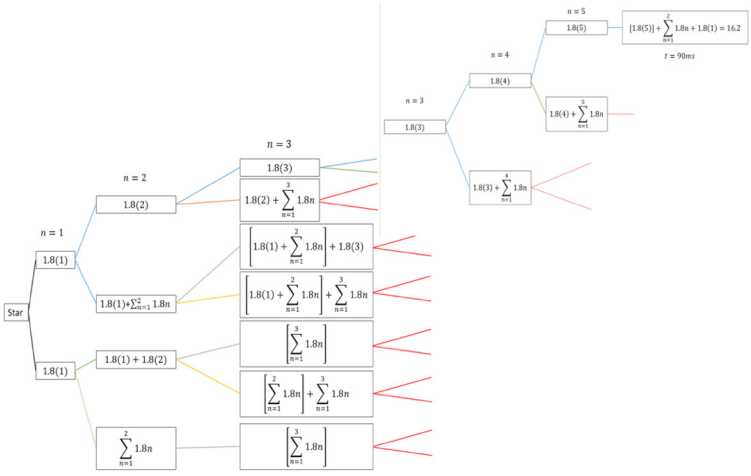

Figura 3 – Desarrollo del algoritmo para cuando $\alpha = 16.2$, los caminos de color rojo son los que fueron rechazados al momento de hacer las combinaciones.

El resultado es $[1.8(5)] + \sum_{n=1}^{2} 1.8(1) = 16.2$ en un tiempo de 90 milisegundos.

4.2. Simulación del Algoritmo en Matlab

Se ha seleccionado Matlab para hacer la simulación del algoritmo, se realizó la trayectoria del brazo robótico haciendo las combinaciones de las funciones discretas presentadas. Se ha utilizado una interfaz gráfica de usuario tal y como se muestra en la Figura 4. Colocando los grados en los que se quiere mover cada junta y oprimiendo el push button llamado "Get position" el algoritmo realiza la trayectoria y trabaja simultáneamente con la transformación lineal presentada para obtener la cinemática inversa del brazo robótico y se muestra una representación del brazo robótico con las tres articulaciones haciendo la trayectoria. Además, se realiza el cálculo del número de combinaciones, tiempo límite superior, tiempo límite superior sin los tiempos de espera agregados por cada pulso de 10 milisegundos de la primera función discreta, tiempo límite inferior y los grados que se obtienen tomando el tiempo límite superior y el tiempo límite inferior.

La obtención de la trayectoria está inspirada en el método de polinomios cúbicos, en este caso, los grados en el tiempo límite inferior son calculados usando los polinomios cúbicos y tomando el tiempo límite inferior para aproximar los grados que se pueden obtener en caso de usar dicho límite. Se presenta en este apartado un ejemplo para cuando el motor α (primera articulación) se mueva a 45 grados. En resumen, el programa calcula los grados que corresponden al tiempo límite superior y el tiempo límite inferior con el método de polinomios cúbicos para comparar los resultados con los resultados del algoritmo propuesto.

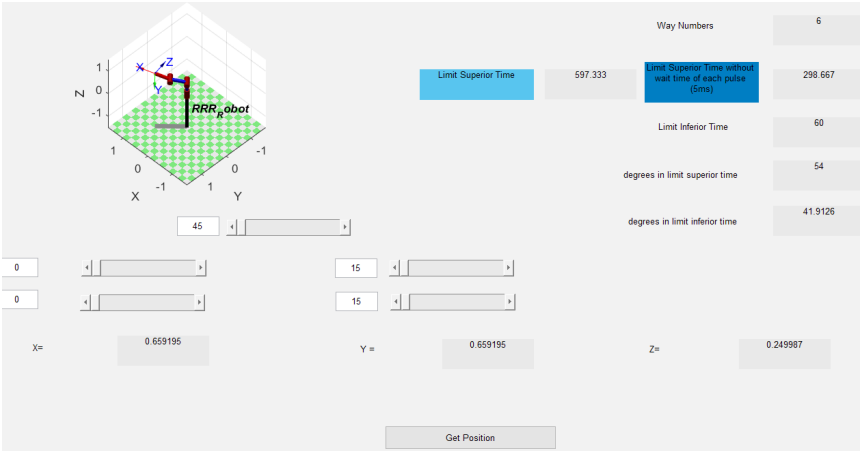

Figura 4 – Simulación del Algoritmo y aplicación de la transformación lineal para representar la trayectoria y la cinemática directa e inversa del brazo robótico.

El eje Z representa al eje α que se mencionó en la transformación lineal. En la interface se puede observar que la coordenada (x, y) igual a (0.659195,0.659195) debido a que $\alpha=45$ grados, y aunque en realidad se pide que en (x, y) se muevan las articulaciones en 15 grados, la transformación lineal ha convertido los 15 grados en $x = 1/12 = y$, después, al momento de transformar el vector (x, y, α), se obtiene el nuevo vector $(kx, ky, k\alpha)$ y para este caso, $k = 8$ ya que para obtener k se debe igualar a la distancia sobre el eje X

o eje Y (en este caso las distancias de los dos eslabones se encuentran en el eje X) de los eslabones que se encuentran fuera del origen (0,0,0) y cada uno tiene de distancia igual a 4 unidades, entonces $k = 8$, también en este ejemplo, la k que multiplica a α se encuentra en el origen, entonces se dice que $k = 1$.

5. Resultados

Los resultados están enfocados en la simulación en MATLAB y en el código de Arduino generado mediante la simulación en MATLAB, se utilizó una pantalla táctil LCD modelo 2.4 TFT y motores paso a paso Nema 17, los resultados se presentan como los valores deseados de α obtenida al tocar la pantalla LCD comparando los con los resultados de la simulación en MATLAB y la ejecución del código de Arduino. Las Tablas 2, 3, 4 y 5 representan a los resultados obtenidos tocando la pantalla LCD para las 4 condiciones.

α	$\left(\frac{y}{x}\right)$	n	$f(n) = 1.8n$	s	$f(s) = \sum_{n=0}^{s} 1.8n$
5.71	0.1	3	5.4	2	5.4
11.3	0.2	6	10.8	3	10.8
16.69	0.3	9	16.2	3	10.8
18	0.35	12	21.6	4	18
21.8	0.4	12	21.6	4	18
26.56	0.5	14	25.2	4	18
27	0.5095	14	25.2	5	27
30.96	0.6	14	25.2	5	27
34.99	0.7	17	30.6	5	27
37.8	0.7756	19	34.2	6	38.7
38.65	0.8	21	37.8	6	38.7
41.98	0.9	23	41.4	6	38.7
45	1	25	45	6	38.7

Tabla 2 – número de pulsos que $f(s)$ y $f(n)$ necesitan para aproximarse a α en la primera condición.

Entonces, se tiene la siguiente función la cual representa la mejor opción para usarse en esta primera condición.

$$f(s,n) := \begin{cases} \sum_{n=0}^{s} 1.8n \;\; si\; 0 \leq \left(\frac{y}{x}\right) \leq 0.35, con\; s = \{1,2,3,4\}, si\; 0.51 \leq \left(\frac{y}{x}\right) \leq 0.79, s = \{5,6\} \\ 1.8n \;\; si\; 0.4 \leq \left(\frac{y}{x}\right) \leq 0.5, si\; 0.6 \leq \left(\frac{y}{x}\right) \leq 0.7 \\ 1.8n \;\; si\; 0.8 \leq \left(\frac{x}{y}\right) \leq 1 \end{cases} \quad (19)$$

α	$\left(\frac{y}{x}\right)$	n	$f(n) = 1.8n$	s	$f(s) = \sum_{n=0}^{s} 1.8n$
45	1	25	45	8	64.8
63.434	2	35	63	8	64.8
75.963	4	42	75.6	8	64.8
80.53	6	44	79.2	8	64.8
81	6.313	45	81	9	81
82.87	8	46	82.8	9	81
84.298	10	47	84.6	9	81
87.137	20	48	86.4	9	81
88.56	40	49	88.2	9	81
88.85	50	49	88.2	9	81
89.427	100	49	88.2	9	81
89.93	900	49	88.2	9	81

Tabla 3 – número de pulsos que $f(s)$ y $f(n)$ que necesitan para aproximarse a α en la segunda condición.

Entonces, la combinación de las funciones es representada como:

$$f(n,s) := \begin{cases} 1.8n \; si \; 1 \leq \left(\frac{y}{x}\right) \leq 1.99 \\ \sum_{n=0}^{s} 1.8n \;\; si \; 2 \leq \left(\frac{y}{x}\right) \leq 2.2 \; or \; si \; 6 \leq \left(\frac{y}{x}\right) \leq 8 \\ 1.8n \;\; si \; 2.3 \leq \left(\frac{y}{x}\right) \leq 5.9 \; o \; si \; 8.1 \leq \left(\frac{y}{x}\right) \leq 1000 \end{cases} \quad (20)$$

α	$\left(\frac{y}{x}\right)$	n	$f(n) = 1.8n$	(s,n)	$f(s,n) = f(s) + f(n)$
91.145	-50	50	90	(9,0)	81
91.6	-30	51	91.8	(9,0)	81
92.29	-25	51	91.8	(9,0)	81
92.86	*-20*	52	93.6	(9,0)	81
93.814	*-15*	52	93.6	(9,0)	81
95.71	*-10*	53	95.4	(9,0)	81
96.34	*-9*	54	97.2	(10,0)	99
97.125	*-8*	54	97.2	(10,0)	99
98.13	*-7*	55	99	(10,0)	99
99	*-6.313*	55	99	(10,0)	99

α	$\left(\frac{y}{x}\right)$	n	$f(n) = 1.8n$	(s,n)	$f(s,n) = f(s) + f(n)$
99.462	-6	56	100.8	(10,1)	100.8
100.8	-5.242	56	100.8	(10,1)	100.8
101.309	-5	56	100.8	(10,1)	100.8
102.6	-4.0473	57	102.6	(10,2)	102.6
104.036	-4	58	104.4	(10,3)	104.4
108	-3.0077	60	108	(10,5)	108
108.434	-3	60	108	(10,5)	108
115.2	-2.125	64	115.2	(10,9)	115.2
116.565	-2	65	117	(10,10)	117
133.2	-1.064	74	133.2	(10,19)	133.2
135	-1	75	135	(10,20)	135

Tabla 4 – número de pulsos que $f(s)$ y $f(n)$ que necesitan para aproximarse a α en la tercera condición.

Se obtiene la siguiente función:

$$f(s,n) := \begin{cases} 1.8n \;\; si - 50 \leq \left(\frac{y}{x}\right) < -7 \\ \sum_{n=0}^{s} 1.8n + 1.8n \; si - 7 \leq \left(\frac{y}{x}\right) \leq -1 \; con \; s = 10 \; y \; 0 < n \leq 20 \end{cases} \quad (21)$$

α	$\left(\frac{y}{x}\right)$	n	$f(n) = 1.8n$	(s,n)	$f(s,n) = f(s) + f(n)$
-0.9999	135.01	75	135	(10,20)	135
-0.99	135.28	75	135	(10,20)	135
-0.9	138.012	77	138.6	(10,20)	135
-0.85	139.012	78	140.4	(10,20)	135
-0.8	141.34	79	142.2	(10,20)	135
-0.75	143.036	80	144	(10,20)	135
-0.7	145.007	81	145.8	(10,20)	135
-0.6	149.036	83	149.4	(10,20)	135
-0.5	153.43	85	153	(10,20)	135
-0.4	158.198	87	156.6	(10,20)	135
-0.3	163.3	90	162	(10,20)	135
-0.2905	163.8	91	163.8	(13,0)	168.8
-0.2	168.69	94	169.2	(13,3)	169.2
-0.158	171	95	171	(13,4)	171

α	$\left(\frac{y}{x}\right)$	n	$f(n) = 1.8n$	(s,n)	$f(s,n) = f(s) + f(n)$
-0.1	174.28	97	174.6	(13,6)	174.6
-0.031	178.2	99	178.2	(13,8)	178.2
-1	180	100	180	(13,9)	180

Tabla 5 – número de pulsos que $f(s)$ y $f(n)$ que necesitan para aproximarse a α en la cuarta condición.

Los resultados de $f(s)$la función son obtenidos mezclando $f(s)$ y $f(n)$ es muy preciso y mas rapido que solo usando $f(s)$ o $f(n)$, en este caso $f(s,n)$ se mantiene en 135 mientras (y/x) es menor que -0.2905. La Figura 5, se muestra el resultado de cada función y la aproximación de la función $f(n)$ y $f(s)$, se puede observar como la combinaciones pueden ser construidas y cuales pueden ser usadas. Por último se obtiene la función para la cuarta condición:

$$f(s,n) := \begin{cases} 1.8n \ si \ 0.999 \leq \left(\frac{y}{x}\right) < -0.3 \\ \sum_{n=0}^{s} 1.8n \ si - 0.29 \leq \left(\frac{y}{x}\right) \leq -0.25 \\ \sum_{n=0}^{s} 1.8n + 1.8n' \ si - 0.25 < \left(\frac{y}{x}\right) \leq -1 \quad con \ s = 13 \ y \ 0 < n' \leq 9 \end{cases} \tag{22}$$

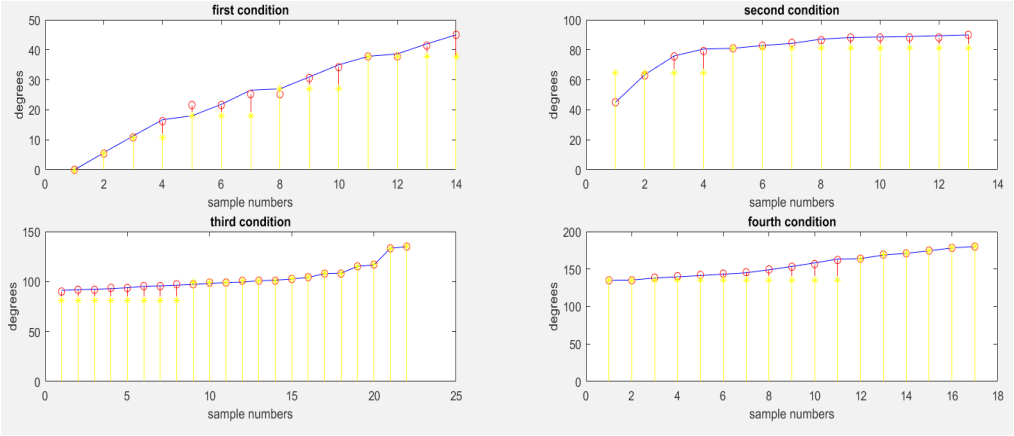

Figura 5 – Resultados de la trayectoria en cada condición en Arduino y en Matlab.

6. Conclusiones

Se ha analizado un método para mejorar el control de un brazo robótico antropomórfico donde se puede calcular fácilmente su cinemática inversa o cinemática directa, además, se puede generar una trayectoria combinando las dos funciones discretas, esto es lo más interesante, debido a que de esta forma se puede utilizar una pantalla táctil para que un paciente con discapacidades/dificultades físicas pueda controlar un brazo robótico tan solo tocar con un solo dedo la coordenada de la pantalla táctil LCD. Entonces, usando un espacio con dos dimensiones, es posible generar una trayectoria en un espacio con tres dimensiones, como se mostró anteriormente, esto se lleva a cabo con una transformación lineal, la cual obtiene una coordenada en el espacio con dos dimensiones, después, se selecciona la condición donde se posiciona coordenada, y de esta forma, saber cuántos grados se debe mover cada articulación y la posición final en el eje X, eje Y y eje α, la cual se representa como el vector de llegada de la transformación lineal. Después, se usa el algoritmo propuesto para traducir los datos que la transformación lineal ha proporcionado, entonces, se genera la trayectoria del brazo robótico moviendo motores paso a paso de cada articulación, sin embargo, el brazo robótico puede tardar como máximo 1.5 segundos, esto hace que los movimientos del brazo robótico sean muy rápidos.

Sería interesante controlar el tiempo de la trayectoria del brazo robótico, sin embargo, con este método no es posible, entonces, para futuros trabajos, se busca encontrar una función que genere la trayectoria del brazo robótico controlando el tiempo en el que el brazo robótico llega a su partida y que además pueda ser fácil de implementar usando diferentes motores y reduciendo el número de combinaciones. Se espera que con este algoritmo y la transformación lineal sea más fácil diseñar una función en la que se pueda utilizar cualquier pantalla táctil LCD y usando cualquier tipo de motor en las articulaciones del brazo robótico antropomórfico para generar la trayectoria. Además, al encontrar una nueva función, será posible que sea mucho más fácil de programarse, más rápido de ejecutar el programa, mejorar el consumo de memoria, y por lo tanto, la ejecución será más fácil y rápida.

Referencias

Allen, S. (2016). New Prostheses and Orthoses Step Up their Game: Motorized Knees, Robotic Hands, and Exosuits Mark Advances in Rehabilitation Technology. IEEE Pulse, 7(3), 6-11.

Atre, P., Bhagat, S., Pooniwala, N., & Shah, P. (2018). Efficient and Feasible Gesture Controlled Robotic Arm. *In: Proceeding of Second International Conference on Intelligent Computing and Control Systems*, (pp. 1—6), Madurai, India: IEEE Publishing.

Badrinath, A.S., Vinay, P.B., & Hegde, P. (2016). Computer Vision based semi-intuitive Robotic arm. *In: Proceeding of 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics*, pp. 563-567. Chennai, India: IEEE Publishing.

Beckerle, P., Kõiva, R., Kirchner, E. A., Bekrater-Bodmann, R., Dosen, S., Christ, O., Abbink, D. A., Castellini, C., & Lenggenhager, B. (2018). Feel-Good Robotics: Requirements on Touch for Embodiment in Assistive Robotics. Frontiers in Neurorobotics, 12, 1-84.

Bovet, S., Debarba, H. G., Herbelin, B., Molla, E., & Boulic, R. (2018). The Critical Role of Self-Contact for Embodiment in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics,* 24(4), 1428-1436.

Burgner-Kahrs, J., Rucker, D.C., & Choset, H. (2015). Continuum Robots for Medical Applications: A Survey. IEEE Transactions on Robotics and Automation, 31(6), 1261-1280.

Chung, C.S., Wang, H., & Cooper, R.A. (2013). Functional assessment and performance evaluation for assistive robotic manipulators: Literature review. The Journal of Spinal Cord Medicine, 36(4), 273-289.

Fernández-Ordoñez, J., Maza, L., Torres-Carrión, P., Barba-Guzmán, L., & Rodríguez-Morales, G. (2019). Experiencia Afectiva Usuario en ambientes con Inteligencia Artificial, Sensores Biométricos y/o Recursos Digitales Accesibles: Una Revisión Sistemática de Literatura. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, (35), 35-53. DOI: 10.17013/risti.35.35–53.

Grau, A., Indri, M., Bello, L.L., & Sauter, T. (2017). Industrial robotics in factory automation: From the early stage to the Internet of Things. *In: Proceeding of 43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 6159-6164. Beijing, China: IEEE publishing.

Jiang, H., Wachs, J.P., & Duerstock, B.S. (2012). Facilitated Gesture Recognition Based Interfaces for People with Upper Extremity Physical Impairments. In: L. Alvarez, M. Mejail, L. Gomez, J. Jacobo (eds.). *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 228—235. Berlin, Heidelberg: Springer.

Kokkinara, E., Slater, M., & López-Moliner, J. (2015). The effects of visuomotor calibration to the perceived space and body through embodiment in immersive virtual reality. ACM Transactions on Applied Perception, 13(1), 1-22.

Levin, M.F., Weiss, P.L., & Keshner, E.A. (2015). Emergence of virtual reality as a tool for upper limb rehabilitation: incorporation of motor control and motor learning principles. Physical Therapy, 95(3), 415-425.

Kruthika, K., Kumar, B.M.K., & Lakshminarayanan, S. (2016). Design and development of a robotic arm. *In: Proceeding of International Conference on Circuits, Controls, Communications and Computing*, pp. 1-4. Bangalore, India: IEEE publishing.

Lara, G., Santana, A., Lira, A., & Peña, A. (2019). El Desarrollo del Hardware para la Realidad Virtual. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, (31), 106-117. DOI: 10.17013/risti.31.106–117.

Makin, T., de Vignemont, F., & Faisal, A. (2017). Neurocognitive barriers to the embodiment of technology. Nature Biomedical Engineering, 1(0014), 1-3.

Murali, A., Sen, S., Kehoe, B., Garg, A., Mcfarland, S., Patil, S., ... Goldberg, K. (2015). Learning by observation for surgical subtasks: Multilateral cutting of 3D viscoelastic and 2D Orthotropic Tissue Phantoms. In: *Proceeding of* International Conference on Robotics and Automation, pp. 1202-1209. Seattle, WA, USA: IEEE Publishing.

Niyetkaliyev, A.S., Hussain, S., Ghayesh, M.H., & Alici, G. (2017). Review on Design and Control Aspects of Robotic Shoulder Rehabilitation Orthoses. IEEE Transactions on Human-Machine Systems, 47(6), 1134-1145

Perez-Marcos, D., Chevalley, O., Schmidlin, T., Garipelli, G., Serino, A., Vuadens, P., Tadi, T., Blanke, O., & Millán, J.D. (2017). Increasing upper limb training intensity in chronic stroke using embodied virtual reality: a pilot study. Journal of NeuroEngineering and Rehabilitation, 14(1), pp. 119.

Proietti, T., Crocher, V., Roby-Brami, A., & Jarrassé, N. (2016). Upper-Limb Robotic Exoskeletons for Neurorehabilitation: A Review on Control Strategies. IEEE Reviews in Biomedical Engineering, 9, 4-14,

Rehmat, N., Zuo, J., Meng, W., Liu, Q., Xie, S.Q., & Liang, H. (2018). Upper limb rehabilitation using robotic exoskeleton systems: a systematic review. International Journal of Intelligent Robotics and Applications, 2(3), 283-295.

Shademan, A., Decker, R.S., Opfermann, J.D., Leonard, S., Krieger, A., & Kim, P.C. (2016). Supervised autonomous robotic soft tissue surgery. Science Translational Medicine, 8(337), 337ra64.

Yenorkar, R., & Chaskar, U. M. (2018). GUI Based Pick and Place Robotic Arm for Multipurpose Industrial Applications. In: *Proceeding of the Second International Conference on Intelligent Computing and Control Systems*, (pp. 200-203). Madurai, India: IEEE publishing.

Young, A.J., & Ferris, D.P. (2017). State of the Art and Future Directions for Lower Limb Robotic Exoskeletons. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 25(2), 171-182.