UNIVERSIDAD DEL BIO-BIO
FACULTAD DE INGENIERIA
DEPTO. DE INGENIERIA ELECTRICA Y ELECTRONICA

UNIVERSIDAD DEL BÍO-BÍO

"SISTEMA DE GUIA DINAMICO PARA BRAZO ROBOT SCORBOT ER-IX MEDIANTE VISION ARTIFICIAL"

AUTOR: **CESAR ENRIQUE SOTO MONTECINOS.**

SEMINARIO PARA OPTAR AL TITULO DE
INGENIERO DE EJECUCION EN ELECTRONICA

CONCEPCION – CHILE
2006

UNIVERSIDAD DEL BIO-BIO
FACULTAD DE INGENIERIA
DEPTO. DE INGENIERIA ELECTRICA Y ELECTRONICA

UNIVERSIDAD DEL BÍO-BÍO

"SISTEMA DE GUIA DINAMICO PARA BRAZO ROBOT SCORBOT ER-IX MEDIANTE VISION ARTIFICIAL"

AUTOR: **CESAR ENRIQUE SOTO MONTECINOS.**

DOCENTE PATROCINANTE: **CRISTHIAN AGUIERA CARRASCO.**

A mis padres,
que toda mi vida
me han apoyado
de manera incondicional
y con muchísimo amor,
que siempre han sustentado
todos mis sueños
y anhelos.

A mis hermanos,
que sin querer
me han permitido ver y entender
cosas que sólo no era capaz.

A mis familiares y amigos
que me ayudaron siempre
cuando los necesité.

A todos ellos,
les estoy eternamente
agradecido.

INDICE

INTRODUCCION

Actualmente, el laboratorio CIMUBB de la Universidad del Bío-Bío cuenta con varios brazos robots, que son programables o manipulables mediante dos formas: Una, es por medio de un computador a través de instrucciones en lenguaje ACL; y la otra, por medio de un control externo llamado “Teach Pendant”.

Estos métodos de programación no permiten una manipulación dinámica o natural de robot tal como lo hacemos con nuestros brazos. Ocurriendo frecuentemente errores en la programación de trayectorias que finalmente concluyen en colisiones.

Así pues, con la intención de disponer de una forma más robusta, flexible y dinámica de realizar tal programación, se desarrolló una aplicación que permite al usuario manipular objetos a través de un robot Scorbot ER IX que es guiado dinámicamente a través de movimientos realizados por la mano de un usuario, movimientos que se capturan mediante de cámaras, que posteriormente realimentarán la posición del robot.

El Capitulo I da una visión general del sistema y una breve explicación en que consiste.

El Capítulo II describe temas relacionados con Visión Artificial, referente a la captura de movimientos mediante cámaras web y el procesamiento de las imágenes adquiridas. Así como también, la obtención de datos de estas imágenes procesadas.

El Capítulo III trata sobre el Modelo Cinemático del robot, utilizado para la resolución de posiciones angulares de cada articulación mediante la Cinemática Inversa. Además, trata sobre la obtención de ciertos parámetros del robot.

El Capítulo IV se enfoca principalmente a la comunicación entre el programa desarrollado en el PC y el controlador del robot.

RESUMEN

El presente informe de seminario da a conocer la forma en que se desarrollo e implementó un sistema de Guía Dinámico para un Robot Scorbot ER-IX mediante Visión Artificial.

Este sistema, basado en un computador PC-compatible, consiste en capturar los movimientos realizados por la mano de un usuario mediante la utilización de cámaras a través de algoritmos de Visión Artificial. Para llevar a cabo tal captura de movimientos, el usuario utiliza un *Guante Señalizador* que consta de marcas dispuestas estratégicamente en distintas partes del guante. Los algoritmos de Visión Artificial permiten procesar las imágenes adquiridas por las cámaras y obtener las posiciones espaciales de las marcas del Guante Señalizador.

Una vez obtenidas las posiciones espaciales de las marcas, el sistema reconstruye el Guante Señalizador en forma 3D mediante Triangularización y Proyección Inversa de puntos. Datos que posteriormente van a un modelo cinemático inverso del robot que permite la resolución de las posiciones que debe tomar para poder llegar a la posición especificada por el usuario.

Una vez resuelta la posición que debe tomar el robot, el programa basado en PC transmite a otro programa residente en el controlador del robot, mediante la puerta serial RS-232C, las posiciones angulares físicas que debe tomar el robot para alcanzar la posición espacial indicada por el usuario.

Como complemento, el sistema dispone de una representación grafica 3D que sirve al usuario de realimentación visual para saber como dispondrán las distintas articulaciones del robot antes de moverse a la posición espacial deseada.

CAPITULO I: *El Sistema de Guía Dinámico*

El Sistema de Guía Dinámico es un sistema que permite al usuario operar un robot tipo Eshed Scorbot ER-IX mediante los movimientos naturales de su mano. Esquemáticamente, el sistema se divide en cuatro módulos principales: *Visón Artificial, Modelo Cinemático, Comunicación, y Representación Gráfica*. La figura 1.1 muestra el esquema mencionado.

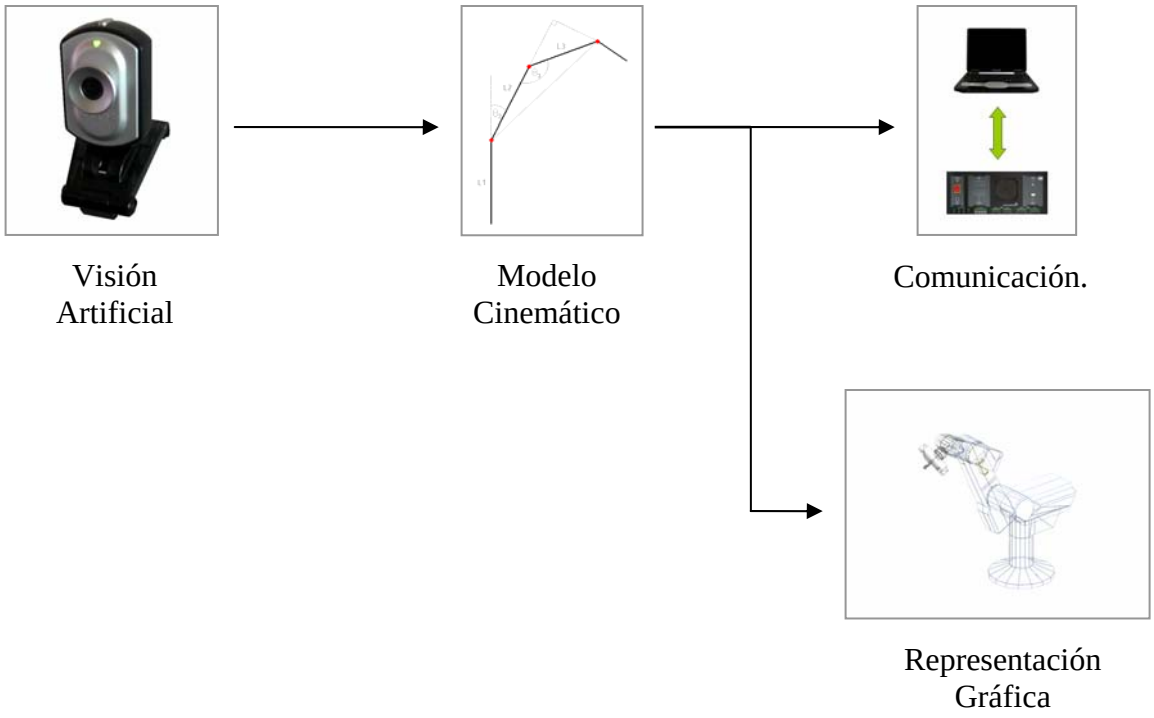
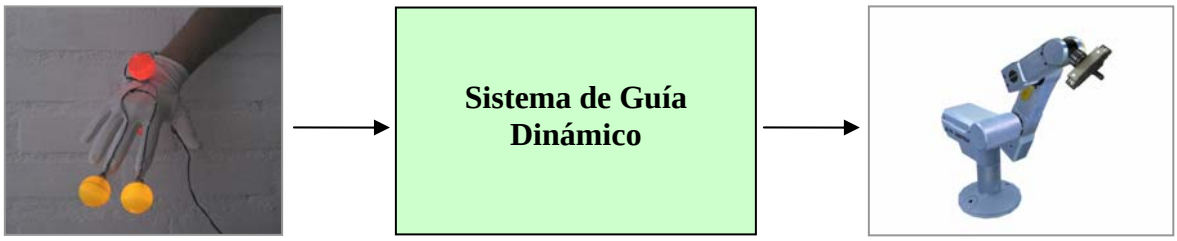


Figura 1.1

El bloque de *Visión Artificial* es el encargado de detectar, procesar y obtener todos los datos relacionados con la mano del usuario mediante la utilización de cámaras Web como sensores de movimiento. El bloque *Modelo Cinemático* toma las variables generadas por el módulo de *Visión Artificial* y determina los ángulos que debe tomar cada articulación del robot para alcanzar la posición espacial indicada por la mano del usuario. *Comunicación* se encarga de establecer y mantener un enlace con el controlador del robot

para enviarle los ángulos de cada articulación por medio del puerto de comunicaciones serie RS-232C. Finalmente, tenemos el bloque *Representación Gráfica* que genera una imagen tridimensional y en tiempo real de cómo se posicionarán todas las articulaciones de robot antes de que éste se mueva.

Desde una perspectiva más general, el sistema interactúa con el usuario y el robot como se muestra en la figura 1.2.



Mano del usuario Brazo Robot

Figura 1.2

CAPITULO II: *Visión Artificial*

Como se ha explicado anteriormente, éste bloque esta orientado a detectar, procesar y obtener todos los datos relacionados con la mano del usuario. Entre estos, tenemos la posición actual, inclinación o *pitch* y la detección de apertura o cierre de los dedos, siendo la posición actual el dato más importante. Debido a la gran cantidad de posiciones espaciales que puede adoptar una mano y con la intención de liberar un poco al usuario de complicadas estructuras de monitoreo de movimientos, se optó por utilizar un sistema *Visión Artificial* basado en cámaras Web para la obtención de los datos mencionados anteriormente. Sin embargo, la obtención de dichos datos no es sencilla ya que generalmente existen otras variables físicas que interfieren en la lectura de las variables de interés. A esto, se suman las limitantes propias de los sensores y el coste computacional de los algoritmos de procesamiento de imágenes. Para compensar estos problemas, se construyó un sencillo guante con marcas auto-iluminadas, llamado desde ahora *Guante Señalizador*, que facilita al sistema de *Visión Artificial* la obtención de los datos de la mano del usuario.

Algorítmicamente, todo el procesamiento de información que realiza este bloque se muestra en la figura 2.1.

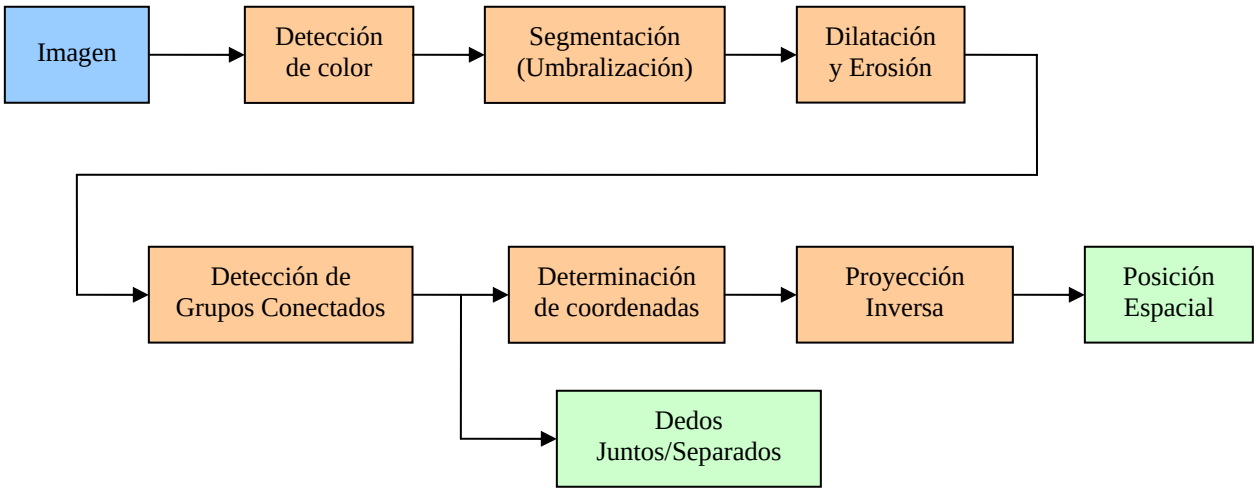


Figura 2.1

2.1 El Guante Señalizador

Como se explicó anteriormente, el objetivo del *Guante Señalizador* es facilitar la lectura de las posiciones de la muñeca y los dedos del usuario, y de otras variables. La figura 2.2 muestra una fotografía del guante con su fuente de alimentación.

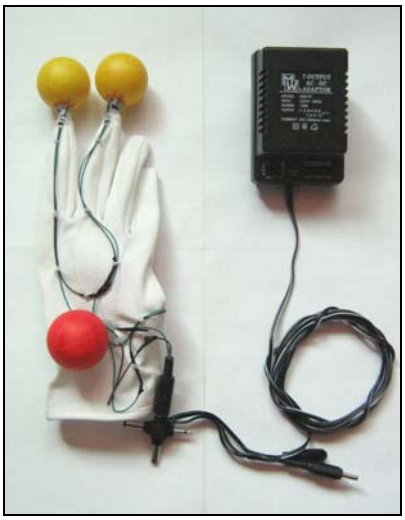


Figura 2.2

Como se puede apreciar en la figura 2.2, el *Guante Señalizador* cuenta con cuatro marcas (una de ellas no se ve ya que el mismo guante la oclusiona); dos amarillas y dos rojas. Las marcas amarillas están puestas en la punta de los dedos índice y medio; y las marcas rojas, sobre y bajo un poco antes de la muñeca. La idea de utilizar colores es ayudar al sistema de *Visión Artificial* a detectar de manera más rápida donde se encuentran las zonas de interés, que en el caso de las marcas rojas es la muñeca y en el de las amarillas son los dedos. No existe ninguna consideración especial en que el color amarillo haya sido asignado a los dedos y el rojo a la muñeca, ya que pudo haberse realizado al revés. Estas zonas marcadas son de especial interés ya que las posiciones obtenidas de ellas serán las variables de entrada para el *Modelo Cinemático*.

Si bien, el marcar las zonas de interés con colores ayuda al sistema de *Visión Artificial* distinguirlas, no es suficiente cuando se opera en ambientes no controlados. Así pues, para compensar este problema tales marcas son auto-iluminadas, es decir, generan luces de colores mencionados anteriormente. Esto permite que el sistema de visión pueda operar en ambientes

menos controlados ya que los colores que generan otros objetos lo hacen por rebote de la luz ambiente y son menos intensos que las marcas auto-iluminadas del guante. Obviamente, si existe una fuente de luz de color rojo o amarillo, el sistema no será capaz de distinguir esa luz de las marcas.

2.1.1 Construcción

Respecto a la construcción del guante, la figura 2.3 muestra una imagen de los principales materiales utilizados.

Figura 2.3

Las marcas fueron fabricadas con pelotas de ping-pong, pintadas con plumones de colores permanentes y dispuestos en el interior de cada una de ellas una pequeña ampolleta de 2,2V. Una vez construidas las marcas, se pegaron al guante por medio de una pistola con silicona. Las ampolletas se conectaron en serie por medio de cables y los terminales generales del circuito a un conector de audio tipo *mono*.

2.1.2 Requerimientos Energéticos

Los requerimientos energéticos del guante son de 1,1 Watts. (4 ampolletas x 2.2V x 0.13A) que son suministrados por un adaptador de 220 a 7.5V. Aunque teóricamente se requiere que el guante sea suministrado con 8.8V es complicado encontrar en el mercado un adaptador con esa característica de tensión, siendo la tensión mas cercana 9V. Sin embargo, no se recomienda utilizar esta magnitud de voltaje ya que un adaptador de mala calidad (alto índice de regulación) podría llegar a quemar las marcas.

2.2 Procesamiento de Imágenes

El Procesamiento de Imágenes es la parte mas importante de todo el sistema de *Visión Artificial* ya que es aquí donde se ejecutan los algoritmos más críticos para la obtención y cálculo de datos.

2.2.1 El Modelo de Color RGB

Este modelo de color es uno de los modelos más populares y ampliamente utilizados en sistemas que trabajan con imágenes. Sin embargo, no es el único ni el más eficiente. Existen otros modelos como el CMY (Cian, Magenta, Amarillo), YIQ (Luminancia, Croma Fase, Croma Cuadratura), HSI (Tono, Saturación, Intensidad) y el HSV (Tono, Saturación, Valor) [1].

Unas de las ventajas principales de este modelo es que sus componentes espectrales primarias rojo, verde y azul, se pueden asociar a un sistema de coordenadas cartesianas. La figura 2.4 muestra gráficamente esta idea.

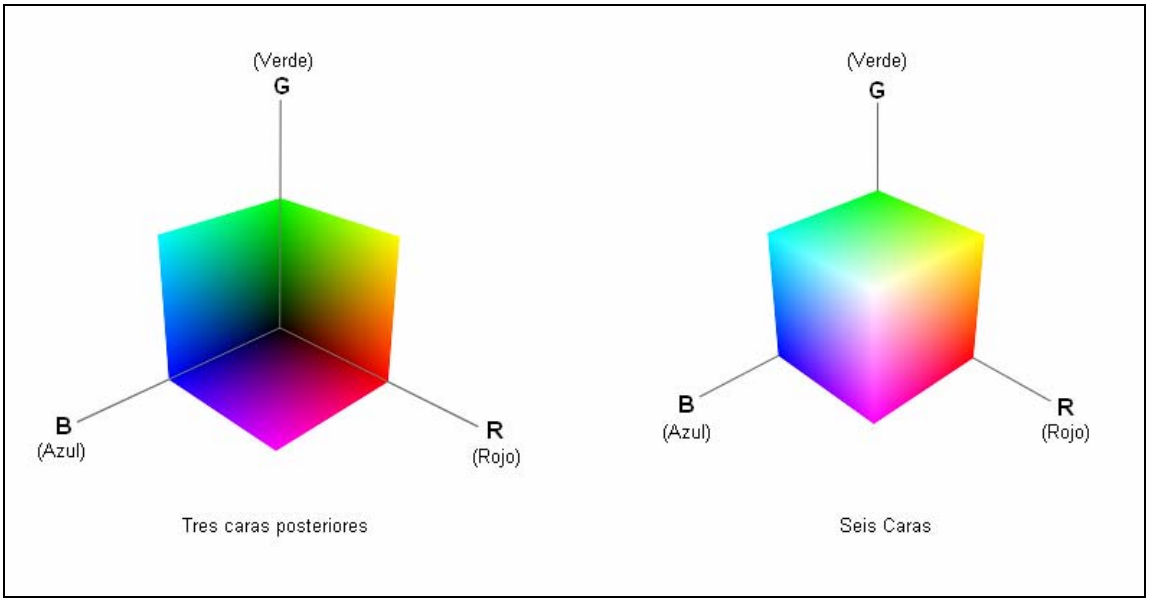


Figura 2.4

Así pues, la especificación de un color se hace de manera similar a la especificación de coordenadas espaciales de un punto, indicando la magnitud de cada componente primaria de color. Por ejemplo, si se desea obtener un color rojo puro solo basta indicar las coordenadas (255, 0, 0). La figura 2.5 muestra un ejemplo de algunas coordenadas de colores comunes.

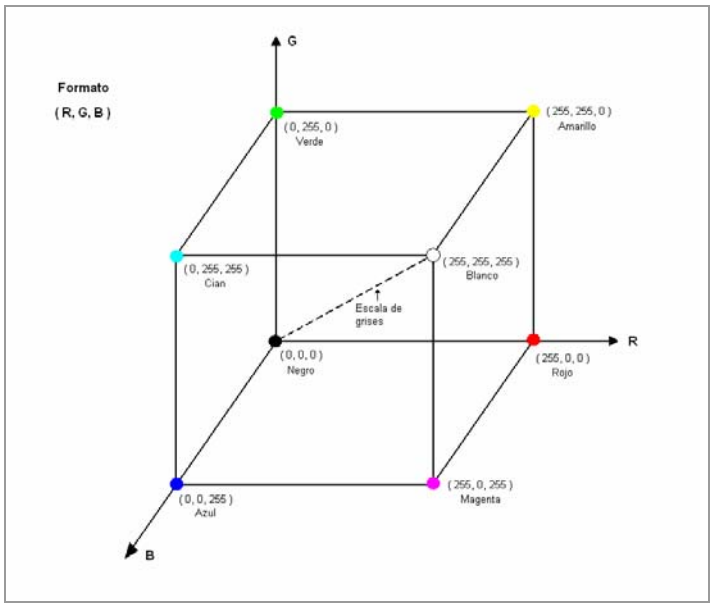


Figura 2.5

Además de la ventaja mencionada anteriormente, la mayoría de las cámaras Web trabajan en su hardware con este sistema de color, lo que significa que cuando capturan una imagen, ésta ya tiene en cada uno de los píxeles la especificación del color en formato RGB. Lo que para el sistema de *Visión Artificial* es muy provechoso desde el punto de vista de la velocidad de procesamiento ya que trabaja directamente sobre el sistema RGB y no necesita realizar ninguna conversión.

2.2.2 Detección de Colores

El proceso de Detección de Colores es el proceso encargado de realizar sobre las imágenes adquiridas, cálculos y mediciones orientados a la búsqueda de los colores rojo y amarillo. Estos colores fueron seleccionados como referencias debido a que son colores que tienen más independencia al ambiente, el perfil de color de las cámaras no distorsiona sus componentes de color y debido al tipo de pintado de las marcas, son aquellos que más luz de color se puede irradiar.

El algoritmo implementado para detectar colores se basa en el modelo de color RGB y en el cálculo de la distancia Euclidiana entre dos puntos en el espacio. Esta combinación es posible gracias a que tal modelo de color puede ser asociado a un sistema de coordenadas cartesiano, por lo que un color específico puede ser interpretado como un punto de coordenadas (R, G, B). La ecuación de distancia Euclidiana se muestra en la ecuación 2.1.

$$d = \sqrt{(R-r)^2 + (G-g)^2 + (B-b)^2} \qquad (2.1)$$

Donde:

- d: Distancia entre dos colores.
- R: Componente roja del color de referencia.
- G: Componente verde del color de referencia.
- B: Componente azul del color de referencia.
- r: Componente roja del color del píxel actual que esta siendo examinado.
- g: Componente verde del color del píxel actual que esta siendo examinado.
- b: Componente azul del color del píxel actual que esta siendo examinado.

La figura 2.6 muestra un ejemplo del cálculo de distancia de colores. En la fotografía se tomaron tres muestras de color nombradas como “Color Actual 1, Color Actual 2, Color Actual 3”. Los cuadros de color beige muestran las componentes RGB de color asociadas a cada muestra. Bajo la fotografía hay tres cubos en colores en donde se puede ver el color de cada muestra como una esfera. Su posición dentro del cubo representa las componentes sus componentes RGB. Así pues, en este ejemplo se calcula la distancia Euclidiana de cada muestra respecto al color amarillo, siendo representada esta distancia por una línea roja que conecta las esferas mencionadas.

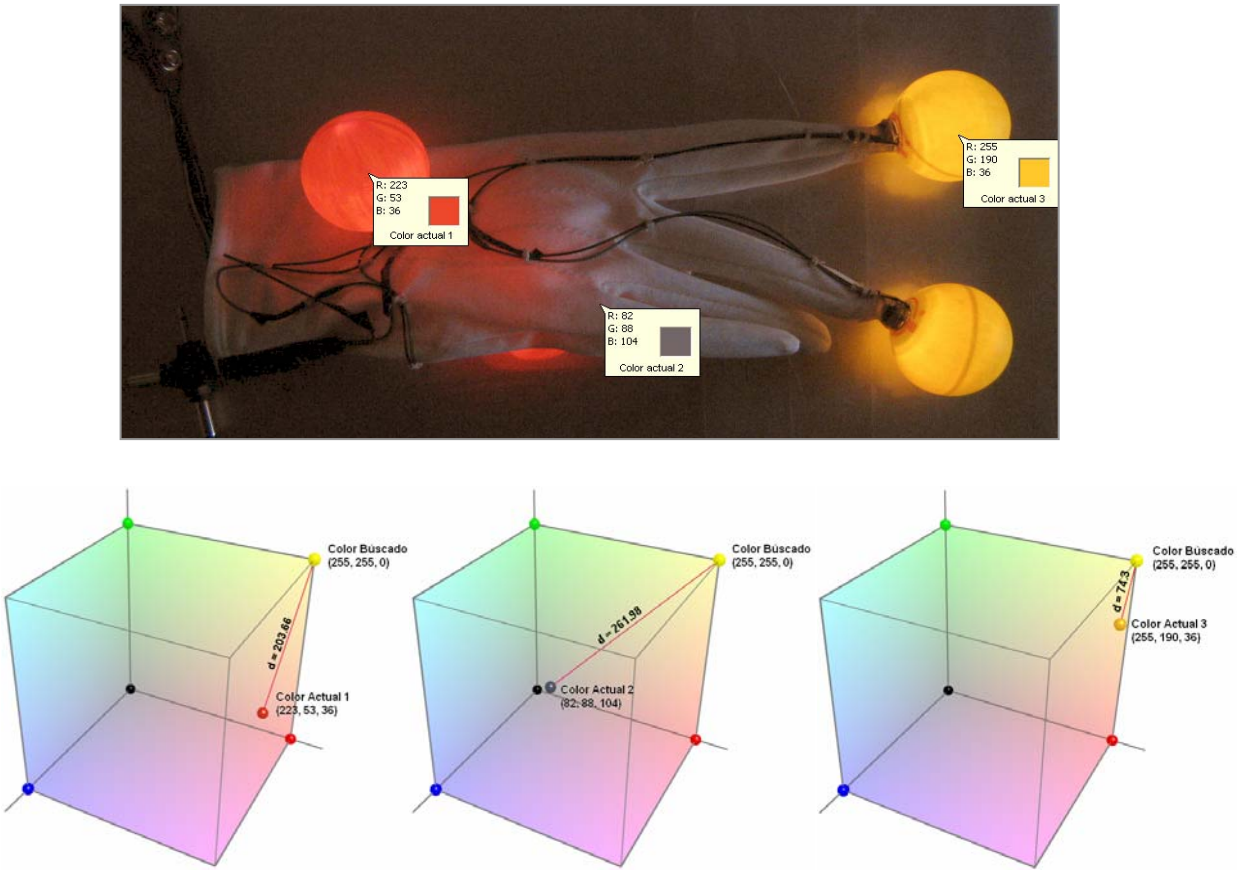


Figura 2.6

Debido a que el cálculo de distancia de color se realiza entre dos puntos en el espacio de color RGB, un punto correspondiente al color de referencia y otro punto correspondiente al color de un píxel cualquiera; en el espacio de una imagen, el cálculo se realiza sobre todos sus píxeles. Esta operación genera otra imagen de intensidades de iguales dimensiones pero conteniendo en

cada píxel la magnitud de la distancia existente entre el color de referencia y el color del píxel de la imagen original. La figura 2.7 muestra este proceso, teniendo como color de referencia el amarillo puro (255, 255, 0).

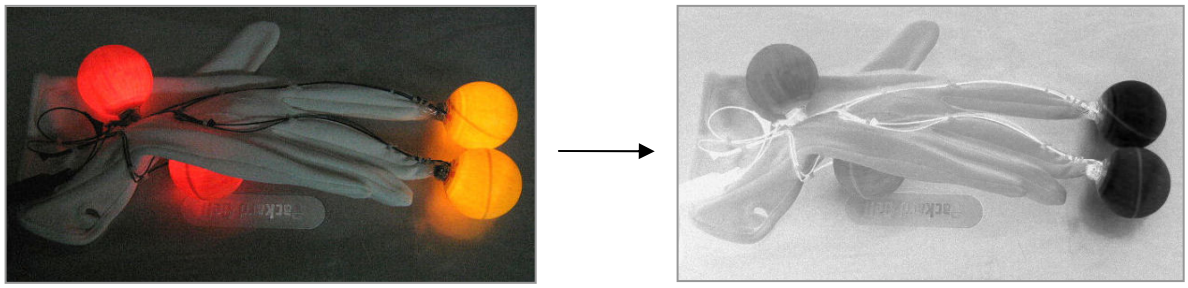


Imagen Original Imagen Distancias

Figura 2.7

En la Imagen Distancias de la figura de arriba se puede apreciar que existe una zona oscura que corresponde a las distancias más pequeñas existentes entre los píxeles amarillos de la Imagen Original y el color de referencia amarillo. En la medida que colores de los píxeles de la Imagen Original se alejen del color de referencia, mayor serán las distancias y más claros serán los píxeles de la Imagen de Distancias.

La Imagen Distancias también puede ser interpretada en forma tridimensional como si fuera un terreno, donde las zonas mas oscuras representan depresiones y las zonas más claras representan elevaciones. La figura 2.8 muestra gráficamente esta idea.

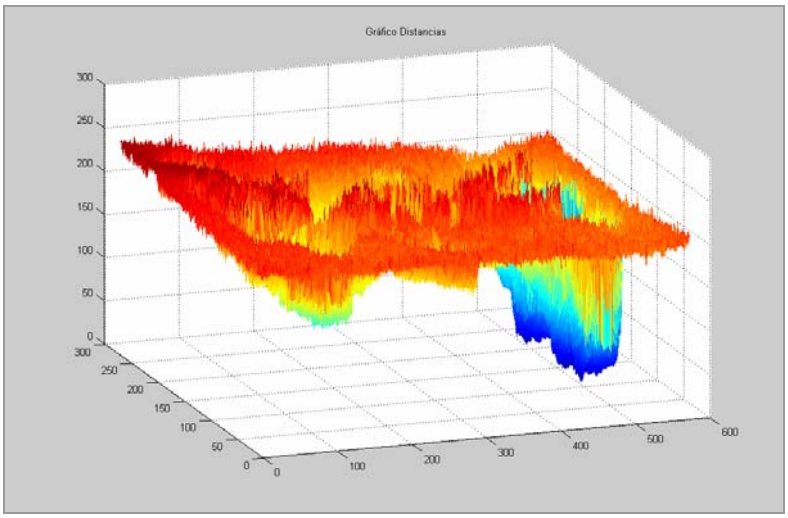


Figura 2.8

La detección del color rojo se realiza de manera análoga. En la ecuación de distancia Euclidiana se cambia las componentes del color de referencia amarillo por las componentes de color rojo (255, 0, 0). Las Figuras 2.9 y 2.10 muestran la detección del color rojo.

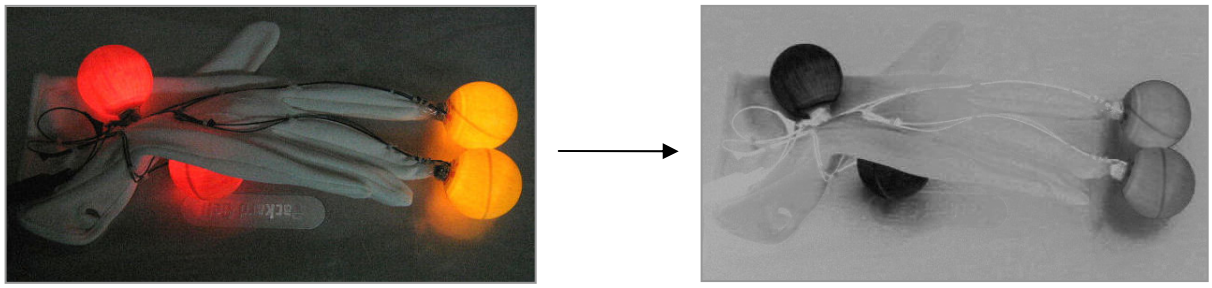


Imagen Original Imagen Distancias

Figura 2.9

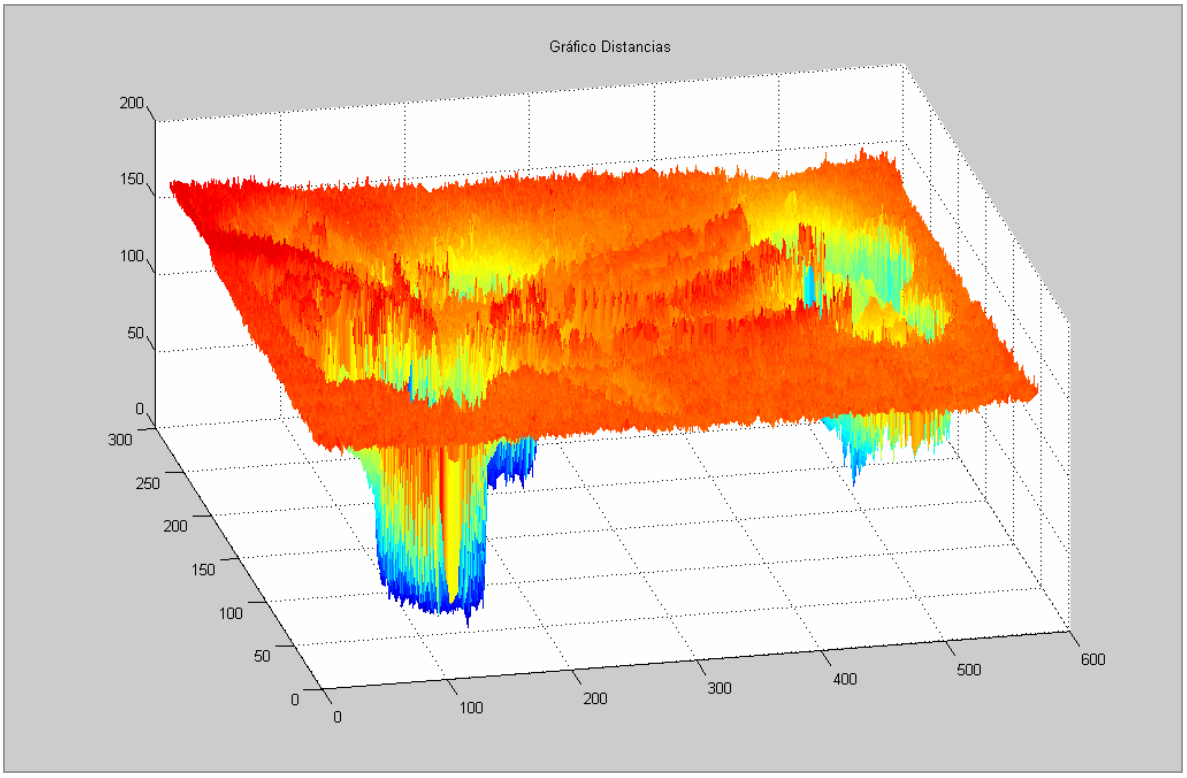


Figura 2.10

2.2.3 Segmentación de Colores mediante Umbralización

La Umbralización es uno de los métodos más importantes en la segmentación de imágenes, además de ser el más sencillo. Básicamente consiste en definir un nivel de decisión que sirva para separar la información que nos interesa del resto o fondo de la imagen. En el caso del sistema de *Visión Artificial*, la función de la Umbralización es operar sobre las Imágenes Distancias y separar las distancias de gran magnitud de las de pequeña magnitud para que de esta manera se aíslen las marcas de interés del guante, del resto de la imagen. Cabe mencionar que los niveles de Umbralización utilizados no son estáticos pero configurables por el usuario. La configuración de estos niveles de decisión se ven con mas detalle en el capitulo V. En figura 2.11 se muestra los resultados de la Umbralización de las distancias del color amarillo y rojo.

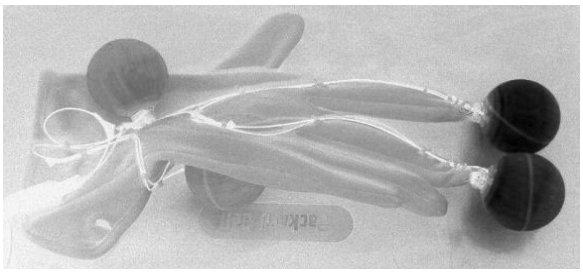


Imagen Distancias color Amarillo

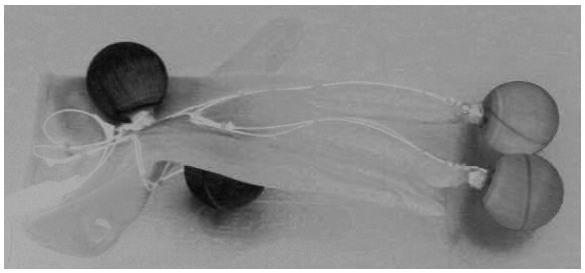


Imagen Distancias color Rojo

Imagen Umbralizada color Amarillo
(Distancia Umbral = 60)

Imagen Umbralizada color Rojo
(Distancia Umbral = 50)

Figura 2.11

2.2.4 Dilatación y Erosión

Son operaciones morfológicas que se realizan sobre las imágenes binarias, esto es, operan sobre la forma del objeto en la imagen. Como argumento requieren dos imágenes, donde una es la imagen a procesar y la otra es el elemento estructural. El elemento estructural es en morfología matemática lo que la máscara de convolución es en los filtros lineales. Los elementos estructurales más comunes son los conjuntos de píxeles que están 4-conectados y 8-conectados, mostrados en la siguiente figura.

0	1	0
1	1	1
0	1	0

4 conectados

1	1	1
1	1	1
1	1	1

8 conectados

Figura 2.12

Los algoritmos asociados a cada tipo de operación son bastante sencillos y se explican a continuación:

Dilatación: Toma cada píxel del objeto con valor “1” y pone a en valor “1” todos aquellos píxeles pertenecientes al fondo que tienen una conectividad 4-conectados u 8-conectados con el píxel del objeto.

Erosión: Básicamente es lo contrario a la Dilatación ya que toma cada píxel del objeto que tiene una conectividad 4-conectados u 8-conectados con los píxeles del fondo y lo pone al valor “0”.

La siguiente figura muestra un ejemplo de los algoritmos explicados.

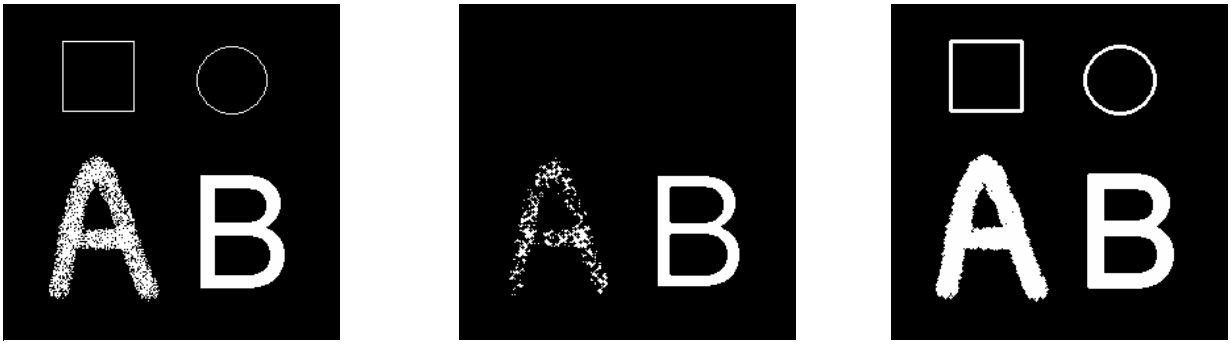


Imagen Binaria Original | Imagen Erosionada | Imagen Dilatada

Figura 2.13

Para el sistema de visión, estas operaciones morfológicas actúan como ayuda para el proceso de Detección de Grupos de Píxeles ya que permiten a filtrar parte del ruido que se podría presentar en las imágenes adquiridas por las cámaras. Sin embargo la utilización de estos queda a criterio del usuario debido a que en determinadas condiciones de luz ambiente y niveles de umbralización podrían empeorar el proceso de detección de marcas. La figura 2.14 muestra este procesamiento en las marcas amarillas del *Guante* Señalizador.

Imagen Original | Imagen Erosionada 3 veces | Imagen Dilatada 3 veces | Imagen Erosionada y Dilatada 3 veces

Figura 2.14

2.2.5 Detección de Grupos de Píxeles

La Detección de Grupos de Píxeles consiste en la búsqueda de píxeles que estén conectados en una imagen binaria o umbralizada, para luego darle un nombre o etiqueta a cada

grupo encontrado. De esta manera se pueden diferenciar varios objetos en una misma imagen. La figura 2.15 muestra un ejemplo de detección de grupos de píxeles.

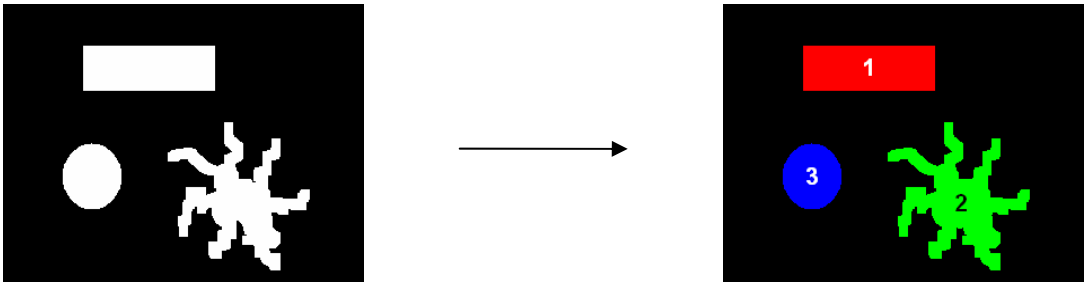


Imagen Umbralizada Grupos Detectados y Etiquetados

Figura 2.15

El algoritmo basado en [2], consiste en identificar mediante una etiqueta cada objeto existente en la imagen. Como objeto se define aquel grupo de píxeles 4 conectados que tienen el mismo color. Para etiquetar se realiza una pasada por la imagen de arriba hacia abajo y de izquierda a derecha, rellenando con la etiqueta los nuevos objetos encontrados. El algoritmo de rellenado es el siguiente:

1º Se marca con la etiqueta el primer píxel encontrado del objeto, y se introduce en una pila su posición (x, y).

2º Se saca de la pila un píxel.

3º Se miran el color de los 4 vecinos del píxel, si alguno de estos píxeles coincide en color se marca con la etiqueta y se introduce en la pila su posición (x, y).

4º Se repiten los pasos 2, 3 y 4 hasta que la pila está vacía.

Para el sistema de visión este algoritmo es muy útil ya que permite diferenciar las posiciones espaciales de los dedos del *Guante Señalizador*, pudiendo obtener otros datos importantes como la detección de apertura/cierre y el punto medio de ellos para el cálculo de la inclinación de la muñeca. La figura 2.16 muestra la detección de grupos de píxeles conectados para los dedos del *Guante Señalizador*.

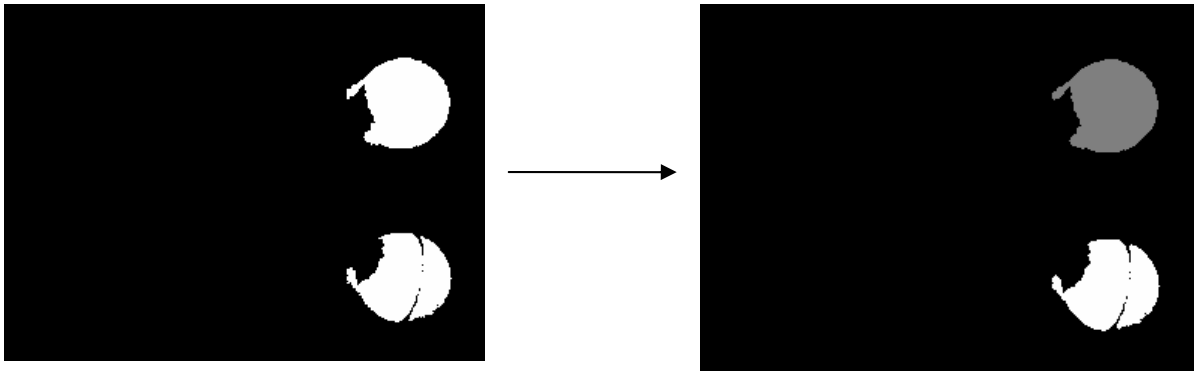


Imagen Umbralizada

Grupos Detectados y Etiquetados
(Indicados por Color)

Figura 2.16

2.2.6 Determinación de Coordenadas de las Marcas

La determinación de las coordenadas de cada marca está basada en el cálculo del centro de masa de los píxeles de una imagen binarizada. De esta manera no importa la forma que tenga el objeto segmentado en la imagen binaria, el cálculo de centro de masa siempre calculará el centro del objeto, a diferencia de otros métodos de obtención de coordenadas como el "mínimo cuadro que encierra el objeto" que no siempre encuentra el centro. Las siguientes ecuaciones se aplican para la obtención del centro de masa de las marcas de guante:

$$\bar{x} = \frac{\sum_{y=0}^{alto-1} \sum_{x=0}^{ancho-1} x \cdot img(x,y)}{\sum_{y=0}^{alto-1} \sum_{x=0}^{ancho-1} img(x,y)} \tag{2.2}$$

$$\bar{y} = \frac{\sum_{x=0}^{ancho-1} \sum_{y=0}^{alto-1} y \cdot img(x,y)}{\sum_{y=0}^{alto-1} \sum_{x=0}^{ancho-1} img(x,y)} \tag{2.3}$$

Cabe mencionar que las pociones resultantes de este proceso están referenciadas al origen de cada imagen obtenida, por lo que aún falta reconstruirlas en forma tridimensional para obtener las posiciones espaciales reales del *Guante Señalizador*.

2.2.7 Suavizado de Movimientos

El sistema de visión está constantemente adquiriendo imágenes desde las cámaras Web, lo que implica que continuamente se están actualizando los cálculos de las posiciones de las marcas del guante. Sin embargo, como todo sistema discreto, solo se trabaja con muestras de las trayectorias realizadas por la mano del usuario, por lo que, dependiendo de la velocidad de adquisición y procesamiento de imágenes es muy probable que frente a movimientos bruscos, el comportamiento del sistema no sea el deseado. De hecho, en un computador con procesador Intel Centrino de 2.4Ghz y 512 MB de RAM, la velocidad de procesamiento de información alcanza a ser de 6 a 8 imágenes por segundo. Debido a esta problemática, la etapa de Suavizado de Movimientos, al igual que las operaciones morfológicas Dilatación y Erosión, ayuda a determinar de mejor manera la posición de las marcas.

El Suavizado de Movimientos consiste en el cálculo de la media de las últimas N muestras almacenadas de las posiciones calculadas por el proceso de determinación de centros de masas, cantidad de muestras que puede ser configura por el usuario. Matemáticamente:

$$\overline{X} = \frac{\sum_{i=0}^{N-1} \overline{x}_i}{N} \tag{2.4}$$

$$\overline{Y} = \frac{\sum_{i=0}^{N-1} \overline{y}_i}{N} \tag{2.5}$$

Donde:

$\left(\overline{X}, \overline{Y}\right)$: Coordenadas de la posición actual suavizadas.

$\left(\overline{x_i}, \overline{y_i}\right)$: Coordenadas de la muestra i-ésima almacenada.

N : Cantidad de muestras recordadas.

2.2.8 Detección de Apertura/Cierre de Dedos

La detección de los estados “abiertos” o “cerrados” de los dedos de la mano del usuario es sencilla, y se basa exclusivamente en la cantidad de grupos de píxeles detectados por el sistema de visión. Cuando las marcas amarillas del *Guante Señalizador* están separadas, el sistema de visión detectará dos grupos de píxeles, por lo tanto se concluye que los dedos están separados. Análogamente, cuando las marcas amarillas están juntas, el sistema de visión detectará solamente un grupo de píxeles, por lo que se considera que los dedos están juntos. Sin embargo, cualquier otra condición respecto a la cantidad de grupos de píxeles será ignorada por el software y se mantendrá el estado anterior detectado. La siguiente figura muestra un ejemplo de lo explicado.

Dedos Abiertos
(Dos grupos de píxeles detectados)

Dedos Cerrados
(Un grupo de píxeles detectado)

Figura 2.17

2.3 Triangularización

La Triangularización es el proceso de determinar posiciones tridimensionales de puntos basados en dos o más imágenes de un mismo objeto, que se obtienen por cámaras separadas espacialmente una distancia determinada. Para lograr la triangularización es necesario determinar ciertos parámetros de las cámaras como la apertura focal, así como también el posicionamiento y la alineación de ellas. La siguiente figura muestra un ejemplo de triangularización con dos cámaras.

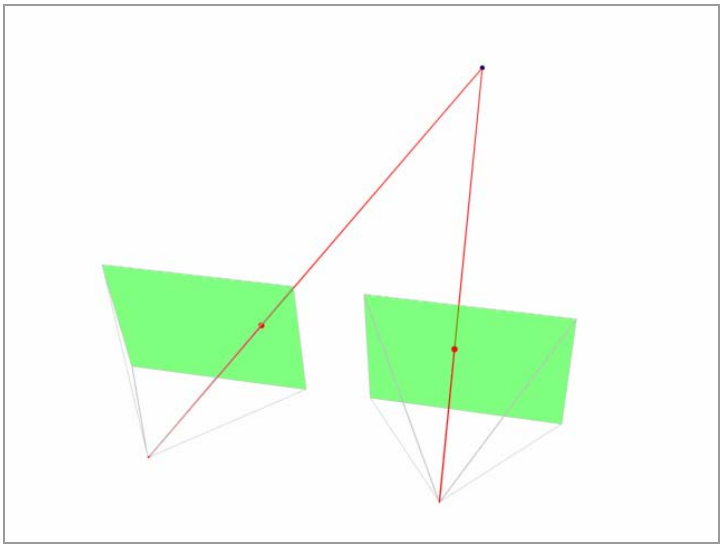


Figura 2.18

2.3.1 Determinación de la Apertura Focal de las cámaras

La determinación de la Apertura Focal de las cámaras es importante en los cálculos de triangulación por las siguientes razones: Permite conocer el volumen visual, calcular la distancia focal existente entre el foco y el plano de proyección, y decidir como posicionar y alinear las cámaras Web. El modelo utilizado para la determinación de este parámetro se muestra en la figura 2.19.

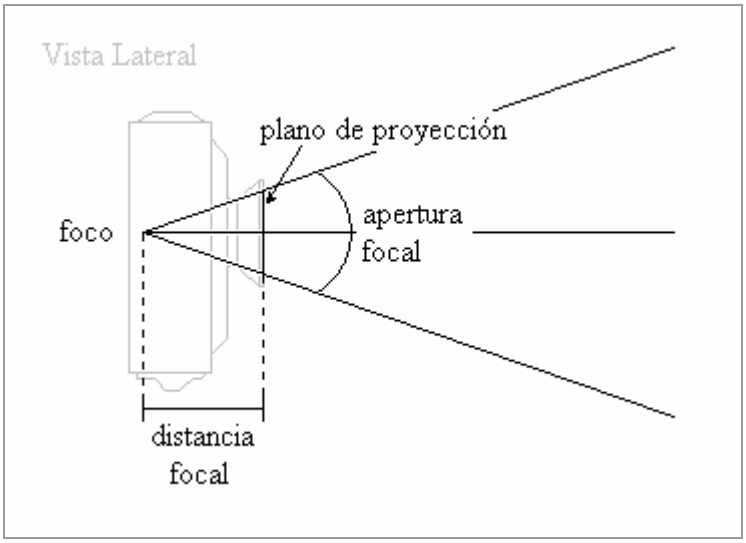


Figura 2.19

Para realizar la determinación de los parámetros mencionados se dispuso una de las cámaras Web Creative NXUltra frente a una pared a una distancia de 100mm entre la lente (plano de proyección) y la pared misma. Luego, se adhirió una hoja de papel (Plano de Medición) a la pared para marcar los límites visuales de la cámara. Una vez realizado este montaje, se procedió a medir el diámetro de la lente poniendo justo frente a ella una regla milimetrada. Es importante aclarar que la medición se hizo a través de la imagen capturada por la cámara y no directamente sobre la lente, debido a que son de interés las dimensiones efectivas que "ve" la cámara y no las de su geometría física. Así pues, el objetivo de esta medición es conocer el ancho del Plano de Proyección que resultó ser de 10mm de longitud. La figura 2.20 muestra la imagen capturada por la cámara en el momento de esta medición.

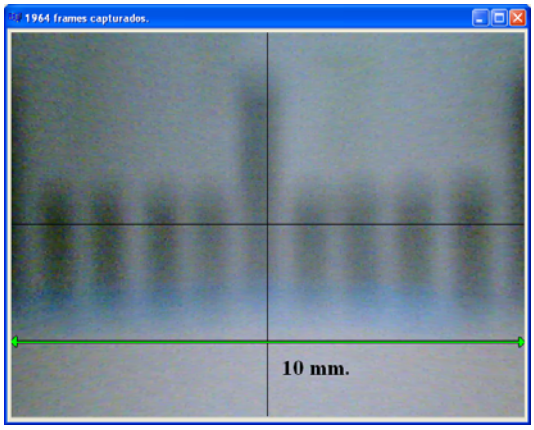


Figura 2.20

Luego, se repite el mismo procedimiento pero con el Plano de Medición (hoja de papel). A través de las imágenes que capturaba la cámara se fueron marcando en la hoja de papel los límites visuales o bordes de la imagen. Finalizado esto, se midió la longitud entre marcas horizontales hechas en la hoja de papel, siendo de 130mm. En la figura 2.21 se muestra una imagen escaneada de la hoja de papel utilizada en la marcación y medición.

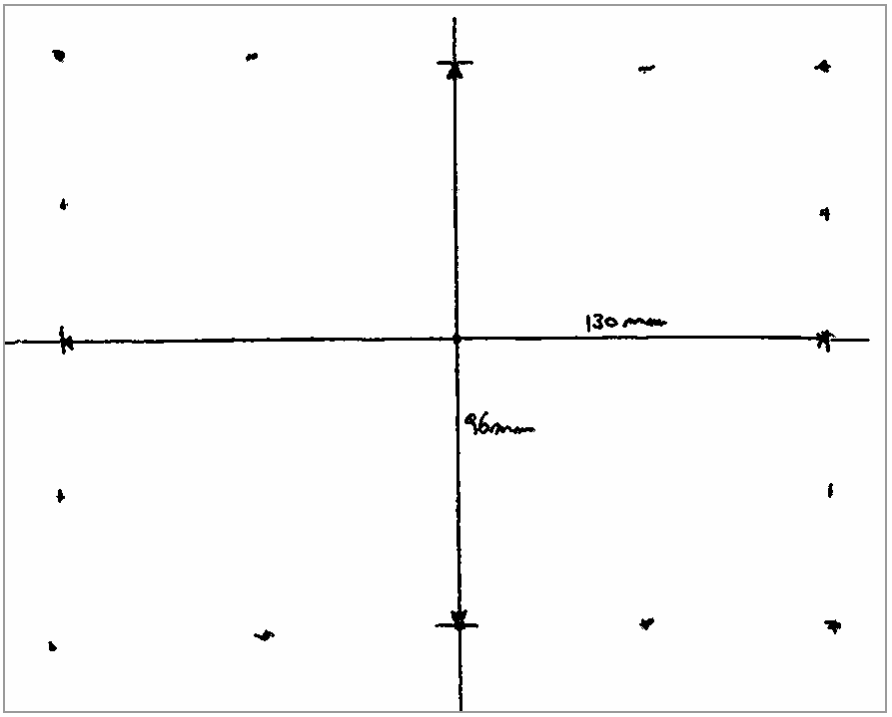


Figura 2.21

Ya obtenidos todos los datos necesarios de las mediciones realizadas, es posible determinar la distancia focal y la apertura focal de las cámaras. La siguiente figura muestra un resumen de las mediciones realizadas.

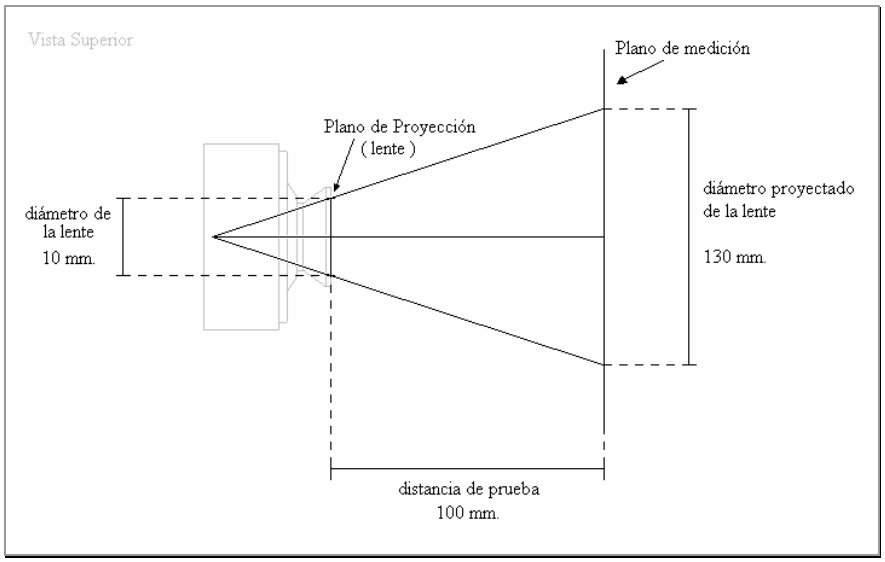


Figura 2.22

Basado en la mitad del modelo de la cámara mostrado en la figura 2.23, el cálculo de la distancia focal utiliza el teorema de los triángulos semejantes como se muestra a continuación:

$$\frac{x}{f+L} = \frac{xp}{f} \tag{2.6}$$

$$f = \frac{L \cdot xp}{x - xp} \tag{2.7}$$

Donde:

x: Mitad del diámetro proyectado de la lente.

xp: Mitad del diámetro del lente.

L: Longitud entre la lente y el plano de medición.

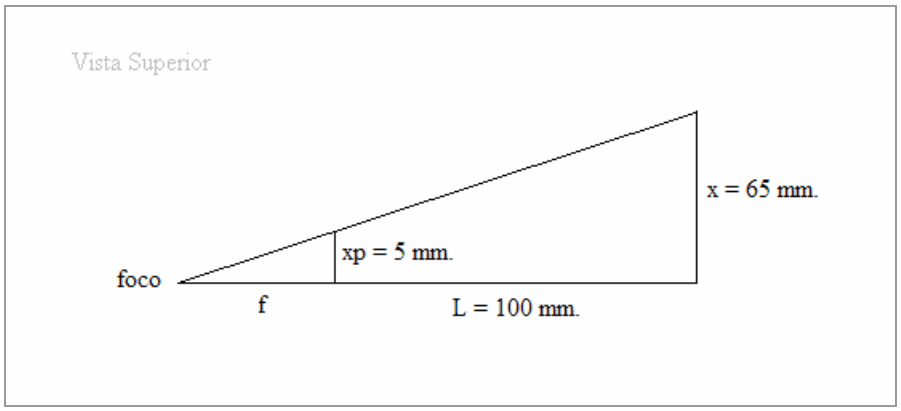


Figura 2.23

Reemplazando valores, tenemos:

$$f = \frac{100 \cdot 5}{65 - 5} = 8.3333mm. \tag{2.8}$$

Conocida la distancia focal es posible conocer la apertura focal. Según [3], el cálculo es como sigue:

$$\theta = 2 \cdot tg^{-1}\left(\frac{A}{2 \cdot f}\right) \tag{2.9}$$

Donde:

θ: Apertura focal.

A: Diámetro de la lente.

f: Distancia focal.

Utilizando los valores obtenidos y calculados previamente, tenemos:

$$\theta = 2 \cdot tg^{-1}\left(\frac{10}{2 \cdot 8.33}\right) = 61.93^{o} \tag{2.10}$$

Por lo tanto, se concluye que la apertura focal efectiva de la cámara Web es de 61.93º.

Por otra parte, si bien la distancia focal es un parámetro constante en la cámara, la apertura focal vertical no es igual a la apertura focal horizontal calculada. Esto debido a que las imágenes que capturan la mayoría de las cámaras no son cuadradas, es decir, su resolución horizontal es distinta a la vertical. Generalmente la horizontal es mayor que la vertical, lo que conlleva a deducir que la apertura focal vertical es menor. Sin embargo, el parámetro de apertura vertical no es utilizado por el sistema de *Visión Artificial*, sino que utilizado el otro parámetro asociado: la distancia focal. De todas maneras se da a conocer al lector la apertura focal vertical efectiva:

$$\theta = 2 \cdot tg^{-1}\left(\frac{7}{2 \cdot 8.33}\right) = 45.58^{o} \tag{2.11}$$

Cabe mencionar que en los cálculos realizados, no se ha considerado la distorsión radial provocada por la geometría convexa del lente, debido a que este problema ya esta solucionado por las cámaras Web Creative NXUltra que traen en su programa de configuración la posibilidad de solucionar este inconveniente. Sin embargo, a continuación se muestran las ecuaciones que según [4], solucionan este problema.

$$x = xd \cdot (1 + k \cdot rd^2) \tag{2.12}$$

$$y = yd \cdot (1 + k \cdot rd^2) \tag{2.13}$$

Donde:

(xd, yd): Son las coordenadas de un píxel de la imagen distorsionada.

k: Es una constante dependiente de la distorsión del lente.

(x, y) : Son las nuevas coordenadas del píxel de la imagen distorsionada.

Aunque ya se conocen todos los parámetros necesario de las cámaras, los objetos que aparezcan en las imágenes capturadas por ellas tendrán unidades de píxeles, por lo que hasta este momento las unidades de estos objetos y la de los parámetros son incompatibles. Por este motivo, es necesario realizar una conversión de unidades para los parámetros de las cámaras. Para ello se seleccionó una resolución espacial de 320x240 píxeles como resolución de trabajo, ya que ésta resolución es la que mejor prestaciones de velocidad procesamiento y sensibilidad espacial ofrece al sistema de *Visión Artificial*. Debe tenerse en cuenta que se toma como referencia la dimensión del Plano de Proyección y no la del Plano de Prueba debido a que todo objeto que este dentro del volumen visual de la cámara se "proyectará" hacia el lente de la cámara, por lo que las dimensiones que adquirirá serán respecto al Plano de Proyección. Teniendo en cuenta estos datos y basado en la relación lineal de magnitudes o "regla de tres", la conversión de unidades se realizó de la siguiente manera: Recordando que la medición del diámetro del lente de la cámara resultó ser 10mm, ésta dimensión es análoga a los 320 píxeles de dimensión horizontal del Plano de Proyección, por lo que pueden ser asociadas en una relación simple:

$$\frac{xp_{mm}}{xp_{pixeles}} = \frac{m_{mm}}{m_{pixeles}} \tag{2.14}$$

$$m_{pixeles} = \frac{xp_{pixeles} \cdot m_{mm}}{xp_{mm}} \tag{2.15}$$

$$m_{pixeles} = \frac{320 \cdot m_{mm}}{10} = 32 \cdot m_{mm} \tag{2.16}$$

Donde:

m_{mm}: Magnitud en unidades de mm. a convertir.

xp_{mm} : Dimensiones del Plano de Proyección en mm.

$xp_{pixeles}$: Dimensiones del Plano de Proyección en píxeles.

$m_{pixeles}$: Magnitud convertida a unidades de píxeles.

Por ahora, el parámetro que interesa convertir es la distancia focal. Más adelante se explicará la conversión de otros parámetros.

$$d_{pixeles} = 32 \cdot 8.3333 = 266.67 \qquad (2.17)$$

La figura 2.24 muestra gráficamente la relación de estos datos en el modelo de la cámara, pero a diferencia de los cálculos mostrados solo se consideró la mitad de las magnitudes, aunque el resultado es mismo debido a que se mantiene la relación.

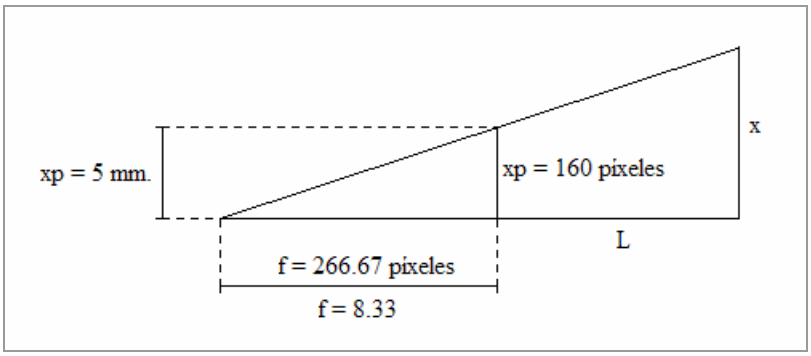


Figura 2.24

2.3.2 Posicionamiento y Orientación de Cámaras

Como se ha explicado anteriormente, la Triangularización se basa en el análisis de imágenes capturadas por cámaras separadas espacialmente una distancia determinada, siendo el posicionamiento más común el mostrado en la figura 2.17, correspondiente al posicionamiento por disparidad paralela. Sin embargo, para el sistema de *Visión Artificial* no es la configuración más adecuada debido a que no resuelve bien el problema de la oclusión. Por este motivo la configuración seleccionada fue el posicionamiento ortogonal de cámaras, mostrada en la siguiente figura:

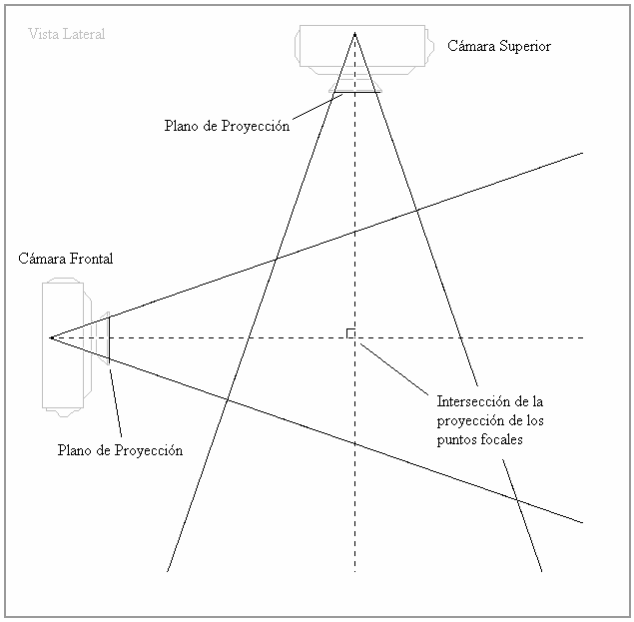


Figura 2.25

Además, este tipo de configuración ofrece la ventaja de que es más exacto para la determinación de las coordenadas espaciales de los objetos.

Este tipo de posicionamiento de las cámaras consiste en poner una arriba y otra enfrente del espacio volumétrico destinado al seguimiento del *Guante Señalizador*, ver figura 2.26.

Figura 2.26

Aunque la distancia existente entre cada Plano de Proyección y la intersección de la proyección de los puntos focales (intersección de líneas punteadas de la imagen 2.25) no es crítica, ésta se fijo en 120 cms. debido que a esta distancia se logró balance razonable entre el espacio volumétrico del *Guante Señalizador* y la peor segmentación que podrían tener las marcas del guante. La siguiente figura muestra todas las dimensiones de posicionamiento de las cámaras como el cubo de medición generado.

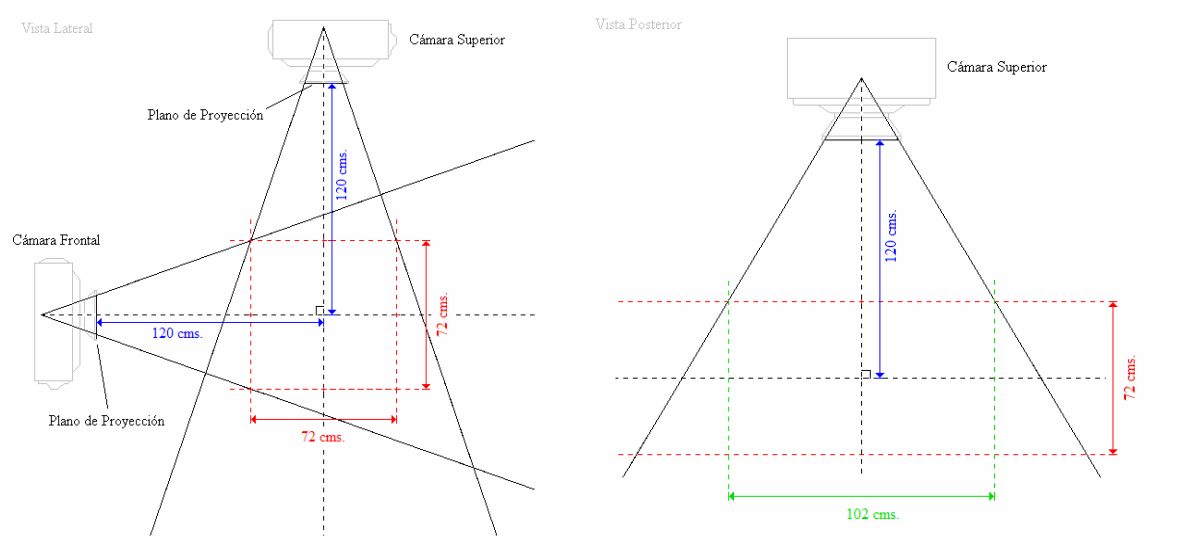


Figura 2.27

Las dimensiones del cubo de medición se pudieron realizar gracias a la determinación de la apertura focal vertical y horizontal realizada en el punto 2.3.1, resultando ser de 102x72x72 cms.

Por otra parte, la determinación de la peor segmentación de las marcas del guante dentro del cubo de medición se realizó de la siguiente manera: considerando que cada marca tiene un diámetro aproximado de 40mm. y que la distancia más lejana a la que podía estar ésta marca son 156.8 cms.(120cms.+72cms./2+8.333mm) en la apertura focal más amplia (apertura focal horizontal), sólo basta calcular a cuantos píxeles equivale cada marca en el Plano de Proyección. Utilizando la relación (2.4):

$$xp = \frac{f \cdot x}{L} = \frac{8.3333 \cdot 40}{1568.3333} = 0.21253mm. \tag{2.18}$$

Y utilizando la relación de conversión (2.14):

$$m_{pixeles} = 32 \cdot 0.21253 = 6.80096_{pixeles} \tag{2.19}$$

Es decir, el diámetro de 40mm. de las marcas del guante, a la mayor distancia permitida dentro del cubo de medición, se proyectarán hacia el Plano de Proyección como masas de píxeles que tendrán como mínimo un diámetro de seis píxeles.

2.3.3 Proyección Inversa de Puntos

La Proyección Inversa de Puntos consiste en determinar las posiciones espaciales XYZ de las marcas del *Guante Señalizador* dentro del Cubo de Medición, a partir de las coordenadas del centro de masas de los píxeles segmentados en el proceso de umbralización. El nombre "Proyección Inversa de Puntos" hace referencia al proceso contrario de proyectar puntos pertenecientes a un volumen de visión en un plano cualquiera utilizando un punto de fuga. En otras palabras, la idea consiste sacar los puntos que están en el Plano de Proyección y llevarlos a su posición original XYZ. La figura 2.28 muestra gráficamente esta idea en el plano XZ.

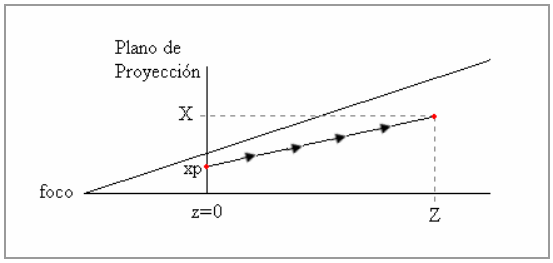


Figura 2.28

La ecuación que rige la proyección entre un punto del espació visual y el plano de proyección es la misma que se mostró en la ecuación (2.4). Debido a la característica bidimensional de los puntos proyectados en el Plano de Proyección, se genera la problemática de que para cada coordenada solo se posee una ecuación, ver ecuaciones (2.20) y (2.21).

$$X = \frac{xp \cdot (f + Z)}{f} \tag{2.20}$$

$$Y = \frac{yp \cdot (f + Z)}{f} \tag{2.21}$$

Aunque se trate de conectar las ecuaciones igualándolas por medio de la coordenada Z, aún tendremos una ecuación y dos incógnitas. Por este motivo fue necesario recurrir a la triangularización de Imágenes, para así poder disponer de otro par de ecuaciones que tuviera relación con las mostradas anteriormente.

Basado en la configuración de cámaras mostradas en la figura 2.24, se seleccionó la cámara superior como sistema de referencia global para el Cubo de Medición, ya que la posición de esta cámara ofrece mayor facilidad en la interpretación de las coordenadas espaciales.

En resumen, debido a la presencia de las dos cámaras, se tiene el siguiente sistema de ecuaciones:

$$x_{1p} = \frac{X_1 \cdot f}{Z_1 + f} \tag{2.22}$$

$$y_{1p} = \frac{Y_1 \cdot f}{Z_1 + f} \tag{2.23}$$

$$x_{2p} = \frac{X_2 \cdot f}{Z_2 + f} \tag{2.24}$$

$$y_{2p} = \frac{Y_2 \cdot f}{Z_2 + f} \tag{2.25}$$

También, se muestra en la siguiente figura los sistemas de referencias de cada cámara y las equivalencias de coordenadas entre los sistemas.

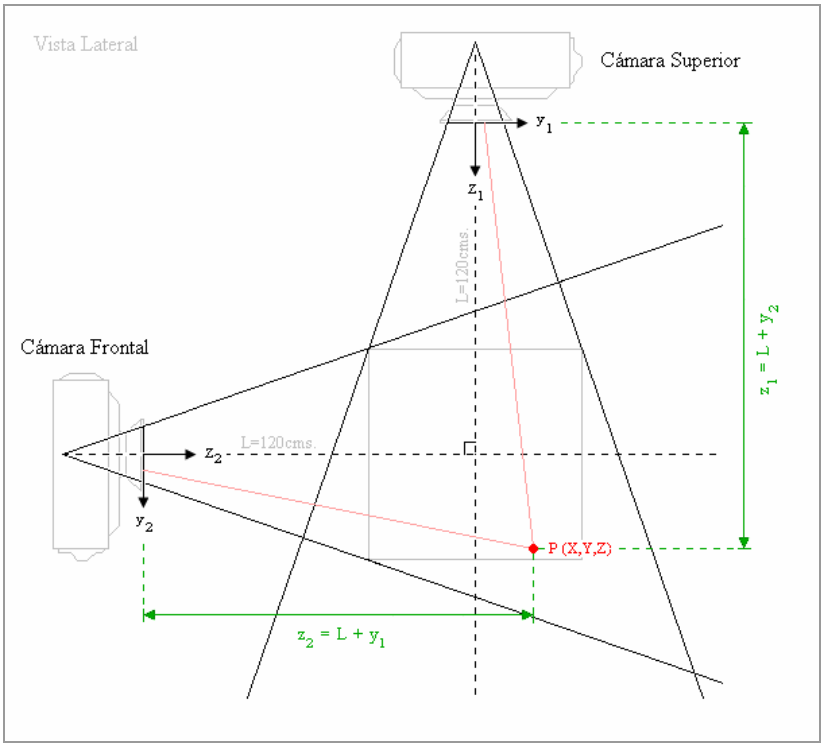


Figura 2.29

Tomando la ecuación (2.23) se despeja la variable de interés Y_1, y considerando que $Z_1 = L + Y_2$, se tiene:

$$Y_1 = y_{1p} + y_{1p} \cdot \frac{L}{f} + \frac{y_{1p}}{f} \cdot Y_2 \qquad (2.26)$$

Como apareció el término Y_2, es necesario despejarla del par de ecuaciones de la otra cámara. Por lo tanto, en base a (2.25) se tiene:

$$Y_2 = \frac{y_{2p}(Z_2 + f)}{f} \qquad (2.27)$$

Reemplazando (2.27) en (2.26) se obtiene:

$$Y_1 = y_{1p} + y_{1p} \cdot \frac{L}{f} + \frac{y_{1p} \cdot y_{2p}}{f} + \frac{y_{1p} \cdot y_{2p}}{f^2} \cdot Z_2 \qquad (2.28)$$

Nuevamente se observa la presencia de un nuevo término perteneciente al sistema de referencia de la cámara frontal. Sin embargo, utilizando la equivalencia de coordenadas $Z_2 = L + Y_1$ se puede referenciar el término Z_2 al sistema de referencia de la cámara superior de la siguiente manera:

$$Y_1 = y_{1p} + y_{1p} \cdot \frac{L}{f} + \frac{y_{1p} \cdot y_{2p}}{f} + \frac{y_{1p} \cdot y_{2p}}{f^2} \cdot (L + Y_1) \tag{2.29}$$

Ordenando y agrupando los términos se concluye que la coordenada Y, proyectada de forma inversa, de cada marca del *Guante Señalizador*, puede determinarse mediante (2.30).

$$Y_1 = \left[\frac{f^2 \cdot y_{1p}}{f^2 - y_{1p} \cdot y_{2p}} \right] \cdot \left[1 + \frac{1}{f} \left(L + y_{2p} \left[1 + \frac{L}{f} \right] \right) \right] \tag{2.30}$$

Cabe mencionar que el subíndice “1” de Y_1 indica que ésta coordenada esta referenciada a la cámara superior.

Como ya se conoce una de las tres coordenadas, la determinación de las ecuaciones para las dos coordenadas restantes es más sencilla y se muestra a continuación. Basado en (2.23), la coordenada Z puede ser determinada mediante:

$$Z_1 = f \cdot \left(\frac{Y_1}{y_{1p}} - 1 \right) \tag{2.31}$$

Pero existe un inconveniente con esta ecuación debido a que es probable, que en algún momento, el término y_{1p} pueda valer cero. Esto conlleva a una división por cero lo que consecuentemente hace que la coordenada Z_1 sea indeterminada. Sin embargo, este percance puede solucionarse despejando la coordenada Z_1 de la ecuación (2.25), teniendo en consideración las equivalencias $Y_2 = Z_1 - L$ y $Z_2 = L + Y_1$. Reemplazando y desarrollando se tiene:

$$Z_1 = \frac{y_{2p} \cdot (L + Y_1 + f)}{f} + L \qquad (2.32)$$

Por último, la coordenada X se puede conocer reordenando de la expresión (2.22) como se muestra a continuación:

$$X_1 = x_{1p} \cdot \left(\frac{Z_1}{f} + 1 \right) \qquad (2.33)$$

Todas las ecuaciones mostradas consideran al centro del Plano de Proyección como origen para el sistema de referencia cartesiano respectivo. Sin embargo, las imágenes adquiridas por las cámaras no tiene el origen en el centro de la imagen, si no que en la esquina superior izquierda. Es por esto que se hace necesario realizar una traslación de las coordenadas de los píxeles referenciados al origen de la imagen hacia el origen del Plano de Proyección. En la siguiente figura, se puede apreciar que debido a la resolución espacial de 320x240 píxeles seleccionada como resolución de trabajo de las cámaras, el centro de la imagen (160,120) corresponde al origen del Plano de Proyección.

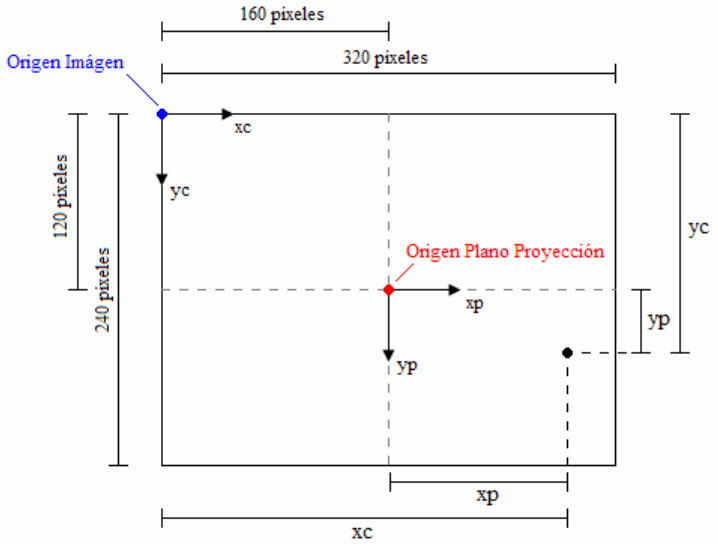


Figura 2.30

Por lo tanto, para un punto cualquiera perteneciente a la imagen adquirida, la traslación entre los sistemas viene determinada por:

$$xp = xc - 160 \tag{2.34}$$

$$yp = yc - 120 \tag{2.35}$$

Donde:

(xc, yc): Coordenadas del punto respecto al origen de la imagen.

(xp, yp): Coordenadas del punto respecto al origen del Plano de Proyección.

CAPITULO III: *El Modelo Cinemático del Robot*

La Cinemática es el estudio del movimiento de cuerpos articulados con respecto a un sistema de referencia sin tener en cuenta sus causas. En el campo de los manipuladores robotizados (o brazos robots), es utilizada para la determinación de relaciones entre la posición y orientación del extremo del robot y los ángulos existentes en sus articulares. El modo de determinar estas relaciones dependerá de las variables que se dispongan y de cuales se necesita conocer. Para ello, la Cinemática se divide en Cinemática Directa y Cinemática Inversa. La Cinemática Directa consiste en determinar la posición y orientación del extremo del robot, con respecto a un sistema de coordenadas de referencia, conocidos los valores de las articulaciones, y la Cinemática Inversa consiste en determinar los ángulos de deben adoptar las articulaciones para alcanzar una posición y orientación conocida.

Existen varios métodos para la determinación de tales relaciones, siendo los más populares y tradicionales: el Método Geométrico, que se basa en relaciones geométricas y trigonométricas de los elementos del robot; y el Método de Transformaciones Homogéneas de Matrices (Denavit-Hartenberg), que asocia a cada elemento o “hueso” un sistema de referencia distinto.

3.1 Modelo Cinemático Inverso

Haciendo una abstracción de la geometría del robot se puede obtener, por simple inspección visual, el modelo cinemático del robot. Tal modelo cinemático se muestra a continuación.

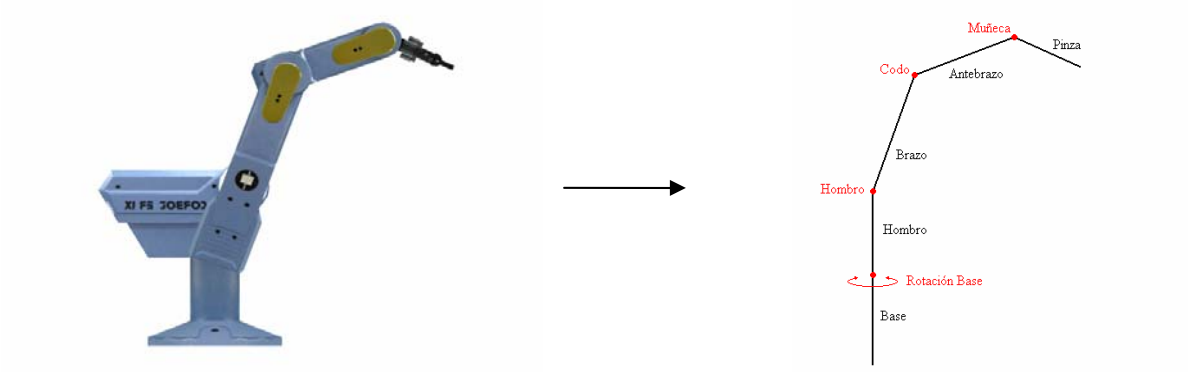


Figura 3.1

En la figura 3.1, los puntos rojos representan las articulaciones del robot, mientras que las líneas negras representan a los eslabones o “huesos” que hay entre las articulaciones.

El método utilizado para la obtención de las ecuaciones que rigen al modelo cinemático inverso fue el Método Geométrico, debido a la simplicidad que presenta este modelo. Sin embargo, este método solo sirve para obtener las ecuaciones de solución de las primeras variables articulares del robot. Por lo que solo es posible conocer los ángulos relacionados a las articulaciones *Rotación Base*, *Hombro* y *Codo*, conocidas las coordenadas espaciales de la articulación *Muñeca*. Aunque a primera vista esto parece una limitante para el desarrollo del sistema, en realidad no lo es, debido a que es posible conocer también el ángulo *Muñeca* gracias a las coordenadas espaciales de las marcas del guante Señalizador obtenidas por el sub-sistema de *Visión Artificial*.

La solución de una parte del modelo cinemático de figura anterior se realiza en base a la figura 3.2, donde se utiliza el Método Geométrico y se toma como sistema de referencia la articulación *Hombro.*

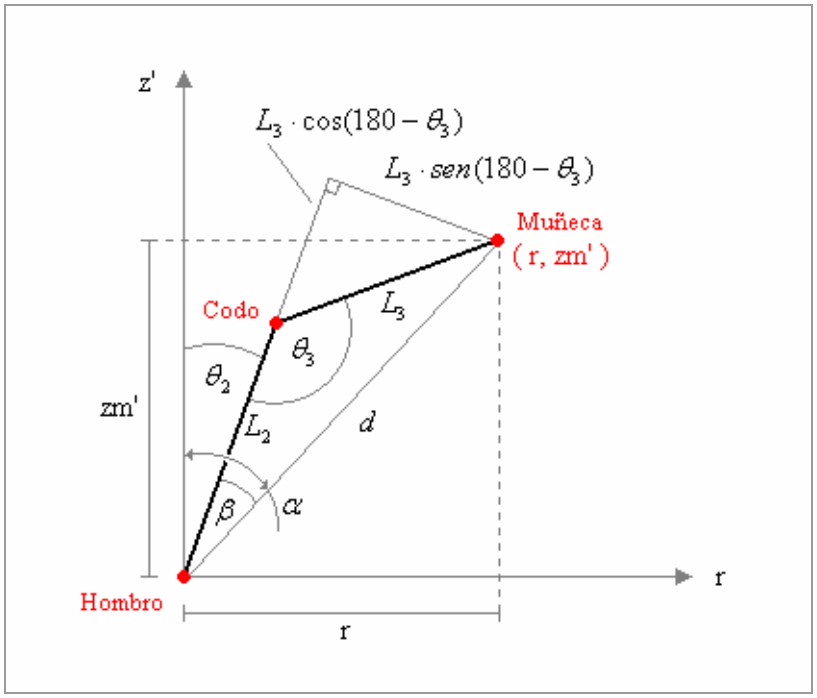


Figura 3.2

Las coordenadas de las marcas de la muñeca de *Guante Señalizador* se asocian a la *Muñeca* del Modelo Cinemático y conocidas las dimensiones físicas del robot correspondientes es posible determinar todos los ángulos mostrados.

Primero que nada, se calcula la distancia entre las coordenadas de la muñeca y el origen del sistema de referencia, debido a que esta magnitud será utilizada en ecuaciones posteriores.

$$d = \sqrt{r^2 + zm'^2} \tag{3.1}$$

Luego, utilizando la "ley de los cosenos" se averigua la ecuación que rige el comportamiento de θ_3.

$$d^2 = {L_2}^2 + {L_3}^2 - 2 \cdot L_2 \cdot L_3 \cdot \cos(\theta_3) \tag{3.2}$$

$$\theta_3 = \cos^{-1}\left(\frac{{L_2}^2 + {L_3}^2 - d^2}{2 \cdot L_2 \cdot L_3} \right) \tag{3.3}$$

Conocido θ_3, solo queda por conocer θ_2. Sin embargo, la determinación de una ecuación que modele su comportamiento no es trivial debido a que no se conocen las coordenadas de la articulación *Codo*. Además, θ_3 solo indica la apertura que hay entre los huesos *Brazo* y *Antebrazo.* Para solucionar esta problemática, se utilizan dos variables auxiliares: α y β. La primera variable, mide el ángulo que existe entre el eje vertical z' y el vector distancia d. Y la segunda mide el ángulo entre el hueso *Brazo* y el vector distancia *d,* siendo este último ángulo independiente de θ_2.

El ángulo α se calcula mediante:

$$\alpha = tg^{-1}\left(\frac{r}{zm'} \right) \tag{3.4}$$

Para calcular β se toma como eje de referencia la distancia d, para luego descomponer la longitud del hueso L_3 en dos distancias ortogonales: $L_3 \cdot \cos(180 - \theta_3)$ y $L_3 \cdot sen(180 - \theta_3)$. Esto presenta la ventaja de que la orientación de $L_3 \cdot \cos(180 - \theta_3)$ siempre será la misma que L_2, por lo que siempre se podrán sumar. Así pues, la determinación de β se puede realizar por medio de:

$$\beta = tg^{-1}\left(\frac{L_3 \cdot sen(180 - \theta_3)}{L_2 + L_3 \cdot \cos(180 - \theta_3)}\right) \tag{3.5}$$

Conocidos α y β, θ_2 se calcula mediante:

$$\theta_2 = \alpha - \beta \tag{3.6}$$

Aunque las ecuaciones obtenidas describen completamente el comportamiento de este sub-modelo, aún falta relacionarlas con el modelo original, además de determinar las ecuaciones para el resto del modelo. La siguiente figura muestra el modelo cinemático original sin la pinza y con las variables que quedan por determinar.

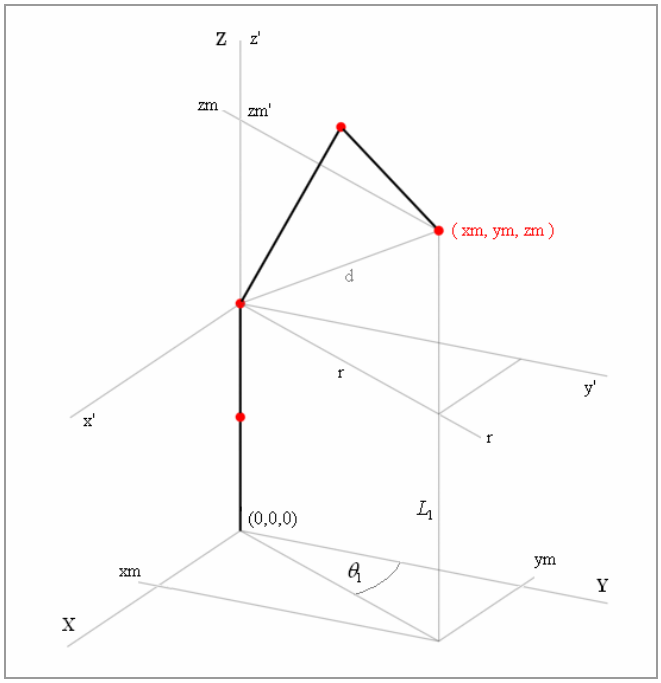


Figura 3.3

En esta figura se puede apreciar que las ecuaciones que faltan por determinar están relacionadas con el ángulo θ_1, que corresponde a la rotación de todos los huesos. El cálculo de este ángulo se basa en las coordenadas de la muñeca proyectadas en el plano XY, por ello:

$$\theta_1 = tg^{-1}\left(\frac{xm}{ym}\right) \tag{3.7}$$

Por lo anterior, la coordenada *r* de la muñeca en la figura 3.2, en realidad, es dependiente de las coordenadas xm e ym.

$$r = \sqrt{xm^2 + ym^2} \tag{3.8}$$

Y por último, se debe trasladar verticalmente el sistema de la figura 3.2 para que quede como el mostrado en la figura 3.3. Para ello, solo se debe sumar la longitud de los huesos *Base* y *Hombro.*

$$zm' = zm - L_1 \tag{3.9}$$

3.2 Determinación de la Inclinación (Pitch) de la Muñeca

La determinación de la posición angular asociada a la articulación *Muñeca* se hizo de manera separada para aprovechar las propiedades que presentan la ubicación de las marcas en el *Guante Señalizador*. Dependiendo de la postura que tenga la mano del usuario, es posible la determinación de su inclinación basándose en la posición espacial de las marcas asociadas a la muñeca y los dedos del usuario. Esto, sobre la consideración de un eje vertical común, que actúa como eje de referencia entre la muñeca del modelo cinemático y la muñeca del *Guante Señalizador*. La siguiente figura muestra como se realiza este enlace.

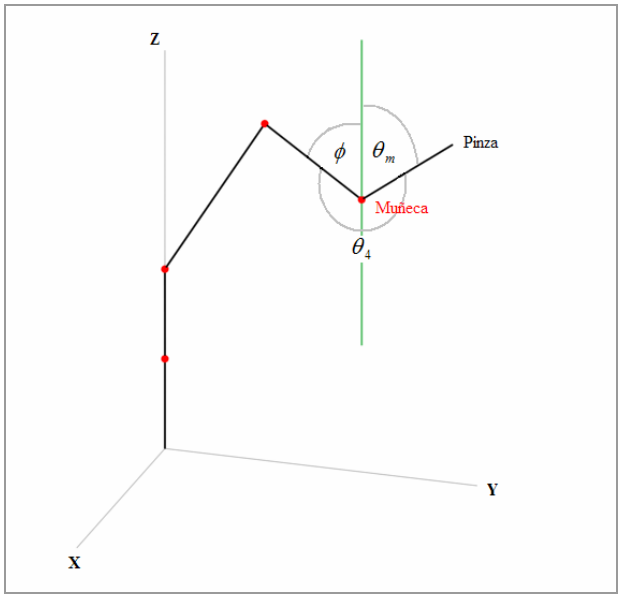


Figura 3.4

De la utilización de este eje de referencia, se desprenden dos variables auxiliares: ϕ, que indica la inclinación que tiene el hueso *Antebrazo* respecto al eje de referencia producto de la acción de la Cinemática Inversa; y θ_m, que indica la inclinación que tiene el hueso *Pinza* producto de las coordenadas espaciales de la muñeca y el punto medio entre los dedos del *Guante Señalizador*. Para determinar las expresiones que permiten conocer las magnitudes de estas variables fue necesario utilizar dos sistemas de referencia auxiliares, uno para cada variable y centrados en la articulación *Muñeca*. La siguiente figura muestra el sistema de referencia local para θ_m.

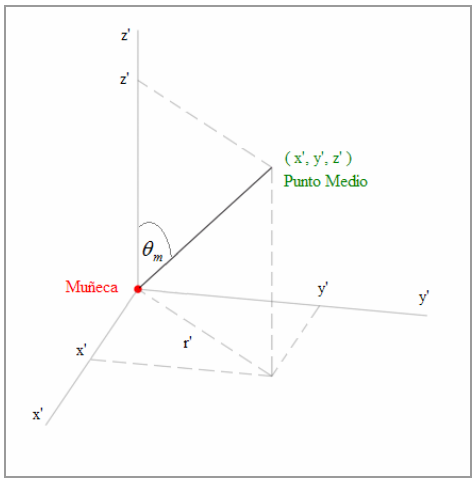


Figura 3.5

Debido a la utilización de un sistema de referencia local, es necesario establecer expresiones que permitan transformar las coordenadas del sistema de referencia global hacia el sistema de referencia local de θ_m. Tales expresiones se expresan a continuación:

$$x' = x_{puntomedio} - x_{muñeca} \tag{3.10}$$

$$y' = y_{puntomedio} - y_{muñeca} \tag{3.11}$$

$$z' = z_{puntomedio} - z_{muñeca} \tag{3.12}$$

Por lo tanto, las ecuaciones que describen el comportamiento de θ_m se muestran a continuación:

$$r' = \sqrt{x'^2 + y'^2} \tag{3.13}$$

$$\theta_m = tg^{-1}\left(\frac{r'}{z'}\right) \tag{3.14}$$

Por otra parte, de manera análoga se realiza la determinación de ϕ. La figura 3.6 muestra el sistema de referencia local para ϕ.

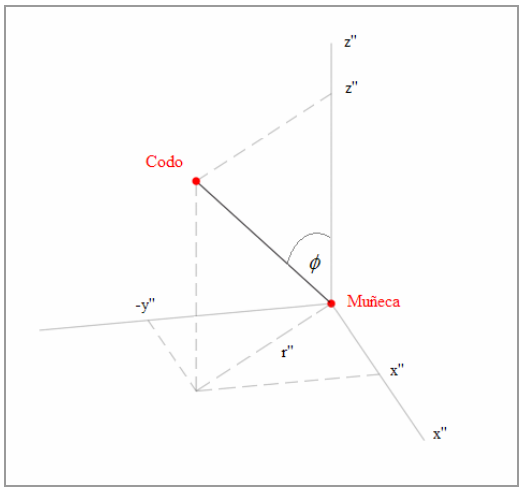


Figura 3.6

Todas ecuaciones que están relacionadas con este sistema son similares al sistema de referencia local de θ, y son mostradas a continuación.

$$x'' = x_{muñeca} - x_{codo} \tag{3.15}$$

$$y'' = y_{codo} - y_{muñeca} \tag{3.16}$$

$$z'' = z_{codo} - z_{muñeca} \tag{3.17}$$

$$r'' = \sqrt{x''^2 + y''^2} \tag{3.18}$$

$$\phi = tg^{-1}\left(\frac{r''}{z''}\right) \tag{3.19}$$

Así pues, ya conocidas las expresiones que permiten conocer las magnitudes de las variables auxiliares, solo falta determinar la expresión que permita conocer θ_4. Considerando los ángulos mostrados en la figura 3.4, la sumatoria de ellos es equivalente a 360º, por lo tanto:

$$\theta_4 = 360 - \phi - \theta_m \tag{3.20}$$

Cabe destacar que esta forma de obtener la inclinación de la Pinza tiene la ventaja de que es independiente de la dirección que tenga, es decir, no es necesario que el usuario disponga su mano en forma alineada con el resto de los huesos del modelo cinemático del robot.

3.3 Obtención de Parámetros Físicos del Robot

Los Parámetros Físicos de Robot son utilizados por las ecuaciones que rigen el comportamiento del Modelo Cinemático Inverso. Debido a esto, son los parámetros los que permiten obtener magnitudes numéricas coherentes con las magnitudes físicas de la realidad. Sin ellos, no es posible determinar si los resultados de las ecuaciones corresponden a la realidad. Dentro de los parámetros que fueron necesarios averiguar están las dimensiones de los “huesos” de robot, las cuentas máximas de los encoders asociados a cada articulación y las aperturas angulares máximas a las que puede llegar cada una.

3.3.1 Longitudes de los Elementos o "Huesos"

La longitud de un elemento o "hueso" corresponde al largo de cada sección que se encuentra entre dos articulaciones. No debe confundirse con el largo de la geometría de cada sección, debido a que el dato que interesa es la distancia entre dos articulaciones, ya que es esta la magnitud que se utiliza en el Modelo Cinemático Inverso. La siguiente figura muestra las cotas en donde se realizaron las mediciones.

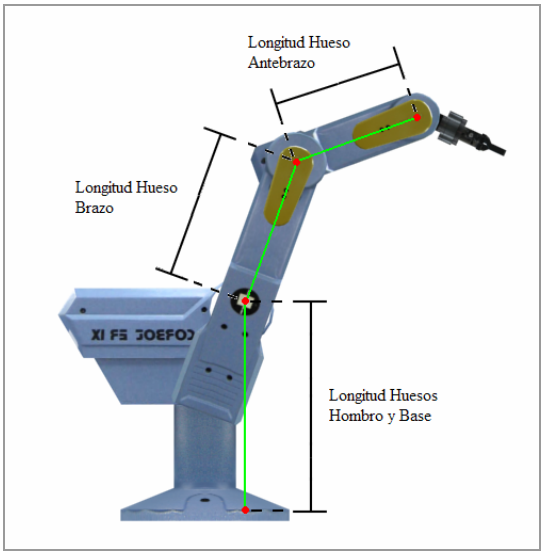


Figura 3.7

Así pues, la siguiente tabla resume las longitudes de los huesos del robot.

Tabla 3.1

Hueso	Longitud
Antebrazo	23 cms.
Brazo	28 cms.
Hombro y Base	40,6 cms.

Cabe mencionar que la longitud de la *Pinza* no se considera debido a que más adelante en el texto se verá que la longitud de esta pieza no influye en la determinación de su inclinación.

3.3.2 Cuentas Máximas y Mínimas de los Codificadores

Los Codificadores (*Encoders*) son dispositivos ópticos que se utilizan en la medición de movimientos lineales o circulares. En el caso del robot Scorbot ER-IX es utilizado para determinar el desplazamiento angular que ha realizado cada articulación. Ver siguiente figura.

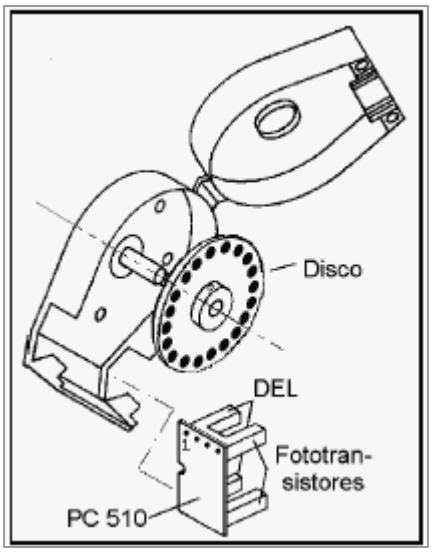


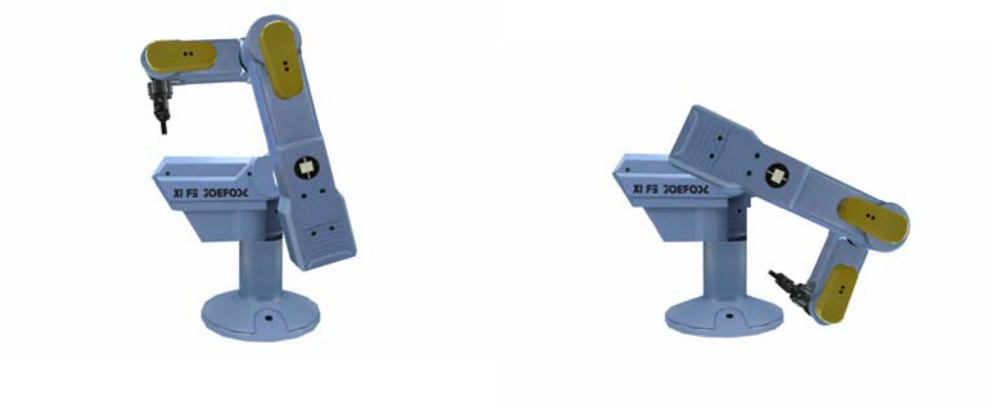
Figura 3.8

Su construcción consta de tres componentes principales: Un diodo emisor de luz infrarroja, un fototransistor receptor de luz infrarroja y una rueda dentada o perforada. El más básico de los funcionamientos consiste en que el movimiento circular que se desee medir, se acopla al eje de la rueda dentada, la cual cada vez que interrumpe el rayo de luz infrarroja hará incrementar la cuenta en una unidad. Existen otras variantes de este sistema de conteo como el 4F, que tiene una resolución 4 veces mayor que el sistema mencionado. Además los codificadores, dependiendo del modelo, permiten conocer la sentido giro en que se ha realizo la rotación.

La determinación de las Cuentas Máximas y Mínimas se debe a la necesidad de asociar estos parámetros a las aperturas angulares máximas que puede lograr el robot en cada articulación. Como se verá más adelante en el texto, una vez que el Modelo Cinemático Inverso haya resuelto los ángulos que debe adoptar cada articulación para llegar a la posición espacial indicada, se deberá hacer la conversión de unidades angulares a unidades de “cuentas del

codificador" ya que solo de esta manera pueden ser especificadas posiciones rotacionales en el robot.

Para realizar tales mediciones, se llevó al robot a las posiciones angulares máximas y mínimas que podía adquirir en cada articulación como muestra la siguiente figura.



Posición Angulares Mínimas Posición Angulares Máximas

Figura 3.9

Luego, mediante el programa *Advanced Terminal Software (ATS)*, ver figura 3.10, se consultó al controlador del robot las cuentas de los codificadores correspondientes a las posiciones angulares extremas alcanzadas.

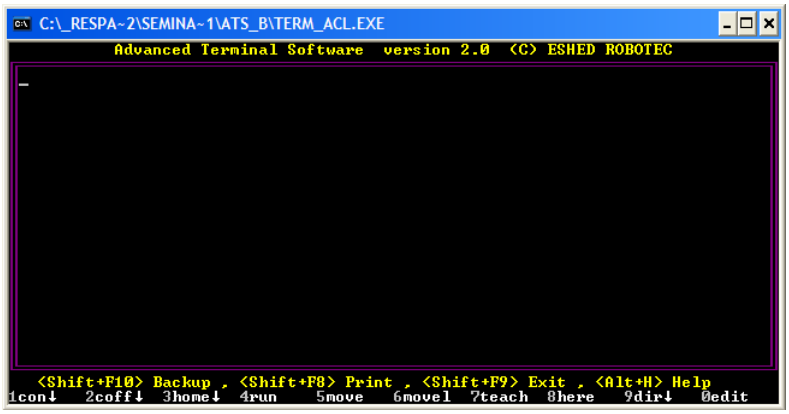


Figura 3.10

La siguiente tabla resume las cuentas del codificador correspondientes a las posiciones angulares máximas alcanzadas por cada articulación.

Tabla 3.2

Eje	Articulación	Cuenta Mínima	Cuenta Máxima
1	Rotación Base	-190.056	138.748
2	Hombro	-74.854	128.283
3	Codo	-115.523	139.049
4	Muñeca	-1.056	200.075

3.3.3 Posiciones Angulares Máximas y Mínimas

Las Posiciones Angulares Máximas y Mínimas corresponden a los ángulos máximos y mínimos que pueden existir en cada articulación del robot. Estas magnitudes son análogas a las Cuentas Máximas de los Codificadores; de hecho están directamente relacionadas.

El método de determinación de estas magnitudes es distinto al anterior, ya que no es necesario realizar mediciones debido a que toda la información necesaria para llevar a cabo los cálculos está disponible en las hojas de especificaciones del robot. Así pues, en base a estos datos, que son mostrados en el Apéndice, los cálculos realizados se muestran a continuación.

Para disponer de un ángulo de referencia para cada articulación, se estableció la siguiente posición como “Origen Vertical”.

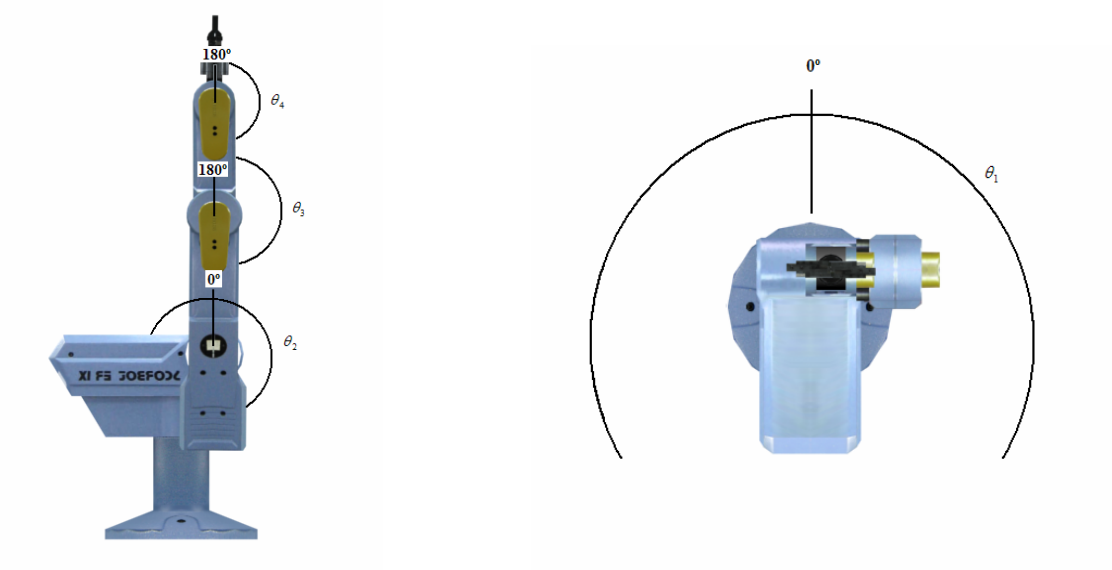


Figura 3.11

El objetivo de disponer al robot en esta posición es poder asignar a cada articulación un ángulo de referencia fijo asociado a las cuentas de los codificadores correspondientes a tal postura, que permita conocer los desplazamientos circulares máximos en unidades angulares basados en las cuentas máximas de los codificadores. Así pues, la siguiente tabla resume los ángulos "origen" para cada articulación:

Tabla 3.3

Eje	Articulación	Origen Vertical (º)	Origen Vertical (cuentas)
1	Rotación Base	0º	-25.654
2	Hombro	0º	-56.752
3	Codo	180º	30.517
4	Muñeca	180º	101.583

No todos los ángulos de referencia fueron seleccionados como 0º debido a que en las articulaciones *Codo* y *Muñeca*, cuando están en la posición "Origen Vertical", en el modelo cinemático corresponden a una apertura de 180º.

Una vez conocidos los ángulos referenciales y las cuentas referenciales de los codificadores, el cálculo de los desplazamientos angulares máximos es sencillo y se detalla a continuación.

Conocida la apertura angular máxima para cada articulación (ver Apéndice) es posible determinar las relaciones que existen entre las cuentas del codificador y las magnitudes angulares.

$$m = \frac{\Delta enco}{\Delta \theta} \tag{3.21}$$

Considerando que *m* se puede expresar como:

$$m = \frac{enco_2 - enco_1}{\theta_2 - \theta_1} \tag{3.22}$$

Se despeja la variable de interés θ_2:

$$\theta_2 = \frac{(enco_2 - enco_1)}{m} + \theta_1 \tag{3.23}$$

Reemplazando los valores correspondientes de cada articulación en la ecuación anterior, la siguiente tabla sintetiza las posiciones angulares máximas y mínimas.

Tabla 3.4

Eje	Articulación	Posición Angular Mínima	Posición Angular Máxima
1	Rotación Base	-135º	+135º
2	Hombro	-12,92º	+132,08º
3	Codo	+59,53º	+269,53º
4	Muñeca	+79,98º	+275,98º

3.4 Utilización del Modelo Cinemático Inverso en la Resolución de Posiciones

La utilización del Modelo Cinemático Inverso para la resolución de las posiciones angulares es simple, y consiste básicamente en que obtenidas las posiciones espaciales de las marcas del *Guante Señalizador*, estas se suministran como datos de entrada al modelo cinemático. Una vez que el modelo cinemático haya determinado que postura debe adoptar el robot para alcanzar el punto especificado, entrega las posiciones angulares que debe tener cada articulación. La siguiente figura muestra esquemáticamente este proceso.

Figura 3.12

Sin embargo, debido a que el Modelo Cinemático es un modelo matemático, las magnitudes que se generan de él son magnitudes en unidades angulares y no pueden ser transmitidas directamente al controlador del robot sin antes ser convertidas a cuentas del codificador. Para ello, utilizando la ecuación (3.22) se obtiene una expresión que permite realizar la conversión de unidades:

$$enco_2 = m \cdot (\theta_2 - \theta_1) + enco_1 \qquad (3.24)$$

Donde:

$enco_1$:	Cuentas del codificador correspondiente a la posición “Origen Vertical”.
$enco_2$:	Cuentas del codificador proporcionales a la magnitud angular transformada.
m:	Relación entre las cuentas del codificador y la apertura angular máxima de la articulación.
θ_1:	Posición angular correspondiente a la posición “Origen Vertical”.
θ_2:	Posición angular que se va a transformar.

Así pues, esta expresión es aplicada en cada una de las magnitudes angulares que entrega el modelo cinemático.

CAPITULO IV: *Comunicación*

El bloque de *Comunicación* es el encargado de establecer y mantener la comunicación con el controlador del robot. Su objetivo es transmitir las posiciones angulares de cada articulación para que el robot alcance la posición espacial indicada por la mano del usuario. Para ello utiliza el puerto de comunicaciones serial RS-232C, ya que este es el único medio físico de interconexión existente entre el computador y el controlado del robot.

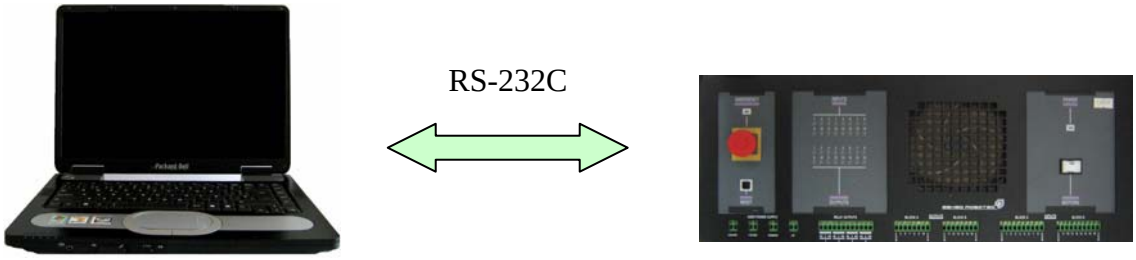
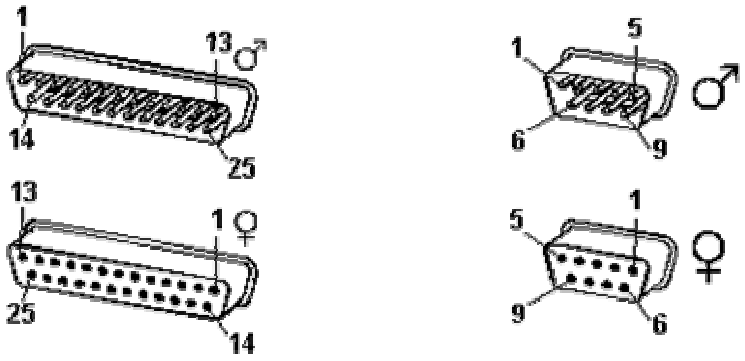


Figura 4.1

En el computador, el sistema de comunicación utiliza las API (*Application Programming Interface)* de Windows para poder acceder al puerto de comunicaciones serial RS-232C, conocido también con el nombre de puerto COM. Por otra parte, en el controlador del robot, el sistema utiliza un programa residente que espera e interpreta los datos enviados por el computador para luego realizar el movimiento. Tal programa residente esta escrito en lenguaje ACL (*Advanced Controller Languaje)*, lenguaje que es parte del controlador del robot.

4.1 El Puerto Serie RS-232C

El puerto serie RS-232C es un estándar que constituye la tercera revisión de la antigua norma RS-232 propuesta por la EIA (*Asociación de Industrias Electrónicas*) y es una de las formas más comúnmente utilizada para realizar transmisiones de datos entre dispositivos digitales. Consiste en un conector que tiene dos versiones: el conector tipo DB-25 de 25 pines y el DB-9 que consta de 9 pines que son mostrados a continuación:



Conector tipo DB-25 Conector tipo DB-9

Figura 4.2

Las señales con las que trabaja son digitales, de +12V (0 lógico) y -12V (1 lógico) para la entrada y salida de datos, para las señales de control utiliza la lógica inversa. La siguiente tabla resume todos los pines que hay para el conector tipo DB-9:

Tabla 4.1

Pin	Función	Nombre Abreviado
1	Carrier Detect	CD
2	Receive Data	RX
3	Transmit Data	TX
4	Data Terminal Ready	DTR
5	Signal Ground	SG
6	Data Set Ready	DSR
7	Request To Send	RTS
8	Clear To Send	CTS
9	Ring Indicator	RI

Las señales TX, DTR y RTS son de salida, mientras que RX, DSR, CTS y CD son de entrada. La tierra de referencia para todas las señales es SG (*Signal Ground*). El resto de las señales, CD y RI, se utilizaron bastante cuando los computadores disponían de un modem externo ya que estas señales permitian saber cuando había una llamada entrante (*RI*) o si la línea telefónica disponía de tono (*CD*).

En los computadores, el puerto serial es controlado por un circuito integrado llamado UART 16550A (*Transmisor-Receptor Asíncrono Universal*). Para comunicarse con este circuito integrado y tener el control del puerto serie, la CPU del computador emplea direcciones de puertos de E/S y líneas de interrupción IRQ. Comúnmente para el COM1 corresponde E/S: 3F8h, IRQ: 4; y para el COM2 E/S: 2F8h, IRQ: 3. Gracias a este tipo de configuración, es posible intercambiar datos con la CPU por medio de los puertos de E/S, mientras que las IRQ producen una interrupción para indicar que ha ocurrido un evento, por ejemplo, que ha llegado un dato o ha cambiado el estado de alguna de las señales de entrada.

La RS-232 puede transmitir datos en grupos de 5, 6, 7 u 8 bits, a velocidades configurables, generalmente a 9600 bps. Además puede ser agregado, opcionalmente, un bit de paridad que indica si el número de bits transmitidos es par o impar, con el objetivo de detectar fallos en la transmisión. A todo lo anterior, se le agregan los bits de parada que pueden ser uno o dos y el tipo de control de flujo de los datos, que puede ser por software (*XON/XOFF*) o por hardware (*RTS/CTS*). En general, el protocolo más comúnmente utilizado es el 8N1 con control de flujo RTS/CTS.

Sin embargo, debido a que el estándar RS-232C no permite indicar ni detectar que protocolo se está utilizando, es necesario establecer en las maquinas que se van a comunicar el mismo protocolo antes de realizar la comunicación.

4.2 La Interconexión Null-Modem

Respecto a la interconexión entre el computador y el controlador del robot, ésta es de tipo NULL-MODEM de 3 hilos con protocolo de control de flujo por hardware emulado, debido a que el controlador del robot trabaja sin protocolo de control de flujo. La siguiente figura muestra este tipo de conexión:

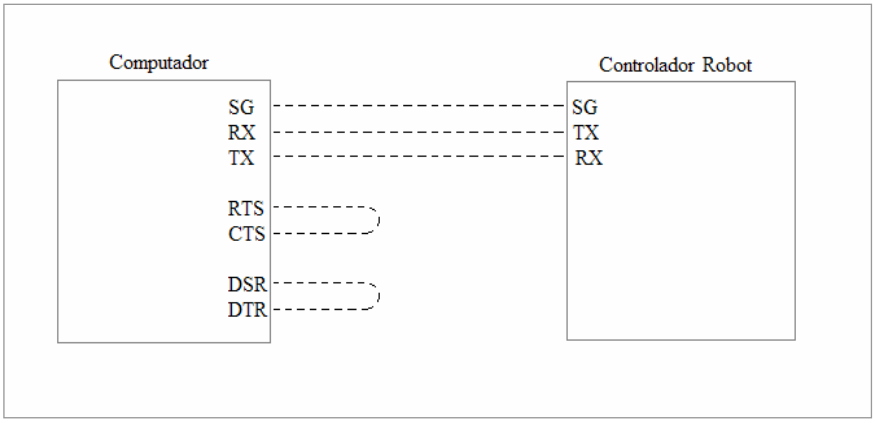


Figura 4.3

En este tipo de conexión, aun que el computador este configurado para trabajar en protocolo de control de flujo por hardware, será transparente para el Controlador del Robot, debido a que el computador a si mismo se "responderá" las señales que envié por los pines RTS y DTR. Sin embargo, al protocolo XON/OFF tiene la desventaja de que no se pueden realizar transmisiones binarias quedando limitada solamente a transmisiones tipo ASCII.

El controlador del robot trabaja solamente en transmisiones tipo ASCII debido a que su sistema operativo permite comunicarse con él mediante cualquier programa que pueda enviar y recibir caracteres por el puerto serial, por ejemplo el "*HyperTerminal*" de Windows. La siguiente figura muestra un ejemplo de comunicación entre este programa y el controlador del robot.

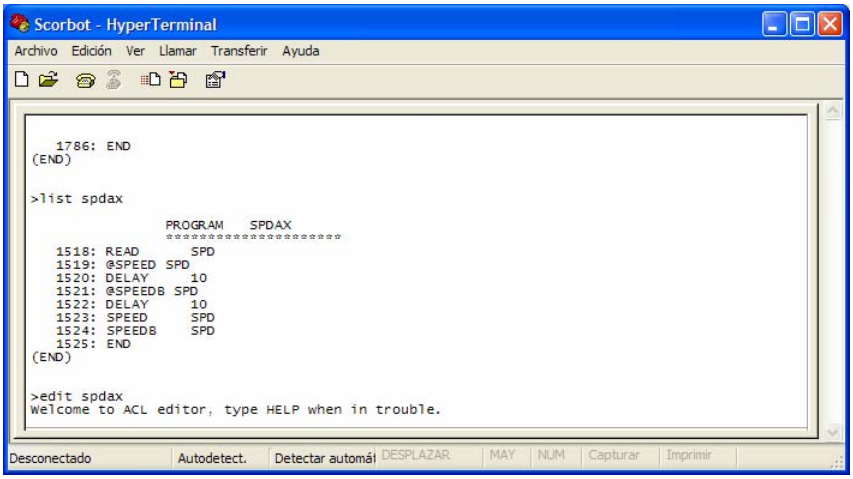


Figura 4.4

4.3 La API de Comunicación de Windows

La API (*Application Programming Interface*) de Comunicación de Windows es un set de funciones que permiten al programador acceder a los recursos de comunicación de sistema operativo, donde los recursos de comunicación pueden ser dispositivos físicos o lógicos. Dentro de los dispositivos físicos están los puertos serie, puertos paralelos, faxs y módems.

Desde la aparición Windows 2000 y posteriores, Microsoft Windows no permite a las aplicaciones acceder directamente al hardware como se podía realizar en sus versiones anteriores como Windows 98. Esto, bajo el alero de ofrecer al usuario una mayor seguridad y estabilidad en el sistema operativo. Así pues, aplicaciones que utilizaban el puerto serie directamente mediante los puertos de E/S son incompatibles. En compensación a esta problemática, Microsoft Windows ofrece al usuario las API de Comunicación, que se encargan internamente de toda la problemática de administración de puertos de E/S y líneas de interrupción IRQ, ofreciendo al usuario la ventaja de abstraerse del tipo de hardware que tenga instalado un computador. La siguiente figura muestra a grandes rasgos la idea explicada.

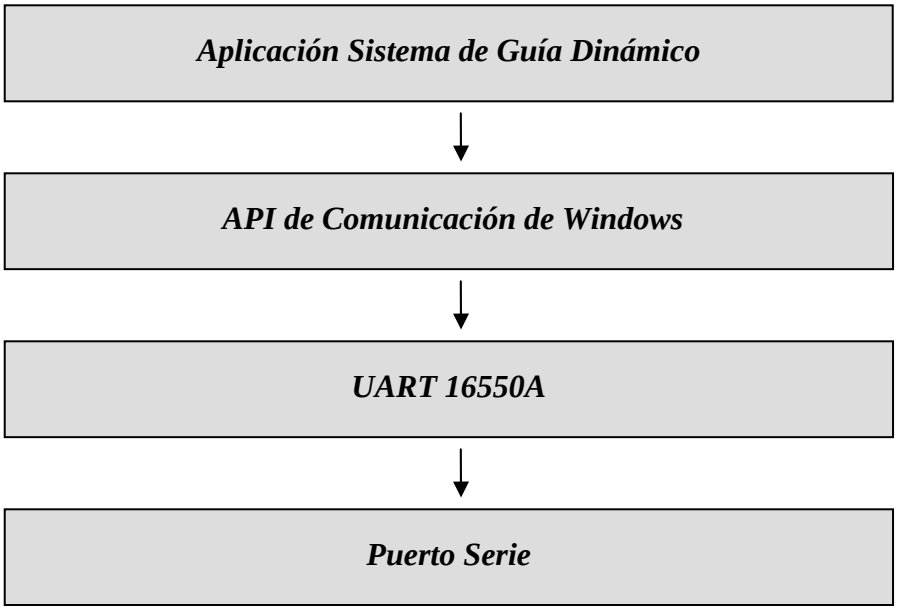


Figura 4.5

4.4 Programación y Utilización del Puerto COM

Antes de poder realizar alguna transmisión por el puerto serie de comunicaciones es necesario programarlo para especificar la velocidad, la cantidad de bits por símbolo a utilizar, si se usará paridad, cuantos bits de parada y el tipo de control de flujo que se utilizará en la transmisión. Para ello es necesario conocer que funciones de la API de Comunicaciones de Windows que permiten realizar tal configuración. A continuación, se nombran y explican brevemente las tales funciones.

CreateFile(): Crea o abre un archivo. Utilizada para abrir el puerto de comunicaciones como si fuera un archivo con permisos de escritura y lectura, pero de uso exclusivo por la aplicación *Sistema de Guiamiento Dinámico*.

CloseHandle(): Cierra un archivo previamente abierto o creado por *CreateFile()*. Permite cerrar el puerto de comunicaciones

GetCommState(): Obtiene la configuración actual del puerto de comunicaciones.

SetCommState(): Permite especificar una configuración especifica para el puerto de comunicaciones.

WriteFile(): Escribe los datos contenidos en un buffer de memoria hacia el archivo previamente abierto por CreateFile(). Permite transmitir los datos o comandos hacia el puerto serie

ClearCommError(): Recupera información sobre actual estado del puerto. Utilizada para conocer cuantos bytes han sido recibidos.

ReadFile(): Lee los datos contenidos en un archivo y lo almacena en un buffer de memoria. Permite recuperar los datos que han sido recibidos y almacenados temporalmente en el buffer de la USART.

Debido a que estas funciones son básicas respecto del objetivo de transmitir las posiciones angulares de cada articulación, fue necesario crear un set de funciones de mayor nivel que permitieran simplificar las tareas de configuración, transmisión y recepción de datos. Tales funciones son: *AbrirPuerto(), CerrarPuerto(), EnviarComando(), Recibir() y WaitForRead()*. El algoritmo escrito en cada función se detalla a continuación:

AbrirPuerto(): Mediante *CreateFile()* se abre el puerto COM como si fuera un archivo con atributos de escritura y lectura. Luego, a través de *GetCommState()* llenar la estructura *DCB* obteniendo la configuración actual que tiene el puerto. En esta estructura, solamente se modifican las variables de interés, que según los parámetros de comunicación del controlador del robot corresponden a: Velocidad de Transmisión de 9600 bps, Modo ASCII, Sin Bit de Paridad, Un Bit de Parada y Símbolos de 8 bits. Una vez realizada la modificación a la estructura DCB se actualiza la configuración del puerto COM a través de *SetCommState()*.

CerrarPuerto(): Llama a la función CloseHandle() que simplemente cierra el archivo asociado al puerto COM que fue abierto con *CreateFile()*.

EnviarComando(): Toma como argumento una cadena de caracteres que serán transmitidos, a la cual se le agrega al final el carácter de control 0Dh para indicar al controlador el final del comando o dato. Luego, esta cadena formateada se envía al puerto COM mediante la función *WriteFile()*.

Recibir(): Esta función tiene como parámetros de entrada un puntero hacia el buffer en donde se almacenarán los datos que han sido recepcionados en el buffer temporal del puerto COM, y un puntero a la variable que indicará la cantidad de bytes leídos. Para ello, utiliza la función *ClearCommError(),* para preguntar cuantos bytes hay que leer del buffer del puerto COM y chequear si ha ocurrido algún error en la transmisión. Si ocurrió alguno, retorna con un código de error, si no, procede a la lectura de los datos mediante la función *ReadFile()*.

WaitForRead(): Esta constantemente llamando a la función *Recibir()* hasta que un dato haya sido recibido. Luego busca en el buffer si existe el carácter "?", si lo encuentra *WaitForRead()* finaliza, de lo contrario vuelve a llamar a la función *Recibir()*. El objetivo de esta función es

esperar por el carácter “?”, que es el carácter que transmite el controlador del robot cada vez que está listo para recibir el dato correspondiente al ángulo de una articulación.

Así pues, a través de estas funciones el proceso de programación se simplifica por lo que cada vez que el sistema de *Visión Artificial* requiera abrir o cerrar, dependiendo de la cantidad de objetos detectados, ejecutará la función *EnviarComando(“open”)* o *Enviar Comando(“close”).* De manera análoga, cuando el *Sistema de Marcas* (explicado en el Capitulo V) detecte que el usuario realizó una marca de punto con el *Guante Señalizador*, este sistema ejecutará las funciones *WaitForRead(),* para esperar sincronismo con el programa residente en el controlador y *EnviarComando(posición_articular_n),* para especificar la posición angular de la articulación *n*. Cabe mencionar que este par de funciones se ejecutan cuatro veces por cada punto marcado, una por cada articulación a actualizar.

4.5 El Lenguaje ACL

El Lenguaje ACL (*Lenguaje de Control Avanzado*), es un lenguaje y entorno de programación de robótica avanzado, multitarea, desarrollado por la empresa ESHED ROBOTEC.

El ACL está almacenado en memorias tipo EPROM dentro del controlador del robot y se puede acceder a él con cualquier computador PC estándar o terminal, por medio del puerto de comunicaciones RS-232.

Dentro de las características del ACL se incluyen:

- Ejecución directa de comandos.
- Control de entrada y salida de datos.
- Programación usuario del robot.
- Ejecución simultanea de programas.
- Ejecución sincronizada de programas.
- Sencilla gestión de archivos.

4.5.1 Creación del Programa Intérprete

El programa interprete es un programa escrito en lenguaje ACL que se ejecuta en el controlador del robot mientras esté en operación el *Sistema de Guía Dinámico*. De hecho, este último es quien administra la inicialización y finalización del programa residente.

Es un programa sencillo que tiene como objetivo recibir las posiciones angulares enviadas por el computador a través del puerto de comunicaciones serie para luego mover el extremo del robot según la postura indicada.

Debido a que el sistema de posicionamiento del robot funciona en base a puntos espaciales programados previamente antes de realizar el movimiento, el programa residente basa su operación en dos de estos puntos, uno correspondiente para el grupo A (brazo robot) y otro para el grupo B (slider). Así pues, el programa residente espera que el computador le envié las posiciones angulares de todas las articulaciones para luego ingresarlas a la estructura de datos que tiene cada punto. Luego, le ordena al robot que realice el movimiento hacia las últimas posiciones angulares que fueron ingresadas en los puntos mencionados. El código escrito para este algoritmo se muestra a continuación.

```
DEFINE AXIS1
DEFINE AXIS2
DEFINE AXIS3
DEFINE AXIS4
DEFINE AXIS7
LABEL    0
READ     GRUPO
IF       GRUPO = 0
         READ     MODO
         READ     AXIS1
         READ     AXIS2
         READ     AXIS3
         READ     AXIS4
```

```
        SETPV    POSA 1 AXIS1
        SETPV    POSA 2 AXIS2
        SETPV    POSA 3 AXIS3
        SETPV    POSA 4 AXIS4
        SETPV    POSA 5 AXIS5
        MOVE     POSA
ENDIF
IF      GRUPO = 1
        READ     AXIS7
        SETPV    POSB 7 AXIS7
        MOVE     POSB
ENDIF
GOTO    0
END
```

4.6 Latencia de Comunicación

La Latencia de Comunicación es el tiempo en que demoran los datos en ser transmitidos entre el computador y el controlador del robot. Esta demora, básicamente se debe ancho de banda que ofrece la comunicación serial, aunque existen otros factores como tiempo en que demora el computador en generar los datos y el tiempo que le toma al controlador del robot en procesar la información.

Como se mencionó anteriormente, el controlador del robot trabaja solamente con la configuración 9600-8N1, que significa 9600 bits por segundo de velocidad, caracteres de 8 bytes, sin bit de paridad y un bit de parada. Por lo que cada byte transmitido hacia o desde el controlador tiene el siguiente formato:

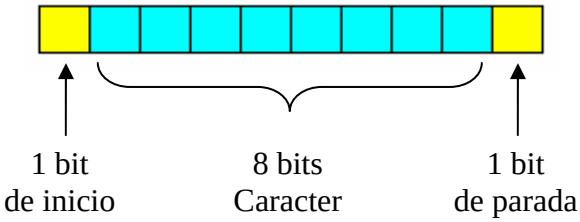


Figura 4.6

Lo que da un total de 10 bits por cada carácter transmitido.

Cada posición articular transmitida es un mensaje de 8 bytes donde: el primer byte corresponde al signo ASCII de la magnitud de las cuentas de encoder; los 6 bytes siguientes, a los dígitos ASCII de la magnitud de las cuentas de encoder; y el último byte, al carácter de control 0Dh (*Enter*) que indica el fin del mensaje. La siguiente figura muestra un esquema del formato del mensaje:

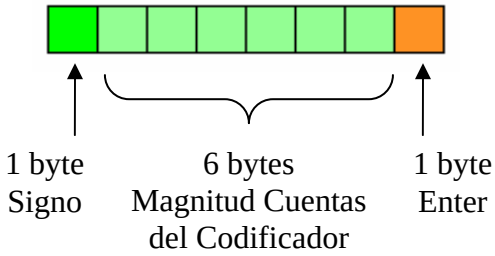


Figura 4.7

Lo que implica que por cada mensaje transmitido, el puerto serie transmite 80 bits.

Por cada marca realizada por el usuario, el sistema actualiza las posiciones angulares de todas las articulaciones lo que conlleva a que tiene que transmitir 4 mensajes. A esto se le suman el mensaje de selección de grupo (ver figura 4.8) y la espera del caracter de sincronización (*carácter "?")* que envía el controlador del robot por cada articulación actualizada.

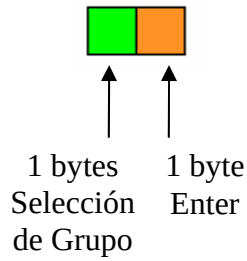


Figura 4.8

Así pues, el peor tiempo de transmisión es cuando se deben actualizar las posiciones angulares de las articulaciones del grupo A. Por lo que los mensajes que deben ser transmitidos y recepcionados son:

Sincronización	1 byte	10 bits
Grupo	2 bytes	20 bits
Sincronización	1 byte	10 bits
Articulación 1	8 bytes	80 bits
Sincronización	1 byte	10 bits
Articulación 2	8 bytes	80 bits
Sincronización	1 byte	10 bits
Articulación 3	8 bytes	80 bits
Sincronización	1 byte	10 bits
Articulación 4	8 bytes	80 bits
Total:	**39 bytes**	**390 bits**

Recordando que la velocidad de transmisión es de 9600 bits por segundo, el tiempo que demora la actualización de las posiciones angulares es de 40,6 mseg. Así pues, cada vez que el usuario realiza una marca, el tiempo de latencia que habrá antes de que el robot comience a moverse es de **40,6 mseg.** Sin embargo, esta magnitud es solo una referencia, debido a que debe considerarse también el tiempo en que demora el controlador en procesar los datos enviados, tiempo que no fue posible determinar debido a que el lenguaje ACL carece de algún comando que permita realizar esta medición.

Ya que este tiempo de latencia es considerablemente pequeño, es razonable pensar cuantas marcas se pueden realizar en un segundo. Así pues, si el proceso de marcado de puntos fuera continuo, es decir, sin esperar que el usuario deba detener su mano para realizar una marca, sólo se podrían realizar poco más 24 marcas por segundo (*1 / 40,6 mseg*). Sin embargo, esto no es posible debido a que el sistema de *Visión Artificial*, según el tipo de computador, no logra procesar más de aproximadamente 8 imágenes por segundo, por lo que la velocidad de marcación de puntos que limitada a la velocidad de procesamiento de imágenes. Así pues, la Latencia de Comunicación es un factor menor frente a la velocidad de procesamiento de imágenes.

PRUEBAS Y RESULTADOS

Para verificar la operatibilidad del sistema se realizaron dos pruebas con el objetivo de comprobar que el funcionamiento de los bloques *Visión Artificial, Cinemática Inversa y Comunicación* fuera el correcto. Además, estas pruebas permitieron conocer el comportamiento real de los movimientos del robot respecto los movimientos de la mano del usuario.

La primera prueba consistió en tomar un bloque de madera y dejarlo en otro lugar, todo controlado desde el *Sistema de Guiamiento Dinámico*. Primero, se realizó un acercamiento de la pinza del robot hacia el bloque de madera, aproximación indicada por el *Guante Señalizador*. Ver figuras 5.1 y 5.2.

Figura 5.1

Figura 5.2

Luego se procedió a tomar el bloque de madera cerrando la pinza para luego levantar al objeto de lugar de donde estaba. Ver figuras 5.3 y 5.4.

Figura 5.3

Figura 5.4

Finalmente, se situó el objeto manipulado encima de otro bloque de madera para luego abrir la pinza y retirar el brazo de esta última posición. Ver figuras 5.5 y 5.6.

Figura 5.5

Figura 5.6

El funcionamiento de cada bloque del sistema fue el esperado. Sin embargo, el tiempo de espera para realizar cada marca era demasiado prolongado, de alrededor de 2 segundos, por lo que se reajusto este tiempo a un valor de 1 segundo.

La segunda prueba es muy similar a la primera, pero con la diferencia de que la distancia desde donde se debe tomar y dejar el objeto es más lejana, obligando al robot trasladarse sobre su riel. Las siguientes figuras muestran los instantes en que se realizó el acercamiento de la pinza y toma del objeto.

Figura 5.7

Figura 5.8

En las figuras 5.9 y 5.10 se puede apreciar el momento en que se realiza el traslado del robot sobre su riel.

Figura 5.9

Figura 5.10

Finalmente, las figuras 5.11 y 5.12 muestran, al igual que en la prueba anterior, el momento en que el bloque de madera es situado en el lugar de destino.

Figura 5.11

Figura 5.12

De las pruebas realizadas, el comportamiento de todo el sistema, incluido el robot, estuvo dentro de lo esperado.

Cabe mencionar que en algunas oportunidades, mientras se repetía la realización de las pruebas, ocurrieron colisiones entre el robot y su entorno. Colisiones que se produjeron producto de una errada conducción por parte de usuario. Sin embargo, frente a este tipo de eventualidad, el sistema no fue capaz de reponerse en algunas oportunidades, forzando a que se debiera reiniciar manualmente la aplicación en el computador y el *Programa Residente* en el controlador del robot.

También se observó una pequeña no-linealidad entre los movimientos realizados por la mano del usuario y las posiciones adquiridas por la pinza del robot, que puede

deberse a alguna variación no considerada de los parámetros de las cámaras o a la poca precisión en las mediciones del los huesos del robot. Sin embargo, no afecta mayoritariamente en el guiamiento del robot.

COMENTARIOS Y CONCLUSIONES

- Gracias a la Visión Artificial fue posible, mediante el procesamiento y análisis de imágenes, obtener datos del mundo real para la acción y toma de decisiones en el sistema.

- La representación gráfica permite visualizar que posición adoptará el robot cuando se realice la marca de un punto, lo que reduce la posibilidad de que el robot colisione con su entorno.

- El sistema ofrece la libertad al usuario de poder controlar el robot según los movimientos naturales de su mano.

- Aunque el sistema es totalmente operable, aun puede ser mejorado en el sentido de afinar o depurar la programación para que pueda responder de mejor manera a situaciones imprevistas como colisiones.

- El rendimiento podría mejorarse al utilizar cámaras industriales en vez de cámaras Web, y utilizar más computadores para aumentar la capacidad de cómputo en paralelo. Sin embargo, esto implica un aumento en el costo del sistema.

- Las marcas del *Guante Señalizador* y el sistema de visión pueden optimizarse para permitir al sistema detectar la rotación de la muñeca del usuario.

- Industrialmente, el sistema podría ampliarse para ser utilizado en la programación remota de brazos robot tipo Eshed Scorbot ER-IX, en la manipulación no programada de objetos específicos, o en el proceso de soldadura de estructuras por ejemplo.

BIBLIOGRAFÍA

[1] Gonzáles/Woods, “Tratamiento digital de imágenes”, Editorial Addison-Wesley, Año 1996, Pág. 245.

[2] Universidad Nacional de Quilmas, “Operaciones Morfológicas en Imágenes Binarias”, http://www.gpi.tsc.uvigo.es/libro/procesim.htm.

[3] Wikipedia, http://es.wikipedia.org/wiki/Apertura_angular.

[4] Luca Iocch, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Italy, “Stereo Vision: Triangulation”, http://www.dis.uniroma1.it/~iocchi/stereo/triang.html.

BIBLIOGRAFÍA

[1] Gonzáles/Woods, “Tratamiento digital de imágenes”, Editorial Addison-Wesley, Año 1996, Pág. 245.

[2] Universidad Nacional de Quilmas, “Operaciones Morfológicas en Imágenes Binarias”, http://www.gpi.tsc.uvigo.es/libro/procesim.htm.

[3] Wikipedia, http://es.wikipedia.org/wiki/Apertura_angular.

[4] Luca Iocch, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Italy, “Stereo Vision: Triangulation”, http://www.dis.uniroma1.it/~iocchi/stereo/triang.html.

ANEXO A: *Especificaciones Técnicas del Robot Scorbot ER-IX*

Estructura del Brazo		Vertical Articulado		
Numero de Ejes		5 más la pinza		
Eje	Movimiento	Rango	Velocidad Efectiva	Velocidad Máxima
Eje 1:	Rotación de la Base	300º	110º/seg.	165º/seg.
Eje 2:	Rotación del Hombro	175º	85º/seg.	127º/seg.
Eje 3:	Rotación del Codo	238º	100º/seg.	150º/seg.
Eje 4:	Inclinación de la Muñeca	200º	150º/seg.	225º/seg.
Eje 5:	Rotación de la Muñeca	360º	220º/seg.	330º/seg.
Radio máximo de operación		691 mm. sin pinza. 846 mm. con pinza.		
Elemento Terminal:		Pinza Neumática. Pinza Eléctrica servo controlada. 75 mm. apertura.		
Referencia:		Posición fijada en los ejes.		
Realimentación:		Encoders ópticos incrementales.		
Actuadotes:		Servo Motores de C.C.		
Transmisión:		Engranajes armónicos y correas dentadas.		
Carga Máxima:		2 Kg. incluyendo la pinza.		
Repetibilidad:		± 0,09 mm.		
Peso		38 Kg.		
Rango de Temperatura:		2º - 40º		

ANEXO B: ***Especificaciones técnicas de cámaras Web Creative NXUltra***

CREATIVE WEBCAM NX Ultra

The enthusiasts choice!

Check Product Availability

Introduction	Features	Specifications	Software
Requirements	Flash Demo	Gallery	Awards
Email Page	Print Page		

SPECIFICATIONS

- 640x480 CCD Sensor for still image capture up to 1280x960*
- Snapshot button and Manual Focus Lens
- Supports video capture at up to 640x480
- Wide field of view (78°) compared to other webcams

** software enhanced*

* Some of the products or bundled products featured here may not be available in your region or country. Please check with your local region site for product specifications and availability.