MINISTERIO DE EDUCACIÓN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE POST-GRADUACIÓN EN INGENIERÍA MECÁNICA

SISTEMA DE VISIÓN COMPUTACIONAL ESTEREOSCÓPICO APLICADO A UN ROBOT CILÍNDRICO ACCIONADO NEUMÁTICAMENTE

por

Daniela Ramírez Montecinos

Disertación para obtención del título de

Maestra en Ingeniería

Porto Alegre, Março de 2017

SISTEMA DE VISIÓN COMPUTACIONAL ESTEREOSCÓPICO APLICADO A UN ROBOT CILÍNDRICO ACCIONADO NEUMÁTICAMENTE

Por

Daniela Ramírez Montecinos

Ingeniera Mecatrónica

Disertación sometida al cuerpo docente del Programa de Post-graduación en Ingeniería Mecánica, PROMEC, de la Escuela de Ingeniería de la Universidade Federal do Rio Grande do Sul, como parte de los requisitos para la obtención del título de:

Maestra en Ingeniería

Área de concentración: Procesos de Fabricación

Orientador: Prof. Dr. Flávio José Lorini

Aprobada por:

Prof. Dr. Rafael Comparsi Laranja…….....PROMEC/UFRGS

Prof. Dr. Alcy Rodolfo dos Santos Carrara.....DEMEC/UFRGS

Prof. Dr. Fabiano Disconzi Wildner.....DEMEC/UFRGS

Prof. Dr. Jackson Manfredini Vassoler

Coordinador de PROMEC

Porto Alegre, 31 de Março de 2017

RESUMEN

En el área industrial los robots forman parte importante del recurso tecnológico disponible para tareas de manipulación en manufactura, ensamble, manejo de residuos peligrosos y aplicaciones varias. Los sistemas de visión computacional se han ingresado al mercado como soluciones a problemas que otros tipos de sensores y métodos no han podido solucionar. El presente trabajo analiza un sistema de visión estereoscópico aplicado a un robot. Este arreglo permite la medición de coordenadas del centro de un objeto en las tres dimensiones, de modo que, le da al robot la posibilidad de trabajar en el espacio y no solo en un plano. El sistema estereoscópico consiste en el uso de dos o más cámaras alineadas en alguno de sus ejes, mediante las cuales, es posible calcular la profundidad a la que se encuentran los objetos. En el presente, se mide la posición de un objeto haciendo una combinación entre el reconocimiento 2D y la medición de las coordenadas x y y de su centro calculadas usando momentos. En el sistema estereoscópico, se añade la medición de la última coordenada z mediante el cálculo de la disparidad encontrada entre las imágenes de las cámaras inalámbricas izquierda y derecha, que convierte al sistema en un visor 3D de la realidad, emulando los ojos humanos capaces de distinguir profundidades con cierta precisión. El sistema de visión computacional propuesto es integrado a un robot neumático de 5 grados de libertad el cual puede ser programado desde la metodología GRAFCET mediante software de uso comercial. Las cámaras del sistema de visión están montadas en el plano lateral del robot de modo tal, que es posible visualizar las piezas que quedan dentro de su volumen de trabajo. En la implementación, se desarrolla un algoritmo de reconocimiento y medición de posición, haciendo uso de software libre en lenguaje C++. De modo que, en la integración con el robot, el sistema pueda ser lo más abierto posible. La validación del trabajo se logra tomando muestras de los objetos a ser manipulados y generando trayectorias para el robot, a fin de visualizar si la pieza pudo ser captada por su garra neumática o no. Los resultados muestran que es posible lograr la manipulación de piezas en un ambiente visualmente cargado y con una precisión aceptable. Sin embargo, se observa que la precisión no permite que el sistema pueda ser usado en aplicaciones donde se requiere precisión al nivel de los procesos de ensamblado de piezas pequeñas o de soldadura.

Palabras clave: Visión computacional estéreo; Reconocimiento de objetos; Robot neumático; Generación de trayectorias.

ABSTRACT

In the industrial area, robots are an important part of the technological resources available to perform manipulation tasks in manufacturing, assembly, the transportation of dangerous waste, and a variety of applications. Specialized systems of computer vision have entered the market to solve problems that other technologies have been unable to address. This document analyzes a stereo vision system that is used to provide the center of mass of an object in three dimensions. This kind of application is mounted using two or more cameras that are aligned along the same axis and give the possibility to measure the depth of a point in the space. The stereoscopic system described, measures the position of an object using a combination between the 2D recognition, which implies the calculus of the coordinates of the center of mass x and y using moments, and the disparity that is found comparing two images: one of the right and one of the left. This converts the system into a 3D reality viewfinder, emulating the human eyes, which are capable of distinguishing depth with good precision.The proposed stereo vision system is integrated into a 5 degree of freedom pneumatic robot, which can be programmed using the GRAFCET method by means of commercial software.The cameras are mounted in the lateral plane of the robot to ensure that all the pieces in the robot's work area can be observed.For the implementation, an algorithm is developed for recognition and position measurement using open sources in C++. This ensures that the system can remain as open as possible once it is integrated with the robot. The validation of the work is accomplished by taking samples of the objects to be manipulated and generating robot's trajectories to see if the object can be manipulated by its end effector or not. The results show that is possible to manipulate pieces in a visually crowded space with acceptable precision. However, the precision reached does not allow the robot to perform tasks that require higher accuracy as the one is needed in manufacturing assembly process of little pieces or in welding applications.

Key words: Computer stereo vision; Object recognition; Pneumatic robot; Trajectory planning.

INDICE

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE SIGLAS Y ABREVIATURAS

API	*Application programming interface*
BLOB	*Binary large object*
CAN	*Controller Area Network*, protocolo de comunicación
C++	Lenguaje de programación de propósito general
GDL	Grado de libertad
GRAFCET	*Graphe Fonctionnel de Commande des Étapes et Transitions*, Gráfico Funcional de Etapa-Transción
DDE	*Dynamic Data Exchange*
IFR	*International Federation of Robotics*
IP	*Internet Protocol Address*, dirección de nodo Ethernet
LAMECC	Laboratorio de Mecatrónica y Control
OPC	*OLE for Process Control*, estándares para comunicación industrial
OpenCV	*Open Source Computer Vision*
RGB	Modelo de colores primarios aditivos rojo, verde y azul
RMS	Media cuadrática del error
RP5GDL	Robot neumático de 5 grados de libertad
SAD	Suma de diferencias absolutas
SGBM	*Semi Global Block Matching*, método de comparación de valores pixels
SPI	*Serial Peripheral Interface*, estándar de comunicación
UFRGS	Universidade Federal do Rio Grande do Sul

LISTA DE SÍMBOLOS

$A[m]$	Altura de montaje de los lentes de la cámara
$c\ [pixeles]$	Sistema de coordenadas de la cámara
c_x	Punto de interés en x (en la imagen)
c_y	Punto de interés en y (en la imagen)
D	Matriz de parámetros de distorsión
d	Disparidad
$DifS$	Diferencia sistemática entre el valor deseado y el valor real de posición
$d(x,y)$	Distancia mínima desde el pixel de coordenadas (x,y) a una línea
E	Matriz esencial
e	Punto epipolar
F	Matriz fundamental
f	Distancia focal
f_x	Distancia focal en x
f_y	Distancia focal en y
$\|G\|$	Magnitud del gradiente
$g_{x,y}$	Gradientes en las direcciones *x*, *y*
H	Matriz Homografía
$Hc\ (x,y)$	Función de convolución que se le aplicará al pixel en la posición (x,y)
$I_3(A,B)$	Función de comparación de momentos
Kg	Kernel para la detección del gradiente. Método de detección de bordes
k_1 , k_2 , k_3	Parámetros intrínsecos de distorsión radial
M	Matriz propia de la cámara (parámetros intrínsecos)
$m_{p,q}$	Momento de una imagen
(O_x, O_y)	Coordenadas del punto central de un sistema de coordenadas
$P(X,Y,Z)$	Punto de observación en el mundo
$p(x,y,z)$	Punto de observación en la imagen
p_1, p_2	Parámetros intrínsecos de distorsión tangencial
Q	Matriz de reproyección
$\tilde{Q}$	Coordenadas de un punto del objeto visto en el mundo
$\tilde{q}$	Coordenadas de un punto del objeto en la imagen (en pixeles)
$\tilde{Q}'$	Coordenadas de un punto del objeto visto en el mundo sólo en 2 dimensiones

R	Matriz de rotación
R_{rect}	Matriz de rectificación
r_l	Matriz de rotación de la izquierdaB
r_r	Matriz de rotación de la derecha
s	Factor de escala para una homografía
T, t	Matriz de traslación
T_b	Distancia entre cámaras, línea de base o traslación
W	Matriz que combina la traslación y rotación de un punto en una imagen
$w[m]$	Sistema de coordenadas del mundo
(x, y, z)	Coordenadas de un pixel horizontal, vertical y perpendicular al plano de la imagen
x_l	Coordenada x en la imagen de la cámara izquierda
x_r	Coordenada x en la imagen de la cámara derecha
x_v , y_v	Coordenadas corregidas tomando en cuenta las distorsiones radiales
$\bar{x}, \bar{y}$	Coordenadas del centro de masa de un objeto
$Z[m]$	Distancia desde la cámara al objeto
$\psi,\ \varphi\ , \theta$	Ángulos de rotación de una imagen en los 3 ejes
π_l, π_r	Planos de proyección izquierdo y derecho
$\emptyset$	Dirección del gradiente
2θ	Doble del ángulo de orientación de un objeto
$\mu_{p,q}$	Momento central de una imagen
τ	Momento de segundo orden
$\sigma(x, y)$	Función de contorno de un objeto

1. INTRODUCCIÓN

La robótica, como rama de la ingeniería, se encuentra actualmente en constante crecimiento y desarrollo. Las mejoras que se fomentan día a día se concentran principalmente en el incremento de la productividad y seguridad en una industria en la que los procesos manuales y tradicionales se hacen cada vez menos eficientes. Específicamente en el área de automatización, los robots inteligentes promueven plantas industriales en la mejora productiva, el ahorro de energía y además eleva los niveles de eficiencia por máquina o grupo de máquinas. En manufactura, los robots están siendo utilizados para generar productos con altos estándares de calidad [Pérez, Rodríguez et al., 2016] y estaciones de trabajo altamente capacitadas para lograr repetitividad en los ensambles.

En 2014, según el IFR (International Federation of Robotics), la venta de robots habría incrementado en 29% respecto del año 2013 con una venta total de más de 229 mil unidades y con un empuje de comercialización constante desde hace varios años.

Desde 2010, la demanda industrial de robots se ha acelerado considerablemente ya que la tendencia en la mejora continua de los métodos de control de los robots cada día está en continuo avance [IFR, 2016]. Más aún, la comisión europea como uno de las regiones líderes en robótica industrial, tenía una expectativa de inversión de 32 billones de dólares para el año 2016.

En el campo de los sistemas de visión, la mejora en los métodos técnicos ha permitido que puedan ser ampliamente usados en la industria, principalmente para inspección de productos y procesos de control de calidad. En el ambiente industrial, la visión implementada como respuesta sensorial del robot, también ha encontrado aplicaciones en mejora de la seguridad, transporte, organización de materias primas y acomodo de piezas, trabajando de forma autónoma o en combinación con humanos como colaboradores identificando objetivos.

Dependiendo del objetivo, la visión de robots está orientada a la detección de la escena o el objeto a movilizar [Pérez, Rodríguez et al., 2016]. En el presente trabajo se hace uso de visión estereoscópica para identificación de piezas y detección de las coordenadas de su posición. Se pretende agregar este sistema a el de un robot desarrollado en el Laboratorio de Mecatrónica y Control (LAMECC) de la Universidade Federal do Rio Grande do Sul (UFRGS), en la ciudad de Porto Alegre, Brasil. El mencionado es un robot de conFiguración cilíndrica con 5 grados de libertad accionado neumáticamente [Medina, 2015].

La visión estereoscópica es un área en el campo de la visión computacional que apunta a un importante problema: la reconstrucción tridimensional de las coordenadas de los puntos para la estimación de la profundidad o coordenada “z”. Un sistema como este consiste en dos o más cámaras posicionadas horizontalmente tomando imágenes simultáneamente que después serán procesadas [Kanellakis, Kyritsis et al., 2015].

El sistema de visión presentado con dos cámaras, integradas al robot mencionado, ayudará al mismo a la identificación de la posición de piezas. De este modo, y con cierta autonomía, éste podrá tomar decisiones respecto a su trayectoria para poder captar los objetos según las imágenes provistas.

1.1. Descripción del Problema y alternativa de solución

Anteriormente se mencionó el robot desarrollado enteramente en laboratorios de LAMECC/UFRGS, un robot neumático con 5 grados de libertad. Este robot fue pensado para la realización de tareas de manipulación de piezas para la producción industrial [Medina, 2015]. Dentro de este alcance es que se desarrolla el presente trabajo, tomando en cuenta que las piezas pueden variar de formas y colores, y que el problema se centra en las coordenadas de dichas piezas como información de entrada para la generación automática de trayectorias del robot. Estos datos, después de ser normalizados y acondicionados para su uso, servirán de información de entrada al sistema de planeación de trayectorias según el esquema desarrollado en Missiaggia, 2014.

Anteriormente un trabajo parecido fue presentado en el formato de coordenadas 2D, es decir, coordenadas x y y que una cámara puede captar desde una posición privilegiada sin obstrucciones y dentro del volumen de trabajo del robot [Medina, 2015]. El caso fue finalmente presentado con éxito, sin embargo, la mejora presentada permite al robot tener conocimiento de una tercera coordenada z.

La reconstrucción 3D planteada por la visión estereoscópica, es un avance al sistema de visión del robot. Esto puede derivar a que en un futuro sus capacidades se expandan y le permitan finalmente ser integrado en un ambiente productivo real.

El punto más importante, sin embargo, del trabajo en sí, es el de integrar los sistemas de planeación de trayectorias y el sistema de visión, logrando en el avance que cada punto de la trayectoria esté dentro del espacio de trabajo del robot y probando la eficacia de la manipulación de piezas.

1.2 Objetivo General

Desarrollar un sistema de visión estereoscópico computacional que permita obtener la posición y orientación de piezas que serán manipuladas por un robot neumático de 5 grados de libertad.

Al mismo tiempo, integrar el sistema de visión con el sistema de generación de trayectorias tomando como base la información de las coordenadas generadas en los tres ejes. Finalmente, se realizarán pruebas de funcionamiento y verificación del sistema de visión acoplado al sistema de navegación del robot.

1.3 Objetivos Específicos

Serían los siguientes en orden de importancia:

1. Trabajar con visión computacional para determinar posición y orientación de objetos en una escena de trabajo (simulando una situación industrial)
2. Poder enviar información al robot para poder modificar sus trayectorias y lograr manipular piezas de forma eficiente
3. Definir el tipo de piezas que pueden ser incluidas en la manipulación y posibles incompatibilidades con el sistema

4. Integrar el sistema de visión con el control del robot, lograr una interface confiable que pueda ser utilizada de manera simple

1.4 Organización del trabajo

El presente trabajo está organizado en los siguientes capítulos:

- Capítulo 1: Una introducción que referencia los motivos que llevaron a escoger el tema del trabajo propuesto, así como la descripción concreta del problema y los objetivos.
- Capítulo 2: Se presenta la fundamentación teórica y el estado del arte del tema abordado.
- Capítulo 3: Se presenta la metodología de reconocimiento de imágenes usada, el proceso de captación de las coordenadas “x”,” y” y “z” y la forma de determinación de trayectorias del robot.
- Capítulo 4: Se presenta el procedimiento de implementación mediante el cual se modifica la trayectoria del robot para lograr la manipulación de piezas
- Capítulo 5: Se muestran los resultados de los experimentos
- Capítulo 6: Se presentan las conclusiones y consideraciones finales respecto de los resultados obtenidos
- Capítulo 7: Se especifican las referencias

2. FUNDAMENTACIÓN TEÓRICA

2.1. Introducción

Actualmente, los manipuladores robóticos están usando nuevas y mejoradas capacidades de sensado añadidas a sus estructuras ya que la complejidad de las tareas que tienen que realizar aumenta. Hoy en día éstos tienen que ser capaces de actuar de forma inteligente aportando soluciones óptimas.

En el área industrial, la integración de los sistemas robóticos al proceso de producción es un hecho que requiere de sistemas eficientes, prácticos y que puedan desarrollar tareas con una determinada repetitividad mínima. En el caso de las operaciones de ensamblado modernas, el requerimiento de robots con avanzados controles de manipulación se hace inminente, siendo necesaria la adopción de técnicas de visión computacional para reconocimiento de objetos y para la identificación de sus coordenadas en el espacio [Pérez, Rodríguez et al., 2016].

La manipulación guiada por un sistema de visión, desde el punto de vista físico, requiere del uso de al menos una cámara o un par de cámaras actuando en un sistema estereoscópico. Un sistema estereoscópico le brinda la posibilidad al robot de tener unos "ojos" semejantes a los de los seres humanos capaces de emular profundidades y poder distinguir, por ejemplo, si un objeto está más cercano que otro o no [Bradski y Kaehler, 2008], [Cyganek y Siebert, 2009] .

2.2 Imágenes digitales, matrices y modos de representación

Las imágenes digitales son muestras gráficas de objetos que son capturadas a partir de fotosensores que miden la cantidad de luz emitida por estos objetos. Una imagen digital se compone de pixeles que son valores generalmente decimales que almacenan el valor de luz que fue captaba en el momento de la toma. El valor mínimo de un pixel es 0 y representa el negro. Cuando el valor de pixel se incrementa, la intensidad de ese pixel también se incrementa [Cyganek y Siebert, 2009]. La cantidad de bits asignados para cada pixel varía según la necesidad, la más común es la de un byte. Estos 8 bits que componen un pixel, por ejemplo, pueden variar el valor del byte hasta los 255 en decimal, lo que sería al ojo humano el blanco.

En el campo de la digitalización, se hace uso del modelo de color aditivo llamado RGB por sus siglas en inglés para *Red, Green y Blue*. Los colores se forman a partir de la correcta combinación de valores decimales del 0 al 255 de los colores primarios aditivos rojo, verde y azul[1] según el modelo RGB. Es por esto que, en el caso más básico, existen al menos tres canales en cada imagen digital a color[Zisserman y Hartley, 2004].

Los colores, en la teoría de procesamiento de imágenes, se llevan al plano matemático con el uso de planos o canales. Nuevamente, cada pixel de la imagen estará definido en función de una cantidad de bits asignados. En el caso del rojo, por ejemplo, un 255 en el valor del pixel definido como entero positivo de 8 bits, sería el color rojo sin tonalidad. En la

[1] En la teoría conservadora, originalmente se consideraban el rojo, amarillo y azul como los colores primarios. Con el avance de la digitalización este modelo fue conservado principalmente en las aplicaciones de artes manuales.

Figura 2.1 [Cyganek y Siebert, 2009], se puede observar un modelo simple de una imagen a color de (i x k) pixeles:

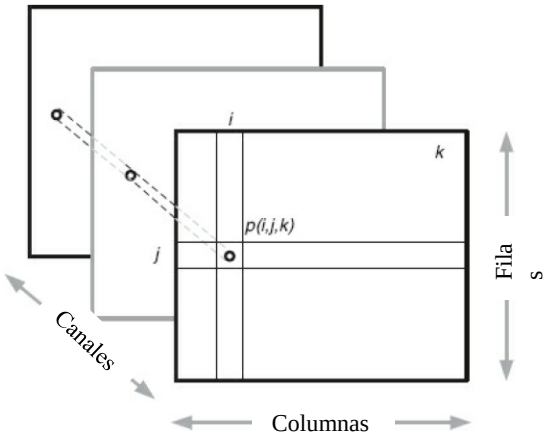

Figura 2.1 - Representación de las matrices (j canales) de una imagen a color.

Las propiedades de la imagen representada por las matrices en la Figura 2.1 serían:

- Profundidad de la imagen: 24 bits o 3x8 bits
- Tres canales: imagen a color $j = 3$
- Resolución: i x k pixeles

2.3 Transformación de imágenes: convolución, suavizado, reducción de ruidos, transformaciones morfológicas

Las imágenes como matrices pueden ser tratadas matemáticamente para lograr determinados resultados visuales. La transformación de imágenes son métodos que se usan para cambiar una imagen en una representación alternativa de datos. El caso más común podría ser, por ejemplo, aplicar la transformada de Fourier con la cual la imagen representa componentes espectrales en lugar de espaciales. El interés de la presente es, sin embargo, las transformaciones espaciales como se verá a continuación.

Las convoluciones, desde su forma aplicada en procesamiento de imágenes, es la base de muchas de las transformaciones. Una convolución se determina por el tipo de "kernel" usado. El mencionado kernel, es un término utilizado esencialmente para describir una matriz de tamaño fijo cuyos componentes se calcularon/escogieron para lograr un efecto en la imagen. Así cada pixel será transformado aplicando la convolución de las matrices usando un determinado kernel [Sobel y Feldman, 1968]. En la Figura 2.2 [Sobel y Feldman, 1968], se puede ver un kernel de convolución de 3x3. En general, las dimensiones de los kernel son impares y de igual número de filas y columnas.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figura 2.2 - Ejemplo de kernel de convolución, en el caso, el operador para detección de bordes en el eje x.

El proceso de convolución se usa para extraer determinadas porciones relevantes de una imagen, esto se puede ver en el apartado donde se hace énfasis en un algoritmo de

detección de bordes. Una convolución o aplicación de filtro, es una operación matemática que se lleva a cabo usando un kernel y una imagen digital como se había mencionado. Matemáticamente se puede expresar el proceso como: Ecuación 2.1 [Cyganek y Siebert, 2009].

$$H(x,y) = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} G(i,j)I(x+i-a_i, y+j-a_j) \quad (2.1)$$

Donde:

$H(x,y)$: Es el cálculo del valor de convolución que se le aplicará al pixel en la posición (x,y)

K: Es el valor de filas y columnas del kernel que se esté utilizando

$(a_{i,}a_j)$: Es la coordenada del valor central del kernel, lo que se conoce comúnmente como el punto "ancla". Por ejemplo, en la Figura 2.3, sería el valor de $(1,1) = 0$

(x,y): Es el valor de las coordenadas del pixel que está siendo utilizado para el cálculo.

Nótese que el valor resultante de la convolución se aplica a cada uno de los pixeles de la imagen, superponiendo el pixel a modificarse con el punto ancla del kernel. Esto se entiende mejor observando la Figura 2.3 [Powell, 2016], donde se muestra la aplicación de un filtro suavizante haciendo uso de un kernel y una operación de convolución.

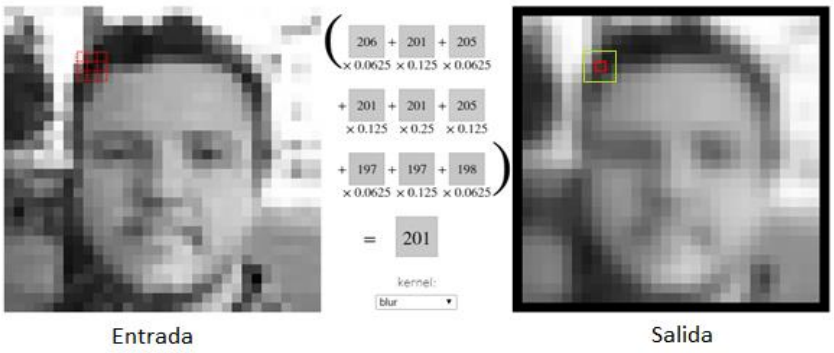

Figura 2.3 - Ejemplo de convolución.

Se puede observar a la izquierda como entrada la matriz 3x3 de la imagen, este "cuadro" es el que el algoritmo utiliza para el cálculo. El resultado igual a 201 será el valor de la pigmentación en gama de grises que se observa resaltado en rojo en la imagen de la derecha.

El efecto que resulta de la aplicación del kernel de blurring o suavizado, es el que se puede observar en la Figura 2.3. Este kernel se conoce como filtro de Gauss y se muestra en la Figura 2.4:

$$\begin{pmatrix} 0{,}0625 & 0{,}125 & 0{,}0625 \\ 0{,}125 & 0{,}25 & 0{,}125 \\ 0{,}0625 & 0{,}125 & 0{,}0625 \end{pmatrix}$$

Figura 2.4 - Kernel para un filtro de Gauss.

Se puede observar que en el punto ancla se tiene el valor más alto, asemejando el punto más alto de la campana de Gauss. La teoría detrás de los kernel Gaussianos es muy extensa, sin embargo, para el propósito de este trabajo, se hará foco en el resultado del uso de este filtro.

El filtro de suavizado o *blurring* que se logra con un kernel Gaussiano, permite desenfatizar las diferencias de los pixeles adyacentes al que es tratado. Modificando el valor de la pigmentación de cada pixel, también logra eliminar el ruido de una imagen. Estas tareas son útiles para poder encontrar en los siguientes pasos, los bordes para reconocimiento de patrones [Powell, 2016].

Otro tipo de filtro, no de convolución, si no se podría decir de "selección" son las transformaciones morfológicas. En una imagen el rango de intensidad, por ejemplo, la diferencia entre la mayor y la menor intensidad de los valores de pixeles en una imagen, da una medida de su contraste[Cyganek y Siebert, 2009]. Existen técnicas estándar como la ecualización por histogramas para mejorar el contraste de una imagen degradada, seleccionando el pixel de mayor contraste y sustituyéndolo en el espacio de los pixeles vecinos. En algunos casos este tipo de filtros funcionan, sin embargo, es probable que se pierdan detalles de la imagen.

2.4 Modelo de cámara "Pinhole"

La captación de una imagen comienza con la detección de la luz del entorno. Esa luz, como rayos, emana desde alguna fuente y viaja a través del espacio hasta que choca con algún objeto. Cuando sucede la colisión, mucha de esa luz es absorbida, mientras que la restante que no se absorbe, se la puede ver como el color de dicho objeto. La luz reflejada es colectada por nuestra retina a través de los ojos o por medio de sensores de la cámara. En visión computacional, es de particular importancia el arreglo geométrico para esta captación [Bradski y Kaehler, 2008].

Un modelo simple pero útil de cómo sucede esto, es el modelo de cámara *PinHole*. El *Pinhole* es una pared imaginaria con un agujero pequeño en el centro que bloquea todos los rayos de luz excepto aquellos que pasan a través de su apertura. El modelo de cámara *Pinhole* permite analizar una geometría básica por donde es posible explorar la geometría de los rayos incidentes. Cabe resaltar que un orificio real de estas características, no es una buena manera de captar imágenes porque no permite pasar la suficiente cantidad de luz en una exposición rápida [Bradski y Kaehler, 2008]. Es por esto que los ojos de un ser vivo y las cámaras usan lentes para captar mejor la luz disponible en un único punto. Sin embargo, se usa este modelo simple porque, tomando en cuenta los lentes, la geometría se transformaría en un modelo mucho más complejo ya que también se añade las distorsiones propias de los lentes.

El modo en el que se manejan estas distorsiones se puede ver más adelante en el apartado 2.5 Calibración de cámaras.

En el modelo básico, la luz entra desde la escena u objeto distante por el orificio creado, el cual, permite ingresar un único rayo de luz desde un punto particular. En una cámara *Pinhole* física, este punto es proyectado en la superficie donde estará la imagen. Como resultado, la imagen en este *plano de imagen* o llamado también, plano de proyección, está

siempre en el foco y el tamaño de la imagen relativa a la distancia del objeto es dada por un solo parámetro de la cámara que es la distancia focal.

En este modelo ideal, la distancia desde la apertura al plano de imagen es precisamente la distancia focal. Esto se muestra en la Figura 2.5 [Bradski y Kaehler, 2008], donde f es la distancia focal de la cámara, Z es la distancia desde la cámara al objeto, X es la longitud real del objeto, x es la longitud del objeto sobre el plano de la imagen, Y es el ancho real del objeto y y es el ancho del objeto sobre el plano de la imagen. Por similitud de triángulos se puede evidenciar que $-\frac{x}{f} = \frac{X}{Z}$ y , $-\frac{y}{f} = \frac{Y}{Z}$, ordenando se obtienen las ecuaciones (2.2) y (2.3):

$$-x = f\frac{X}{Z} \tag{2.2}$$

$$-y = f\frac{Y}{Z} \tag{2.3}$$

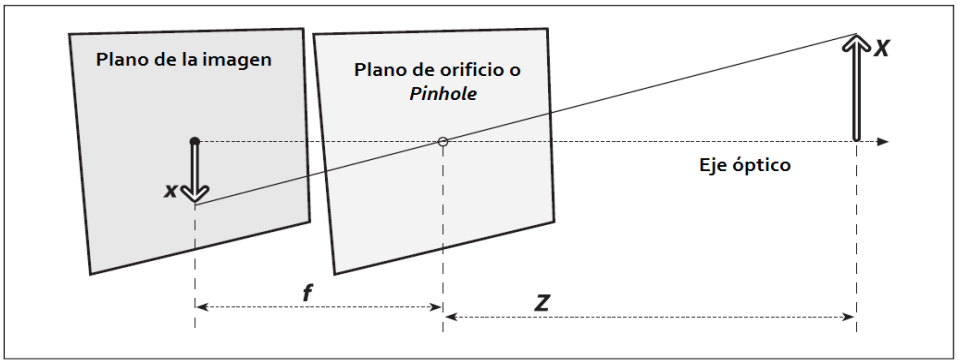

Figura 2.5 - Modelo de cámara Pinhole.

El orificio en el plano permite pasar solamente aquellos rayos de luz que intersectan un punto particular en el espacio; estos rayos entonces forman una imagen "proyectándola" en el segundo plano [Bradski y Kaehler, 2008].

De donde también se deriva la siguiente ecuación fundamental:

$$z = f \tag{2.4}$$

A fin de entender cómo los puntos en el mundo real están relacionados matemáticamente a los puntos en el plano de la imagen proyectada en 2 dimensiones, se tiene que tomar en cuenta los siguientes dos sistemas de coordinadas de interés Figura 2.6 [Cyganek y Siebert, 2009]:

2 El sistema de coordenadas externo (denotado por W), el cual es independiente de los parámetros de la cámara
3 El sistema de coordenadas de la cámara (denotado por C)

Los dos sistemas de coordenadas están relacionados por la traslación expresada por la matriz T, y la rotación representada por la matriz R [Cyganek y Siebert, 2009].

En la Figura 2.6 se muestra la representación de un objeto proyectado en el plano II, el cual aparece cuadriculado asemejando los pixeles o bytes que guardan la información de la imagen (representando el modo por el cual una cámara digital capta las imágenes).

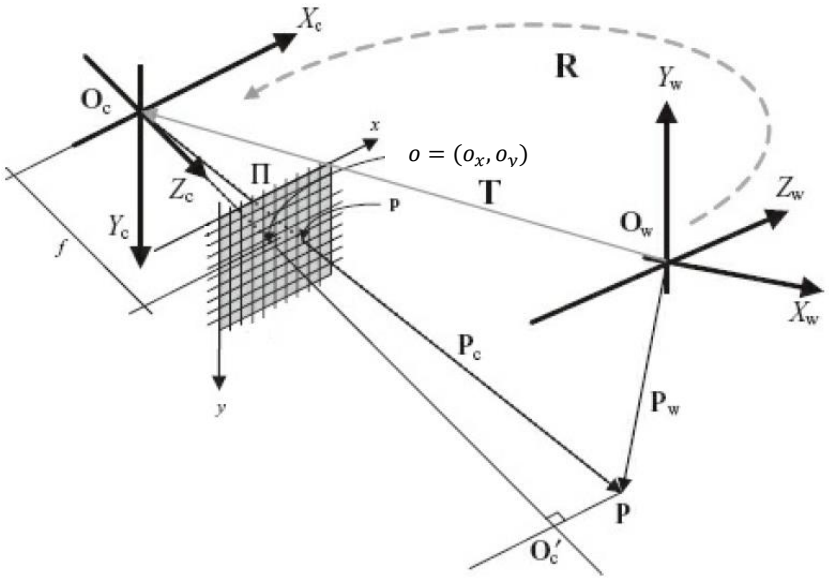

Figura 2.6 - Modelo Pinhole de la cámara con dos ejes de coordenadas externa W e interna C

La proyección del punto O_c en el plano II en la dirección de Z_c, determina el punto principal de las coordenadas locales (O_x, O_y). El eje principal es la línea entre los puntos O_c y O'_c. La distancia desde el plano de la imagen proyectada al punto principal es conocida como la distancia focal f (antes ya mencionada).

Se observa también que el punto p es una imagen del punto P cuyo centro de proyección es el punto O_c en el plano II. Las coordenadas de los puntos p y P en el sistema de la cámara se pueden denotar como sigue [Cyganek y Siebert, 2009]:

$$P = [X, Y, Z]^T \tag{2.5}$$

$$p = [x, y, z]^T \tag{2.6}$$

Por otro lado, y agregando detalles al modelo *Pinhole*, este puede ser definido mediante los siguientes parámetros:

a) Parámetros Intrínsecos
b) Parámetros Extrínsecos

2.4.1 Parámetros extrínsecos e intrínsecos del modelo de cámara Pinhole.

Parámetros extrínsecos

La descripción matemática de una escena dada depende del sistema de coordenadas elegido y está basado solamente en la posición del plano de la imagen, con lo que se determina el lugar exacto de la posición de la cámara. En el caso más sencillo, se suele elegir el sistema de coordenadas de la cámara como la referencia. Sin embargo, se puede complicar

todo un poco más si se tiene más de una cámara como participante del sistema. Esto a raíz que se debe conocer la posición (relativa) de cada cámara que se incluya [Bradski y Kaehler, 2008].

El cambio desde el sistema de coordenadas C de la cámara hacia el sistema de coordenadas W del mundo real se puede calcular observando que existe una traslación T y una rotación R. El vector de traslación T, describe un cambio en la posición de los centros de coordenadas, mientras que el vector de rotación R describe los cambios correspondientes a las direcciones de los ejes de cada sistema. Este último cambio es descrito como ortogonal[2] y de 3x3 dimensiones.

Para un punto dado P, las coordenadas relativas a la cámara C y las coordenadas relativas al mundo real W están conectadas por la siguiente fórmula: Ecuación (2.7) [Bradski y Kaehler, 2008]

$$P_c = R(P_w - T) \tag{2.7}$$

Donde P_c expresa el punto P en el sistema de coordenadas de la cámara, P_w es el punto en el sistema de coordenadas externo, y R y T son las matrices de rotación y traslación respectivamente. En el caso de la rotación, ésta puede ser descrita en términos de la multiplicación de un vector de coordenadas y una matriz de dimensiones apropiadas. La rotación es equivalente a introducir una nueva descripción de posición de un punto en un nuevo sistema de coordenadas. Rotar el sistema de coordenadas un ángulo θ es rotar el punto deseado alrededor del origen de ese sistema de coordenadas por el mismo ángulo θ. La representación de la rotación en dos dimensiones como una multiplicación de matrices se puede ver en la Figura 2.7 [Bradski y Kaehler, 2008].

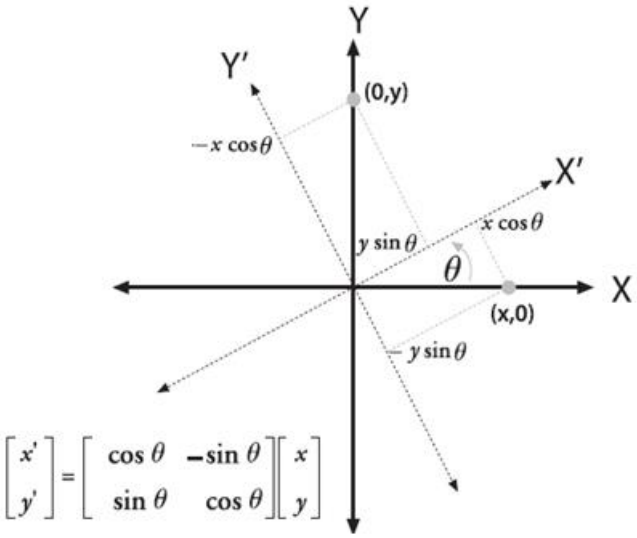

Figura 2.7 - Puntos de rotación θ grados.

[2] Lo que es $RR^T = 1$

En este caso respecto del eje z, es lo mismo que rotar los ejes de las coordenadas por θ. Por simple trigonometría se puede ver cómo la rotación cambia las coordenadas de un punto [Bradski y Kaehler, 2008].

La rotación en tres dimensiones puede ser descompuesta en una rotación de dos dimensiones alrededor del eje en el cual las mediciones en el eje pivote se mantienen constantes. Si se rota los ejes x, y y z en determinada secuencia según los ángulos ψ, φ y θ, el resultado es una rotación total que se puede llamar matriz R y que es dada por el producto de las matrices $R_x(\psi), R_y(\varphi)$ y $R_z(\theta)$, donde: Ecuaciones (2.8), (2.9) y (2.10), [Bradski y Kaehler, 2008].

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{2.8}$$

$$R_y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \tag{2.9}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Así finalmente, las matrices tanto de rotación como de traslación se pueden especificar como: Ecuaciones (2.11) y (2.12), [Bradski y Kaehler, 2008].

$$R = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}_{3x3} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}_{3x3} \tag{2.11}$$

$$T = O_W - O_C = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}_{3x1} \tag{2.12}$$

Como conclusión se dice que, los parámetros extrínsecos son aquellos parámetros geométricos que permiten cambiar el sistema de coordenadas de la cámara hacia el sistema de coordenadas externo o del mundo real y viceversa.

Parámetros intrínsecos

Los parámetros intrínsecos de la cámara relacionan el modelo geométrico de la misma con la distorsión que se añade por el uso de lentes. Estos parámetros pueden ser resumidos como [Cyganek y Siebert, 2009]:

1. Los parámetros de la transformación proyectiva[3] como tal: Para un modelo *Pinhole*, estos son dados por la distancia focal.
2. Los parámetros que mapean el sistema de coordenadas de la cámara en un sistema de coordenadas de la imagen: Asumiendo que el origen de la imagen constituye un punto $O = (O_x, O_y)$, por ejemplo el punto central, y dado que las dimensiones físicas de los píxeles (usualmente expresados en µm) en el plano de la cámara en las dos

[3] La transformación proyectiva es la operación u operaciones necesarias para convertir la Figura dada en otra, de manera que exista entre los elementos origen y transformados una relación biunívoca

direcciones son constantes y están dadas por h_x y h_y, una relación entre las coordenadas de la imagen x_c y y_c y las coordenadas de la cámara x y y puede ser establecida como sigue: Ecuaciones (2.13) y (2.14)

$$x = (x_c - O_x)h_x \tag{2.13}$$

$$y = (y_c - O_y)h_y \tag{2.14}$$

Donde el punto (x, y) le pertenece al sistema de coordenadas de la cámara "C", mientras que $(x_c,\ y_c)$ y (O_x, O_y) al sistema del plano local de la cámara. Es común asumir que $x_c \geq 0$ y $y_c \geq 0$. Por tanto, el punto de origen del plano de la cámara $(x_c,\ y_c) = (0,0)$ se transforma al punto $(-O_x h_x, -O_y h_y)$ del sistema "C". También es común asumir que $h_x = h_y = 1$, bajo esta presunción el punto del ejemplo es simplemente $(-O_x, -O_y)$ en las coordenadas en "C".

3. Las distorsiones geométricas correspondientes a los parámetros físicos de los elementos ópticos de la cámara: Las distorsiones que se encuentran en un sistema óptico real se deben principalmente de la no-linealidad de sus elementos, al igual que a la dependencia de los parámetros ópticos de la longitud de onda de los rayos de luz incidentes. En el primer caso se habla de la aberración esférica, la aberración comática, el astigmatismo, la curvatura del campo y las distorsiones[4]. El segundo caso está relacionado a la aberración cromática[5]. En la mayoría de las situaciones prácticas, se puede modelar estos fenómenos cuyos valores aumentan los puntos más distantes del centro de la imagen. Las distorsiones radiales pueden ser modeladas usando una corrección no lineal (*offset*) en las coordenadas reales de un punto en una imagen dada. Habrá que aclarar en este punto, que las distorsiones radiales son el resultado de la forma de los lentes, mientras que las distorsiones tangenciales son el resultado del proceso de ensamblado de la cámara como tal. Para las distorsiones radiales, este valor es 0 en el centro (óptico) de la cámara e incrementa mientras más cerca se esté a la periferia. En la práctica estas distorsiones son pequeñas y pueden ser caracterizadas por los primeros términos de la serie de Taylor alrededor de $r = 0$. En el caso de las cámaras más baratas, generalmente se usan los primeros dos términos k_1 y k_2. Para aproximaciones más exactas o para cámaras con muchas distorsiones se añade el término k_3: Ecuaciones (2.15) y (2.16) [Cyganek y Siebert, 2009]

$$x_v = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.15}$$

$$y_v = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.16}$$

Donde $r^2 = {x_v}^2 + {y_v}^2$, k_1 , k_2 y k_3 son los nuevos parámetros intrínsecos que modelan la influencia de las distorsiones radiales en el sistema óptico; x y y son las coordenadas ideales de un punto en una imagen dada (tomados como sin distorsiones). Finalmente x_v y y_v son las coordenadas corregidas tomando en cuenta las distorsiones radiales.

[4] Debido a algunos defectos de diseño o imperfecciones en los lentes u otros componentes de la cámara, se generan estas distorsiones que resultan en fuentes puntuales de luz fuera de eje.
[5] Diferente a la aberración cromática, este tiene que ver con la focalización de diferentes colores en el mismo punto

La segunda distorsión real más considerada es la tangencial. Esta distorsión nace a partir de los defectos de manufactura resultantes de los lentes que no están totalmente paralelos al plano de la imagen. Este tipo de distorsión puede ser mínimamente caracterizado por dos parámetros adicionales p_1 y p_2 como sigue: Ecuaciones (2.17) y (2.18)

$$x_v = x + \ [2p_1 y + p_2(r^2 + 2x^2)] \tag{2.17}$$

$$y_v = y + \ [2p_2 x + p_1(r^2 + 2y^2)] \tag{2.18}$$

En suma, se toman en cuenta cinco coeficientes de distorsiones. OpenCV[6] usa el procedimiento planteado por [Heikkila y Silven, 1997] para encontrar x_v y y_v como parámetros intrínsecos de la cámara. Finalmente, estos cinco coeficientes se usan en la mayoría de las rutinas de calibración.

2.5 Calibración de la cámara

2.5.1 Parámetros propios de la cámara y sus características

De la ecuación (2.7) se ve que la combinación de ecuaciones para P_c y las correcciones que describen los parámetros intrínsecos de la cámara irán a formar el sistema básico de ecuaciones que deberán ser resueltas para lograr la calibración de las cámaras, [Bradski y Kaehler, 2008].

Se vio que una rotación en tres dimensiones puede ser especificada con tres ángulos y que una traslación en tres dimensiones puede ser especificada con los tres parámetros (x, y, z). De allí, por simple observación, se tienen seis parámetros a especificar. Los parámetros intrínsecos de la cámara, por otro lado, tienen otros cuatro parámetros f_x, f_y, c_x y c_y (mismos que se discutirán más adelante pero que son respectivamente: distancia focal en x, distancia focal en y, punto de interés en x y punto de interés en y) lo que hace un total de diez parámetros que deberán ser resueltos para cada vista de una imagen. Nótese también que estos parámetros permanecerán fijos entre todas las vistas [Bradski y Kaehler, 2008].

Los parámetros que sí van a cambiar entre vistas son los otros seis pertenecientes a la traslación y la rotación. De ahí se concluye que, para poder resolver los otros cuatro parámetros intrínsecos de la cámara, se requieren por lo menos dos vistas por cada eje [Bradski y Kaehler, 2008].

Antes de entrar en el tema de calibración como tal, se presenta el *objeto* que se usa de herramienta para la calibración. Este objeto es una cuadrícula plana que se alterna entre cuadros negros y blancos que usualmente se denomina “tablero de ajedrez”.

2.5.2 Objeto “tablero de ajedrez”

En principio, cualquier objeto caracterizado como el mencionado “tablero de ajedrez” puede ser usado como objeto de calibración. Un tablero de ajedrez, desde la vista del plano superior, muestra una imagen donde es más fácil distinguir la posición de los pixeles de las

[6] OpenCV (Open Source Computer Vision) librerías de funciones de programación libres que principalmente se usan para desarrollos en visión computacional.

esquinas de los cuadros. El objetivo de usar este tipo de patrones para la calibración es el de localizar los pixeles mencionados de modo que se logra cuadricular la imagen [Brown, 1971].

En la práctica, además, se deben hacer uso de algoritmos alternos también encontrados en OpenCV como el dado para encontrar las esquinas a nivel de subpixel.

En la Figura 2.8, se puede ver un ejemplo de lectura y dibujo de esquinas usando un tablero de ajedrez como los ya mencionados. Las líneas oblicuas que unen los extremos superior e inferior describen el orden de lectura de cada punto.

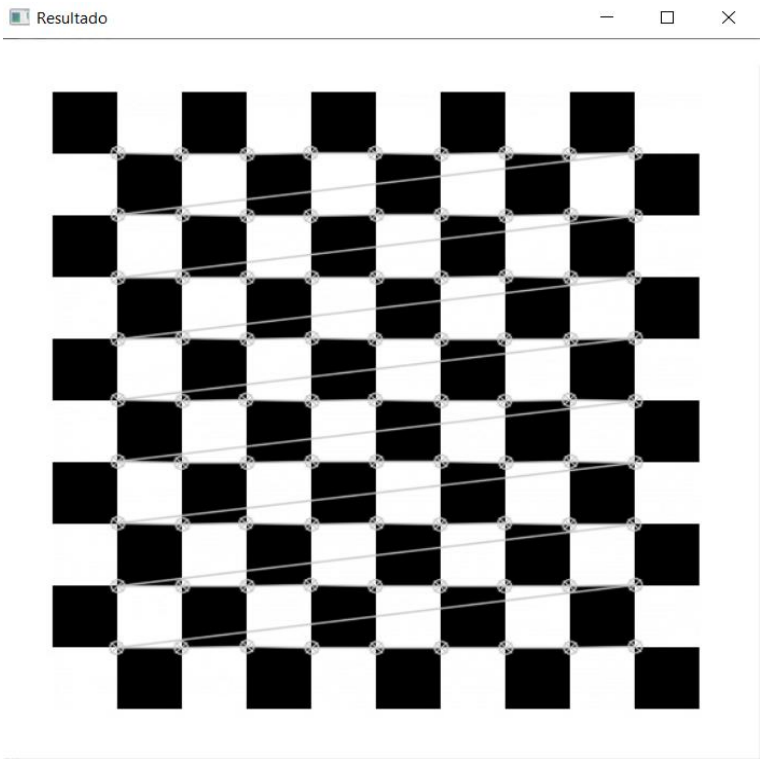

Figura 2.8 - Resultado de la búsqueda de esquinas usando las funciones básicas de cvDrawChessBoardCorners() y cvFindChessboardCorners(). Fuente propia

Sabiendo entonces, como antes se dijo, que los puntos en un plano se corresponden a otros que se toman desde su perspectiva cuando pasa a través de un orificio, se va a explorar un poco el tema de esta transformación que está contenida en la llamada homografía, matriz de 3x3.

2.5.3 Homografía

En visión computacional, se conoce a la homografía como un mapeo a modo de proyección de un plano a otro. Es posible expresar este mapeo en términos de una multiplicación de matrices si se usan coordenadas homogéneas para expresar el punto del objeto visto en Q y el punto q del plano de la cámara. Véase que el punto Q es mapeado en q. Se define: Ecuación (2.19) y (2.20) [Bradski y Kaehler, 2008]

$$\tilde{Q} = [X \quad Y \quad Z \quad 1]^T \tag{2.19}$$

$$\tilde{q} = [x \quad y \quad 1]^T \tag{2.20}$$

En la literatura se expresa la homografía como: Ecuación (2.21)

$$\tilde{q} = sH\tilde{Q} \tag{2.21}$$

Donde s es un factor de escala arbitrario que sugiere que la homografía está definida dentro de un cierto límite. Esta transformación se puede resolver mediante geometría, siendo la observación más importante que H tiene dos partes: la transformación física que esencialmente localiza el plano del objeto que se está viendo y la proyección que introduce los parámetros intrínsecos de la cámara. La Figura 2.9 ejemplifica lo expuesto: [Bradski y Kaehler, 2008]

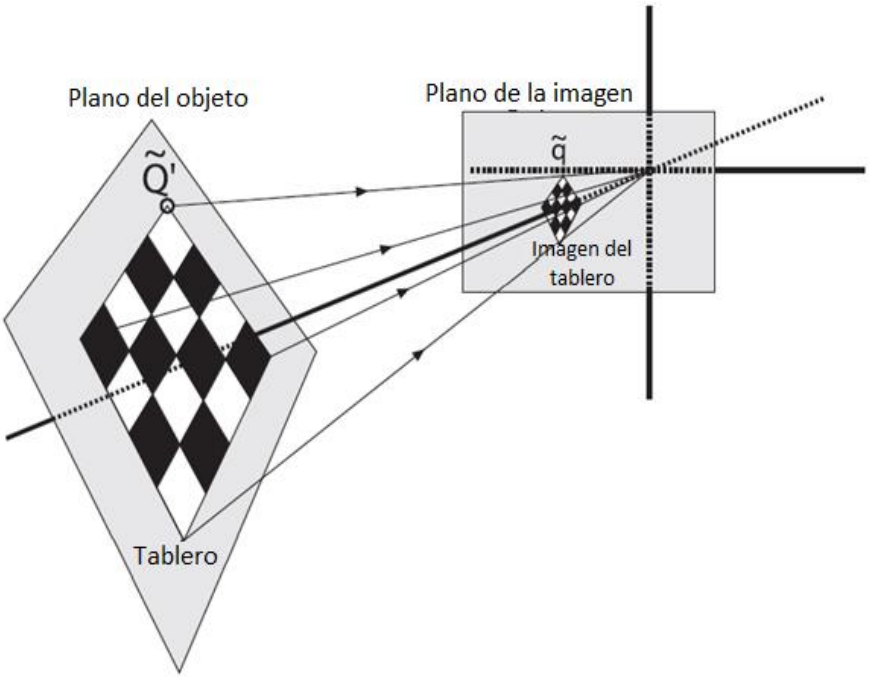

Figura 2.9 - Homografía, mapeo desde el plano de un objeto al plano de la imagen.

La transformación física es parte de la suma de los efectos de rotación R y traslación T, las cuales relacionan el plano que se observa con el plano de la imagen. Ya que se está trabajando en coordenadas homogéneas, se puede combinar estas dos matrices en una sola como sigue: Ecuación (2.22) [Bradski y Kaehler, 2008]

$$W = [R \quad t] \tag{2.22}$$

Antes se habían mencionado los parámetros propios de la cámara, ahora se los va a incluir en una matriz que se nombrará M:

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.23}$$

De esto y usando la ecuación (2.21) se puede concluir que: Ecuación (2.24)

$$\tilde{q} = sMW\tilde{Q} \tag{2.24}$$

Donde $H = MW$, describe la homografía. En la práctica se observa que las imágenes captadas por la cámara son estrictamente tomadas en 2 dimensiones. Lo que permite simplificar las ecuaciones como sigue[7]: [Bradski y Kaehler, 2008]

$$\tilde{q} = sMW\tilde{Q}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = sM[r_1 \quad r_2 \quad r_3 \quad \boldsymbol{t}\ ] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$= sM[r_1 \quad r_2 \quad \boldsymbol{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.25}$$

La homografía H, que mapea los puntos de un objeto de un solo plano en el plano de la imagen tomada se describe entonces como $H = M[r_1 \quad r_2 \quad \boldsymbol{t}]$, donde: Ecuación (2.26)

$$\tilde{q} = sH\tilde{Q}' \tag{2.26}$$

Se usa $\tilde{Q}'$, porque como se estipuló antes, esta proyección es tomada sólo en 2 dimensiones.

Para poder calcular la homografía, es necesario usar múltiples imágenes del mismo objeto. Al mismo tiempo este procedimiento permite calcular las traslaciones y rotaciones de cada vista, así como los valores de los parámetros intrínsecos de la cámara [Zhang, 2000] y [Brown, 1971].

La homografía H relaciona las posiciones de los puntos de una imagen fuente con los puntos de una imagen destino (por ejemplo, una imagen en el mundo proyectada a manera de puntos en el plano de la cámara), según las siguientes ecuaciones: Ecuaciones (2.27) y (2.28) [Bradski y Kaehler, 2008]

$$P_{destino} = HP_{fuente} \tag{2.27}$$

$$P_{fuente} = H^{-1}P_{destino} \tag{2.28}$$

Siendo: $$P_{destino} = \begin{bmatrix} x_{dst} \\ y_{dst} \\ 1 \end{bmatrix} \ ; \ P_{fuente} = \begin{bmatrix} x_{fte} \\ y_{fte} \\ 1 \end{bmatrix}$$

Se puede ver que es posible hacer el cálculo de la homografía H sin saber nada de los parámetros intrínsecos de la cámara. De hecho, con múltiples homografías de variadas vistas y una lista de puntos con sus correspondencias es que los algoritmos de OpenCV, pueden hallar los valores intrínsecos. Se mostrará todo esto en el siguiente punto que es justamente, la calibración [Bradski y Kaehler, 2008] .

[7] Se debe tomar en cuenta que el vector **t** es un vector de tres componentes que son los desplazamientos del objeto en los ejes.

2.5.4 Calibración

La calibración de la cámara está relacionada con hallar los parámetros intrínsecos y de distorsión de la misma. La cantidad de parámetros que será necesario hallar son los cuatro intrínsecos (f_x, f_y, c_x, c_y) y los cinco parámetros de las distorsiones ya mencionados antes, tres radiales (k_1, k_2, k_3) y dos tangenciales (p_1, p_2).

Los parámetros intrínsecos están directamente ligados a la geometría 3D al igual que los parámetros extrínsecos. Lo parámetros propios de distorsión están ligados a la geometría 2D, entonces en general, se trata con los parámetros de estas dos clases separadamente. Tres puntos de las esquinas de tres cuadros en el tablero de ajedrez que se había mencionado, y cuyas coordenadas en el mundo real se conocen, son suficientes para resolver los cinco parámetros de distorsión tomando en cuenta que proporcionan seis ecuaciones (tres por eje). Es así, que una sola vista del tablero de ajedrez es necesario para calcular los parámetros de distorsión. Por supuesto, que más vistas son utilizadas para asegurar un cálculo robusto [Bradski y Kaehler, 2008].

Para el cálculo de los parámetros extrínsecos se necesita saber dónde está el tablero de ajedrez en el espacio. Esto requiere la descripción de los tres parámetros de rotación y los tres de traslación, que hacen un total necesario de seis vistas del tablero de ajedrez, ya que en cada vista el tablero se va a mover. Si se suma a esto las otras cuatro variables de los parámetros intrínsecos, se tienen un total de diez parámetros ligados a la geometría 3D que hay que resolver [Bradski y Kaehler, 2008].

Se tienen N esquinas y K imágenes del tablero de ajedrez (en diferentes posiciones):

- K imágenes del tablero proporcionan $2NK$ puntos conocidos en la imagen (se usa el multiplicador 2 porque cada punto en la imagen tiene dos coordenadas: una en x y otra en y)
- Ignorando los parámetros de distorsión por un momento, se tienen cuatro parámetros intrínsecos y $6K$ parámetros extrínsecos. (Dado que es necesario encontrar las coordenadas de los seis parámetros del tablero en cada una de las K vistas)
- Resolviendo se ve que se requieren $2NK \geq 6K + 4$ ó $(N-3)K \geq 2$ vistas del tablero

Al parecer, si $N = 5$ entonces $K = 1$. Así que sería necesaria sólo una imagen del tablero para resolver los parámetros mencionados. Esto no es totalmente verdad dado que se está usando el tablero de ajedrez para la calibración de la cámara tratando de lograr una homografía deseada para ajustarse a cada vista. La homografía sólo puede producir ocho parámetros o cuatro pares (x, y). Esto porque sólo cuatro puntos se pueden sacar de la expresión de una vista en perspectiva plana. Así que no importa cuántas esquinas sean detectadas en un plano, sólo cuatro de ellas tienen la información que se requiere [Bradski y Kaehler, 2008].

Por cada vista del tablero, la ecuación puede dar sólo cuatro esquinas de información o $(4-3)K > 1$, lo que muestra que $K > 1$. Esto implica que dos vistas de un tablero 3x3 es lo mínimo requerido para resolver el problema de calibración. Considerando el ruido y la estabilidad numérica, en general para tener mejores resultados serán necesarias entre 7 u 8 vistas. En la práctica, haciendo uso del software OpenCV y la función CalibrateCamera(), se pudo calibrar las cámaras haciendo uso del tablero mencionado. Nótese que las matemáticas

que se involucran en este cálculo, no forman parte del propósito de este trabajo, sin embargo, se mencionan los algoritmos del cálculo de [Zhang, 2000] y [Brown, 1971], utilizados para resolver los parámetros intrínsecos y las distorsiones respectivamente.

2.6 Visión Estereoscópica

En visión computacional se entiende la capacidad estereoscópica como la observación de una escena desde dos puntos diferentes, para el caso, desde dos cámaras en diferentes puntos. Ambas cámaras están situadas a una cierta distancia, pero en el mismo plano horizontal (en el caso más común). Este nuevo arreglo permite la observación de una imagen con diferente calidad, es decir se logra la percepción de la profundidad a la que se encuentra un objeto por medio de la triangulación de los puntos designados [Kanellakis, Kyritsis et al., 2015].

Este modo de visualización ya se conocía ya que se toma de la forma en la que los seres humanos perciben la profundidad del espacio con el uso de los dos ojos alineados horizontalmente.

Las computadoras logran percibir la profundidad encontrando correspondencias entre los puntos que han sido observados por la cámara de la izquierda y los puntos que han sido observados por la cámara de la derecha o viceversa. Con estas correspondencias y conociendo la línea base de separación entre las cámaras, es posible calcular la posición 3D de los puntos observados. Sin embargo, en la práctica, el cálculo estereoscópico involucra los siguientes pasos: [Bradski y Kaehler, 2008].

1. La remoción matemática de las distorsiones radiales y tangenciales de los lentes, este tema ya se había abordado antes.
2. El ajuste de los ángulos y las distancias entre cámaras lo que sería conocido como rectificación, tema que será abordado más adelante.
3. Hacer la correspondencia entre las imágenes. El producto de esta tarea es lo que se conoce como disparidad que es la diferencia entre las coordenadas en el eje x de las imágenes de la cámara izquierda y derecha: $x_l - x_r$[8]
4. Si se conoce el arreglo geométrico de las cámaras, entonces se puede armar un mapa de disparidad por triangulación. Este proceso es llamado re-proyección y como resultado proporciona un mapa de profundidades.

2.6.1 Triangulación

Se puede asumir que se tiene un arreglo de imágenes perfectamente alineadas, sin distorsión y con la línea base conocida tal como se muestra en la Figura 2.10. En esta Figura se puede ver que las dos cámaras tienen imágenes coplanares, ejes ópticos paralelos (o rayos principales paralelos, entendiéndose como rayo principal el centro de proyección O que pasa a través del punto principal c) y distancias focales iguales $f_l = f_r$. También se puede asumir que los puntos principales c_x^{left} y c_x^{right} están calibrados y tienen las mismas coordenadas en pixeles en las imágenes de la izquierda y derecha respectivamente. [Bradski y Kaehler, 2008].

[8] Por simplicidad se menciona a la izquierda como $l = (left)$ y a la derecha $r = (rigth)$

En estas condiciones, las imágenes estarán alineadas por filas de pixeles, es decir, que cada línea de pixeles en una cámara estará alineada con la fila de pixeles correspondiente en la otra cámara. Se dice que este arreglo es paralelo desde la vista frontal. Se incluye también el punto P, en el mundo físico, cuyas coordenadas horizontales serán descritas por x^l y x^r, respectivamente a las cámaras izquierda y derecha [Bradski y Kaehler, 2008].

En este caso simplificado, tomando x^l y x^r como las coordenadas horizontales, la profundidad donde se encuentra la imagen es inversamente proporcional a la disparidad entre las vistas. Esta disparidad se define simplemente por $d = x^l - x^r$. Se puede observar que es simple llegar a este resultado usando triángulos semejantes, Figura 2.10 [Bradski y Kaehler, 2008].

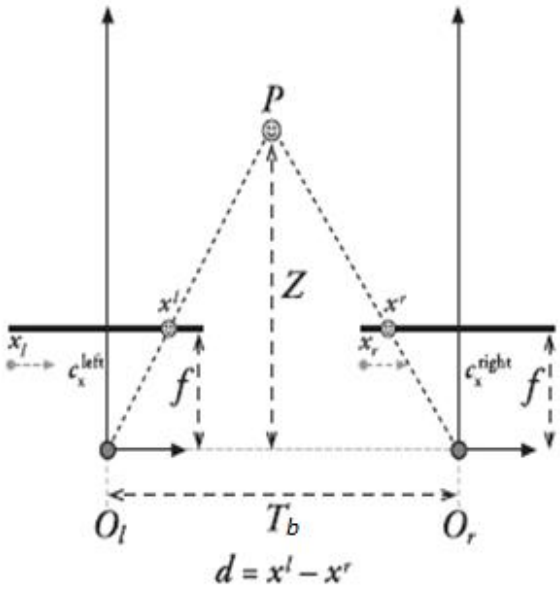

Figura 2.10 - Arreglo de imágenes en el caso estereoscópico ideal.

De la Figura 2.10 también se tiene: Ecuación (2.29)

$$\frac{T_b - (x^l - x^r)}{Z - f} = \frac{T_b}{Z}$$

$$z = \frac{fT_b}{x^l - x^r} \tag{2.29}$$

Donde:

$T_b = distancia\ entre\ cámara, línea\ de\ base\ o\ traslación$

$Z = profundidad$

$f = distancia\ focal$

$x_l, x_r = planos\ de\ imagen$

$x^l, x^r = coordenadas\ en\ x$

$P = punto\ principal\ observado$

$d = disparidad$

$O_l, O_r = centros\ de\ proyección$

Ya que la profundidad es proporcional a la disparidad, cuando la disparidad se acerca a cero, la profundidad se hace muy larga y eventualmente difícil de calcular. Cuando existe mucha disparidad, por otro lado, la profundidad se mantiene en valores pequeños mesurables. La consecuencia de este comportamiento es que el sistema estereoscópico no tiene una muy alta resolución de detección de profundidades. Lo que muestra que, para poder calcularla, cada objeto debería estar dentro del rango mesurable de lejanía desde el punto donde se montan las cámaras [Bradski y Kaehler, 2008].

2.6.2 Geometría Epipolar

En esencia, esta geometría combina los modelos de dos cámaras *pinhole* (una por cada lado) y dos puntos llamados epipolos. Por otro lado, para cada cámara existe un punto de proyección que antes se presentaron como O_l o O_r, según el lado que corresponda, y un par de planos de proyección π_l y π_r respectivamente. El punto P en el mundo físico tiene una proyección en cada uno de los planos de proyección de las cámaras, a estos puntos se los va a denominar p_l y p_r.

Ahora se puede introducir dos nuevos puntos de interés que son los epipolos. Un epipolo e_l (respecto de e_r) en el plano de imagen π_l (respecto de π_r), es definido como la imagen del centro de proyección de la otra cámara O_r (respecto de O_l). El plano en el espacio formado por el punto observado P y los dos epipolos e_l y e_r (o equivalentemente a través de dos centros de proyección O_l y O_r) es llamado plano epipolar y las líneas formadas por $p_l e_l$ y $p_r e_r$ son llamadas líneas epipolares [Bradski y Kaehler, 2008].

Para ilustrar esto que se comenta, se muestra la Figura 2.11 [Bradski y Kaehler, 2008]:

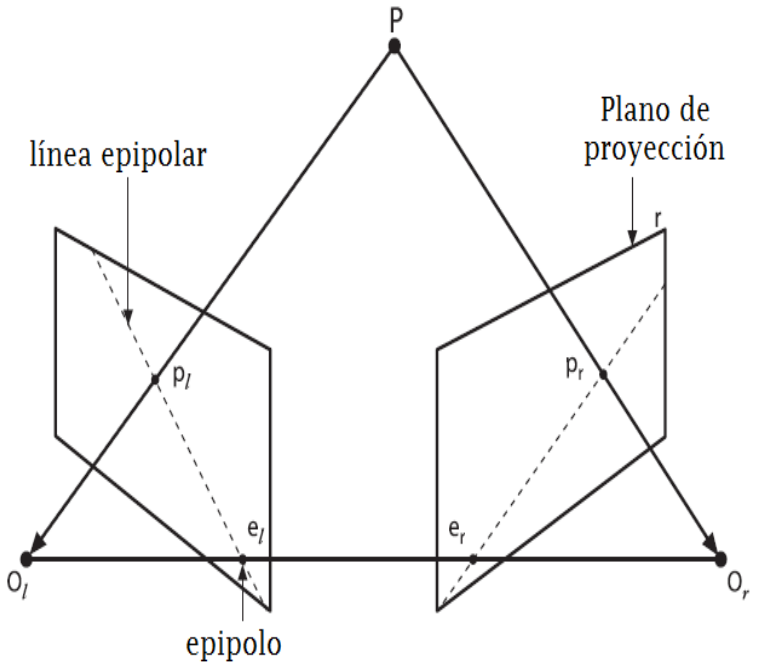

Figura 2.11 - Geometría Epipolar.

Se puede ver que la línea epipolar que es formada por el epipolo e_l y la proyección de la línea de visión entre O_r y P (donde se encuentran múltiples puntos p_r) permite después conocer lo que en visión computacional es la medida epipolar [Bradski y Kaehler, 2008].

La medida epipolar dice que es posible buscar puntos en 2D usando dos cámaras y encontrar el valor en una sola dimensión ya que la otra estaría descansando en las líneas epipolares. Esto ayuda a ahorrar recursos computacionales y evitar puntos que, en algunos casos, no guardan ninguna correspondencia con la imagen que se está captando [Zisserman y Hartley, 2004].

2.6.3 Matrices esencial y fundamental

La matriz esencial E contiene la información acerca de los valores de rotación y traslación que relacionan las dos cámaras. Por otro lado, la matriz fundamental F contiene información sobre los parámetros intrínsecos que se había mencionado antes. Cabe aclarar que, la matriz F relaciona las dos cámaras en coordenadas de pixel [Bradski y Kaehler, 2008].

Se escogen las coordenadas centradas en O_l que pertenecen a la cámara izquierda (como podrían ser las de la derecha). En estas coordenadas, el punto observado sería P_l y el origen de la otra cámara estaría situado en T. Este arreglo se puede describir por la ecuación (2.7) tomando en cuenta las dos cámaras: Ecuación (2.30) [Bradski y Kaehler, 2008]

$$P_r = R(P_l - T) \tag{2.30}$$

El punto clave en esto es añadir a este arreglo el plano epipolar. Un plano puede ser representado de varias formas, pero para el propósito planteado, se puede decir que se tienen x puntos en un plano y un vector normal a ellos n que pasa por el punto a, describiendo la siguiente ecuación: Ecuación (2.31)

$$(x - a) \cdot n = 0 \tag{2.31}$$

En el caso más simple, se sabe que el plano epipolar pasa por P_l y por T, de ahí que un vector perpendicular a estos sería simplemente $P_l \times T$, el cual se puede reemplazar en lugar de n. Así la ecuación en la que todos los posibles puntos que pasan por P_l hacia T y que contienen ambos vectores sería: Ecuación (2.32) [Bradski y Kaehler, 2008]

$$(P_l - T)^T (T \times P_l) = 0 \tag{2.32}$$

Nótese que acá el producto punto anterior se reemplaza por un producto entre matrices por la transpuesta del vector normal. Si se combina la ecuación (2.30) y la (2.32), se tiene: Ecuación (2.33)

$$(R^T P_r)^T (T \times P_l) = 0 \tag{2.33}$$

Sabiendo que: $R^T = R^{-1}$

Para evitar tener el producto vectorial, se puede reemplazar por un producto punto según:

$$T \times P_l = SP_l \qquad donde: S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & -T_x & 0 \end{bmatrix} \tag{2.34}$$

Lo que da: $(P_r)^T RSP_l = 0$, en esta ecuación lo que interesa es el componente RS que resulta ser la **matriz esencial *E*** [Bradski y Kaehler, 2008].

La matriz esencial E contiene toda la información acerca de la geometría de las dos cámaras relativa una a la otra, pero no contiene información acerca de las cámaras en sí mismas. En este trabajo, en el momento de la práctica, se hizo usó la matriz fundamental.

Para encontrar la relación entre un pixel de una imagen y la correspondiente línea epipolar en la otra imagen, se debe introducir la información de parámetros intrínsecos de las dos cámaras. Para hacer esto, para p (coordenada en pixel) se sustituye q y los valores intrínsecos de la cámara. Se sabe que un punto $q = Mp$ (donde M es la matriz de valores intrínsecos). Entonces en la ecuación de la matriz esencial donde $RS = E$: Ecuación (2.35) [Bradski y Kaehler, 2008]

$$(p_r)^T E p_l = 0 \tag{2.35}$$

Reemplazando la igualdad $(p_r)^T = (q_r M_r^{-1})^T$ y $p_l = q_l(M_l^{-1})$ de la ecuación (2.35) encontramos la ecuación de la matriz fundamental, Ecuación (2.36):

$$F = (M_r^{-1})^T E (M_l^{-1}) \tag{2.36}$$

Finalmente, se puede relacionar: Ecuación (2.37) [Bradski y Kaehler, 2008]

$$q_r^T F q_l = 0 \tag{2.37}$$

En la práctica se observa que la matriz F es de rango 2, tiene siete parámetros, dos de los cuales son los correspondientes epipolos y tres corresponden a la homografía que relaciona los planos de las dos cámaras.

En OpenCV existe una función que relaciona todos estos parámetros y que permite hallar la matriz fundamental. En la práctica, será necesario establecer los puntos exactos para el cálculo de esta matriz. En las siguientes secciones se analiza cómo es posible conseguir esos puntos y correspondencias.

2.6.4 Calibración Estereoscópica

La calibración estereoscópica es el proceso por el cual es posible calcular la relación geométrica de las dos cámaras en el espacio [Bradski y Kaehler, 2008]. Esta calibración depende de la matriz de rotación R y el vector de traslación T entre las dos cámaras. En el apartado 2.5, se vio que para cualquier punto en P en coordenadas 3D es posible hallar los valores de calibración de una cámara y la otra por separado. Sin embargo, el cálculo de la calibración de ambas cámaras por separado se ve afectado por problemas de ruido y redondeos en cifras.

Está claro que para ambas cámaras los valores de la matriz R y el vector T deben ser iguales [Bradski y Kaehler, 2008]. La solución de este problema está en un algoritmo que toma la media de los valores hallados para R y T y los plantea como solución inicial aproximada. El algoritmo iterativo que ayuda en este proceso es el de Levenberg-Marquardt[9] que encuentra los mínimos locales del error de la re-proyección de las esquinas del tablero de ajedrez en las dos cámaras.

Simplemente como resumen se puede agregar que la calibración estereoscópica calcula la matriz de rotación R y el vector de traslación T que permite poner la cámara izquierda y la derecha en el mismo plano con el objetivo de tener imágenes coplanares [Bradski y Kaehler, 2008].

En el siguiente punto se verá cómo lograr que las imágenes queden también "alineadas" respecto de, como se comentó, las filas de pixeles entre una cámara y la otra.

2.6.5 Rectificación Estereoscópica

El objetivo de la rectificación es lograr que las líneas epipolares se alineen para que el cálculo de la disparidad se haga de forma más sencilla. Esto se logra cuando las filas de pixeles correspondientes entre la imagen de la izquierda y la imagen de la derecha se alinean exactamente. Para comprender mejor lo mencionado se presentan las Figuras 2.12 [Zisserman y Hartley, 2004] y 2.13 [Kanellakis, Kyritsis et al., 2015] a continuación:

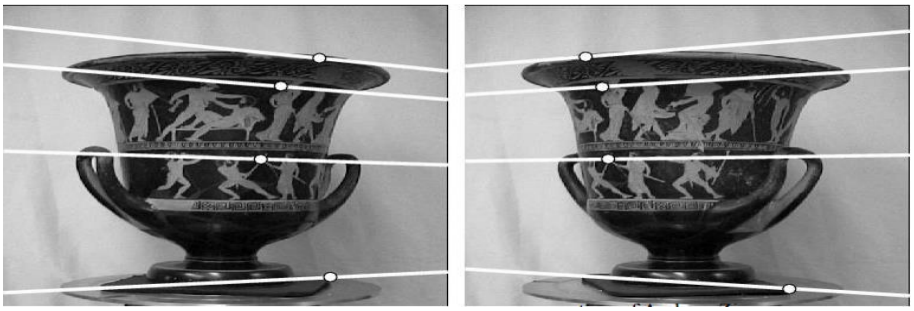

Figura 2.12 - Líneas Epipolares en imágenes tomadas desde dos cámaras dispuestas en modo estereoscópico. Véanse los puntos de correlación.

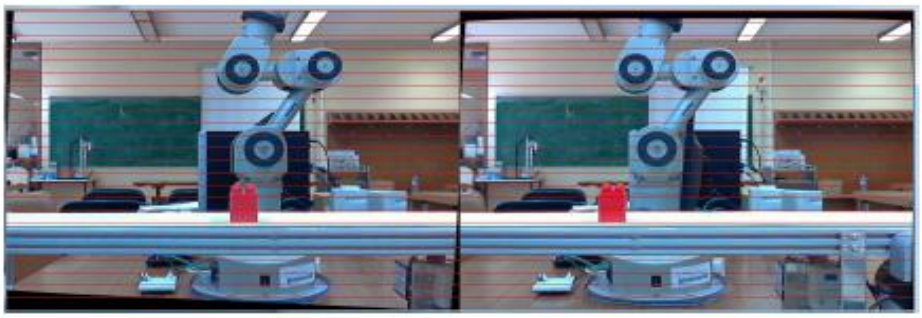

Figura 2.13 - Líneas Epipolares en imágenes tomadas desde dos cámaras dispuestas en modo estereoscópico.

[9] Este algoritmo es implementado a través de OpenCv llamando la función cvStereoCalibrate()

Véase claramente que estas imágenes están rectificadas mostrando que las filas de pixeles entre las dos tomas se corresponden.

Las Figuras anteriores muestran el objetivo de la rectificación, la necesidad de reproyectar la imagen de las dos cámaras en un mismo plano, con las líneas perfectamente alineadas en una conFiguración paralela y frontal.

Se quiere imágenes en las que las filas de las dos cámaras estén alineadas después de la rectificación para que la correspondencia estereoscópica[10] sea más confiable y requiera menos procesamiento [Bradski y Kaehler, 2008].

Dadas las matrices de rotación y traslación (R, T) entre dos imágenes estereoscópicas, el algoritmo de Bouguet's para la rectificación de las mismas, minimiza la cantidad de cambios en las re-proyecciones de las dos imágenes, de modo que también se minimizan las distorsiones. Este algoritmo también trata de maximizar el área de visión de la imagen [Bradski y Kaehler, 2008].

Para minimizar las distorsiones, la matriz de rotación R que rota del plano de la cámara derecha al plano de la cámara izquierda, se reparte en dos mitades para las cámaras. Estos resultados se denominan matrices r_l y r_r[11]. Cada cámara rota sólo la mitad de la rotación total, así que el rayo principal de proyección de cada cámara queda siempre paralelo al vector suma del rayo original. Tal rotación entonces pondrá las cámaras en el mismo plano, alineándolas, pero no mediante las filas. Para calcular R_{rect}, se crea una matriz de rotación empezando por la dirección del epipolo e_1. Tomando como punto principal (c_x, c_y) como el origen de la imagen de la izquierda, la dirección del epipolo (normalizada) va directamente a lo largo del vector de traslación entre los centros de proyección de las dos cámaras: Ecuación (2.38) [Bradski y Kaehler, 2008]

$$e_1 = \frac{T}{\|T\|} \tag{2.38}$$

El siguiente vector e_2, deberá ser ortogonal a e_1. Para e_2 se escoge una dirección ortogonal al rayo principal. Esto se consigue usando el producto punto de e_1 con la dirección del rayo principal y después se normaliza este valor para tener un valor unitario. Ecuación (2.39)

$$e_2 = \frac{[-T_y \quad T_x \quad 0]^T}{\sqrt{T_x^2 + T_y^2}} \tag{2.39}$$

El tercer vector ortogonal se forma simplemente con el producto cruz de los otros dos vectores: Ecuación (2.40)

$$e_3 = e_1 \times e_2 \tag{2.40}$$

[10] Entendiendo que la correspondencia estereoscópica se trata de encontrar el mismo punto en las dos vistas de las cámaras.
[11] Como se había propuesto antes *l* = *left* para la izquierda y *r* = *right* para la derecha

Así la matriz resultante será aquella que rota la cámara izquierda alrededor de su centro de proyección de modo que las líneas epipolares quedan horizontales y los epipolos por consecuencia quedan en el infinito.: Ecuación (2.41) [Bradski y Kaehler, 2008]

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix} \tag{2.41}$$

Después para lograr la alineación de las filas de las dos cámaras se puede usar: Ecuaciones (2.42) y (2.43)

$$R_1 = R_{rect} r_l \tag{2.42}$$

$$R_r = R_{rect} r_r \tag{2.43}$$

Por otro lado, las matrices de proyección pasan un punto en 3D en coordenadas homogéneas 2D como sigue: Ecuación (2.44) [Bradski y Kaehler, 2008]

$$P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{2.44}$$

Donde las coordenadas de la pantalla pueden ser calculadas como: $(\frac{x}{w}, \frac{y}{w})$. Los puntos en 2D pueden también ser re-proyectados en tres dimensiones mediante las coordenadas de la pantalla y la matriz intrínseca como: Ecuación (2.45) [Bradski y Kaehler, 2008]

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & b & a \end{bmatrix} \tag{2.45}$$

Donde $b = -1/T_x$ y $a = (c_x - c'_x)/T_x$

Aquí el punto principal x en la imagen de la derecha sería c'_x. Si los rayos principales intersectan en el infinito, entonces $c_x = c'_x$, así a se hace 0. Finalmente, dado un punto en 2D homogéneo y su disparidad asociada d, se puede re-proyectar el punto en 3D usando: Ecuación (2.46) [Bradski y Kaehler, 2008]

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \tag{2.46}$$

Las tres coordenadas serían entonces: $(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W})$.

2.6.6 Correspondencia Estereoscópica

La correspondencia en sistemas estereoscópicos se trata de la formación de pares de puntos que en dos vistas de diferentes cámaras resultan ser el mismo. Esta correspondencia puede ser calculada solamente en las áreas visuales en las cuales las vistas de dos cámaras se traslapan. Una vez más, esta es la razón por la que cuando los planos de las vistas son coplanares, es más sencillo hallarla. Es por esto que cuando se saben las coordenadas físicas de las cámaras y los tamaños de los objetos en la escena, se puede derivar la medición de la profundidad desde la triangulación por medio de la medida de disparidad $d = x^l - x^r - (c_x^l - c_x^r)$ (sabiendo que los rayos principales intersectan al infinito) [Bradski y Kaehler, 2008].

En el software OpenCV existe una función que implementa de forma rápida y efectiva el algoritmo de correspondencia. Esta función cvFindStereoCorrespondenceSGBM(), usa una "suma de diferencias absolutas" (SAD por sus siglas en inglés), mediante cuadros pequeños que buscan una correspondencia de puntos entre las imágenes rectificadas de la izquierda y la derecha. Este algoritmo busca solamente una correspondencia relevante (de alta textura) entre los puntos de ambas imágenes. Es así que es usado generalmente en escenas muy texturizadas.

Existen tres pasos que sigue este algoritmo al momento de ser implementado [Bradski y Kaehler, 2008]:

1. Se aplica un filtro para normalizar el brillo de la imagen y mejorar la textura
2. Las correspondencias se buscan a lo largo de la línea epipolar horizontal usando una ventana SAD
3. Se aplica otro filtro para eliminar algunos puntos que sean falsas correspondencias

En el primer paso se aplica un filtro con el que se mejora el brillo y la textura. El objetivo principal de este filtro es el de "suavizar" la imagen para reducir el ruido en ella. Existen muchos métodos para suavizar una imagen. Los más comunes son las técnicas que pueden ser aplicadas en OpenCV como [Cyganek y Siebert, 2009]:

1. El método homogéneo
2. De curva Gaussiana
3. La media
4. El modo bilateral

El aplicar un filtro es igual que la convolución, es el tratamiento de una matriz por otra que se llama kernel. Por ejemplo, el valor de 5x5 que es usado en un kernel homogéneo, es conocido como un filtro normalizador. Para más información referirse al apartado 2.3

En el segundo paso, la correspondencia es calculada por una ventana SAD. Para cada característica en la imagen de la izquierda, debe corresponder alguna de la derecha en la misma fila. Después de la rectificación, cada fila es una línea epipolar, así que es la misma en ambas imágenes [Bradski y Kaehler, 2008]. Al mismo tiempo ambos puntos deberán tener suficiente textura para poder ser detectados y por supuesto no estar ocluidos.

Para que este algoritmo empiece a buscar correspondencias será necesario estipular la cantidad de disparidades a buscar y el valor de disparidad mínimo. Estos valores se definen

como el Horóptero [Bradski y Kaehler, 2008], que vendría a ser el volumen que se cubre en la búsqueda de disparidad de un objeto.

Puede que algunos puntos no puedan ser encontrados en la otra imagen, por el ruido o porque están ocluidos, a estos puntos se los llama inserciones. Estas inserciones no afectan el orden de los objetos, así que pueden ser ignorados.

Finalmente, se puede notar que mientras más pequeño sea el valor de incremento de la disparidad Δd, se puede mejorar la resolución de medición de las profundidades ΔZ según la siguiente ecuación: Ecuación (2.47) [Bradski y Kaehler, 2008]

$$\Delta Z = \frac{Z^2}{fT}\Delta d \tag{2.47}$$

Después de la correspondencia el siguiente paso es el post-filtrado. Este filtrado elimina las correspondencias falsas entre las dos vistas, especialmente las que se van a hallar en los bordes. En el caso de los bordes, suelen aparecer muchos errores ya que el procesamiento de estas correspondencias depende mucho de la cantidad de iluminación, sobre todo cuando esto impulsa un contraste máximo.

Por último, con toda esta información se podría hacer un mapa de profundidades desde la re-proyección en 3D.

2.6.7 Mapas de profundidades desde la reproyección 3D

Para poder hacer el reconocimiento de la imagen deseada, el último punto es hacer uso del mapa de profundidades para reconstruir el objeto en 3D. Se va hablar ahora de la matrix de reproyección Q que fue vista en la ecuación (2.45). Usando esta matriz se puede definir las coordenadas 3D de los puntos como: $\left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}\right)$ [Bradski y Kaehler, 2008]. Para poder hacer uso de estas características simplemente es necesario extraer el valor de la profundidad multiplicando matrices.

En OpenCV se puede usar la función cvReprojectImageto3D() para reconstruir el objeto. Esta rutina transforma cada pixel (x, y) a lo largo de la imagen tomando en cuenta la disparidad hallada o el vector $[x\ y\ d]^T$ mediante el uso de la matriz Q. La salida es una matriz de imagen del mismo tamaño de la que se usa como entrada. En la Figura 2.14 [Bradski y Kaehler, 2008], se muestran ejemplos de reconstrucción 3D.

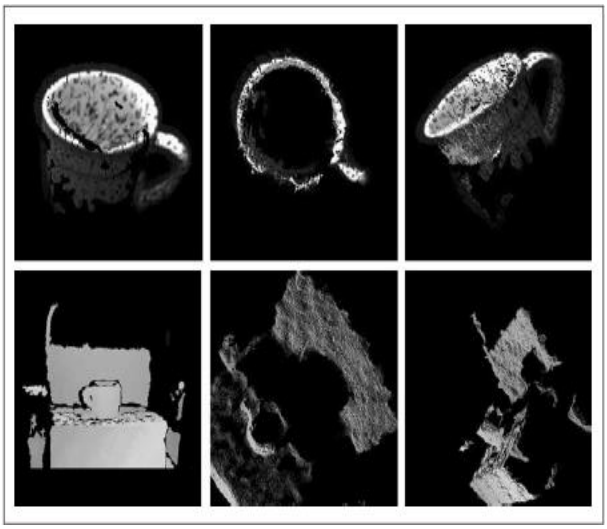

Figura 2.14 - Reconstrucción 3D a partir de valores de disparidad.

2.7 Detección de bordes, método propuesto por Canny

Una vez que una cámara toma una imagen, se presenta el problema del reconocimiento del patrón buscado. Es de suponer que no todos los objetos que pudieran ser tomados por la cámara son de interés, es así que, la forma de los objetos resulta ser la información más importante para la detección del objeto patrón deseado.

El propósito de la detección de bordes es en general, la reducción de la cantidad de información de una imagen, mientras se preservan las propiedades estructurales de la misma para su procesamiento posterior. Existen muchos algoritmos de detección de bordes, sin embargo, el más confiable y que se usa como estándar fue planteado por Canny, 1986.

El procedimiento planteado por Canny sigue los siguientes criterios:

- La de minimizar la detección falsa de bordes, esto se logra a partir de maximizar el radio de detección de ruido
- Los bordes localizados en la imagen deberían poder estar lo más cerca posible de los bordes reales
- Un borde real no debería resultar en más de un borde detectado.

El algoritmo utilizado [Canny, 1986]:

1. Aplicación de un filtro de suavizado: con el objetivo de remover ruido o información no relevante
2. Encontrar gradientes: Los bordes de la imagen deben ser marcados donde existan gradientes de mayor valor
3. La no-supresión de máximos: Todos los máximos valores de gradiente deberán ser marcados como bordes
4. Detección de bordes por histéresis: Los bordes finales se determinan por la supresión de todos los bordes que no estén conectados por algún otro.

Se verán un poco más en detalle los pasos más relevantes.

2.7.1 Cálculo de gradientes

El algoritmo de Canny, básicamente busca bordes donde la intensidad en la escala de grises cambia de más de un pixel a otro. Estas áreas se encuentran determinadas por gradientes en la imagen. El gradiente de cada pixel en la imagen ya suavizada, se puede determinar aplicando lo que se conoce como el Operador de Sobel [Sobel y Feldman, 1968]. Se debe mencionar que fueron Sobel y Feldman quienes generaron la base de detección de bordes que luego Canny usaría como primera propuesta para sus mejoras.

Según Sobel y Feldman, 1968, las magnitudes del gradiente, conocidas como intensidad de los bordes, podrán ser determinadas por la siguiente ecuación: Ecuación (2.48)

$$|G| = \sqrt{g_x^2 + g_y^2} \tag{2.48}$$

Donde g_x y g_y son gradientes en las direcciones x y y respectivamente. También será necesario guardar las direcciones de los bordes dado que los resultados de bordes detectados pueden quedar un poco alejados de los bordes reales: Ecuación (2.49)

$$\emptyset = \tan^{-1}\left(\frac{|g_y|}{|g_x|}\right) \tag{2.49}$$

2.7.2 La no-supresión de máximos

El propósito de este paso es convertir los bordes suavizados de la imagen en bordes definidos. Esto se puede lograr preservando los máximos locales en el gradiente de la imagen y eliminando todos los demás [Sobel y Feldman, 1968].

Una vez calculado el ángulo del gradiente, se lo debe aproximar al paso más cercano de 45°. Para que se entienda mejor, se dice que un pixel tiene 8 pixeles alrededor que lo rodean y a los que se les llama vecinos. Cada uno dispuesto desde el centro cada 45°. Si, por ejemplo, el ángulo del gradiente es 90° los pixeles de comparación serían el de arriba y el de abajo. En el caso de ángulo menor a 45°, los pixeles de comparación serían los de izquierda y derecha. Es en esta comparación que se eliminarían todos los pixeles con valores de gradiente bajos y se conservan los que tengan valores de gradiente máximos [IITD, 2016].

En la Figura 2.15 [IITD, 2016], se ve que todos los cuadros resaltados en blanco, serán bordes a conservarse. Estos pixeles tendrán el valor máximo del gradiente. Una vez hecha la inspección, el algoritmo desecha los de menores gradientes y como resultado el borde queda definido y deja de ser suavizado.

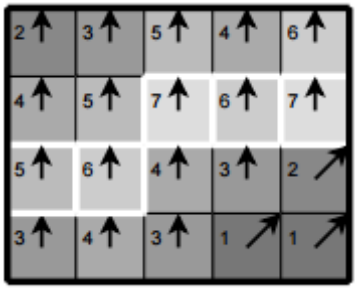

Figura 2.15 - Ejemplo de la no-supresión de máximos.

2.7.3 La detección de bordes por histéresis

Los bordes más pigmentados o “fuertes” son interpretados como bordes reales y pueden ser inmediatamente incluidos en la imagen de bordes final. Los bordes menos pigmentados o “débiles” son incluidos sólo si están conectados a los bordes fuertes.

La secuencia es implementada usando el análisis BLOB (por sus siglas en inglés: Binary Large Object). Los pixeles que se detectan como bordes se dividen en BLOBs conectados usando la lógica de los 8 pixeles vecinos. Aquellos que contengan por lo menos un pixel fuerte o muy pigmentado serán preservados, mientras que los demás serán descartados [IITD, 2016].

2.8 Momentos y comparación de contornos

Los momentos son funciones definidas que se desarrollan a partir del análisis de imágenes como Figuras planas en dos dimensiones. La base matemática de los momentos invariantes en Figuras planas fue planteada por [Hu, 1962]. Estas funciones resumen varios aspectos de la forma de los objetos y su tamaño. Se incluye este tema ya que será útil para el reconocimiento de patrones por comparación de contornos.

Se define el momento de orden $(p+q)$ de un contorno en términos de Riemann de tal forma que Ecuación (2.50) [Medina, 2015]:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q \sigma(x,y)\, dx\, dy \tag{2.50}$$

Donde: $p, q = 0,1,2 \ldots$

Desde esta definición se puede observar que m_{00} es básicamente la longitud de pixeles del contorno. Matemáticamente, algunos autores también toman el momento m_{00} como el área dentro del contorno. En el caso de este trabajo, sin embargo, se estará trabajando con contornos vacíos que sólo muestran la forma del objeto deseado, por tanto, se toman como equivalentes ya que la longitud y el área son lo mismo en un espacio discreto de pixeles [Bradski y Kaehler, 2008].

En la ecuación (2.49), la función definida como $\sigma(x,y)$ desde la teoría de Hu, sería la que describe la forma del objeto para el cual se calculan los momentos. En este caso, esta función contiene la información del valor del pixel en la coordenada (x,y). En una imagen binaria (blanco y negro) esta función tendrá el valor de 1 o 0, según se convenga que el contorno se muestre en blanco o negro. Al tratar una imagen digital, en [Bradski y Kaehler, 2008] las integrales con sustituidas por un sumatorio Ecuación (2.51)

$$m_{pq} = \sum_{i=1}^{n} \sigma(x,y) x^p y^q \tag{2.51}$$

En OpenCV, se tiene formulada una función que calcula los momentos de un contorno definida como CvgetHuMoments(). Esta función calcula los momentos según se mencionó antes, basada en la matemática propuesta por Hu. La característica más importante que se mantiene es que mediante este cálculo se obtienen momentos invariantes que no se afectan por la traslación del objeto ni por el tamaño en la imagen [Bradski y Kaehler, 2008].

Los momentos de primer orden son de principal interés ya que con ellos es posible calcular el centroide de un objeto, estos momentos están dados por m_{10}, m_{01}.

El momento central es básicamente el mismo que se describe en la ecuación (2.51), excepto que los valores de x y y usados en las fórmulas están desplazadas por los valores principales: Ecuación (2.52) [Bradski y Kaehler, 2008]

$$\mu_{p,q} = \sum_{i=0}^{n} \sigma(x,y)(x-\bar{x})^p(y-\bar{y})^q \tag{2.52}$$

Donde los valores $\bar{x}$ y $\bar{y}$ son las coordenadas del centro de masa. El cálculo de momentos respecto al centro de masa del objeto permite obtener valores invariantes respecto a la traslación de los objetos. Los valores de los momentos de primer orden no cambiarán en estos puntos, así: Ecuación (2.53) y ecuación (2.54) [Bradski y Kaehler, 2008]

$$\bar{x} = m_{10}/m_{00} \tag{2.53}$$

$$\bar{y} = m_{01}/m_{00} \tag{2.54}$$

Finalmente, se puede definir la orientación de un objeto en una imagen, como el ángulo de un eje que pasa a través del objeto de modo que el momento de segundo orden respecto de ese eje sea mínimo, [Spong, et al., 2006]. Esto es el equivalente, se podría decir, al eje de menor inercia del objeto en dos dimensiones.

Por una línea dada en la imagen, el momento de segundo orden del objeto está dado por: Ecuación (2.55) [Spong, et al., 2006]

$$\tau = \sum_{i=0}^{n} d^2(x,y)\sigma(x,y) \tag{2.55}$$

Donde $d(x,y)$ es la distancia mínima desde el pixel de coordenadas (x,y) a la línea. Se minimiza esta función respecto de todas las posibles líneas en el plano de la imagen derivando parcialmente e igualando a cero. Para hacer esto, se usan los parámetros σ, θ que describen una línea ejemplo y se deriva parcialmente la ecuación (2.55) respecto de estos parámetros, figura 2.16.

Sea la línea descrita por la ecuación:

$$d(x,y) = x\cos\theta + y\sin\theta - \rho \tag{2.56}$$

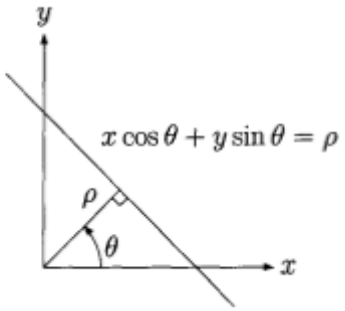

Figura 2.16 - Representación gráfica de la línea planteada por la ecuación 2.55.

Se calcula la derivada parcial respecto de ρ como: [Spong, Hutchinson y Vidyasagar, 2006]

$$\frac{d}{d\rho}\tau = \frac{d}{d\rho}\sum_{i=0}^{n}(x\cos\theta + y\sin\theta - \rho)^2\sigma(x,y)$$
$$= -2(\cos\theta\, m_{10} + \sin\theta\, m_{01} - \rho m_{00})$$
$$= -2m_{00}(\bar{x}\cos\theta + \bar{y}\sin\theta - \rho) \qquad (2.57)$$

Se observa que el término entre paréntesis, es la ecuación de una línea que pasa por el centro de masa. Ahora se deriva la ecuación (2.54) respecto al parámetro θ, sabiendo que la línea debe pasar por el centro de masa para hacer la función mínima donde $x' = 0$, $y' = 0$ (una línea que pasa por el origen) se hace el siguiente reemplazo, donde: $x' = x - \bar{x}$ y $y' = y - \bar{y}$

$$x'\cos\theta + y'\sin\theta = 0$$

$$\tau = \sum_{i=0}^{n}(x'\cos\theta + y'\sin\theta)^2\,\sigma(x,y)$$

$$= \cos^2\theta\sum_{i=0}^{n}(x')^2\sigma(x,y) + 2\cos\theta\sin\theta\sum_{i=0}^{n}(x'y')\sigma(x,y) + \sin^2\theta\sum_{i=0}^{n}(y')^2\sigma(x,y)$$

$$= \cos^2\theta\,\mu_{20} + 2\cos\theta\sin\theta\,\mu_{11} + \sin^2\theta\mu_{02}$$

Aplicando la derivada parcial y usando identidades trigonométricas [Spong, Hutchinson y Vidyasagar, 2006]:

$$\frac{d}{d\rho}\tau = \frac{d}{d\theta}\cos^2\theta\,\mu_{20} + 2\cos\theta\sin\theta\,\mu_{11} + \sin^2\theta\mu_{02}$$

$$\tan 2\theta = \frac{\mu_{11}}{\mu_{20}-\mu_{02}} \qquad (2.58)$$

Lo que permite conocer la orientación de la línea que pasa por el centro de masa y una idea de la orientación del objeto en la imagen. Un ejemplo de este cálculo es proporcionado por [Spong, Hutchinson y Vidyasagar, 2006] (Figura 2.17).

Figura 2.17 - Imagen que muestra los centroides y la orientación de cada objeto en la imagen.

2.9 Robot neumático de 5 grados de libertad

El sistema de visión estereoscópico que se presenta, se pensó desde un inicio como agregado de algún sistema robótico el cual fuera capaz de reconocer piezas para manipularlas en el espacio de trabajo válido del robot.

En LAMECC se cuenta con un robot de 5 grados de libertad neumático desarrollado por Sarmanho, 2014. Este robot que Sarmanho denominó RP5GDL (robot neumático de 5 grados de libertad), fue usado para la integración del sistema de visión a fin de lograr que algunas piezas pudieran ser manipuladas por medio de la garra que posee. A continuación, se detallan las características básicas del robot.

El robot desarrollado por Sarmanho, 2014, posee 5 grados de libertad con sus juntas en la conFiguración RPP:RR, donde R representa una junta rotacional y P una prismática. Como se había mencionado, este robot llamado RP5GDL, es representado de forma simplificada en la Figura 2.18 obtenida del trabajo de Sarmanho, 2014.

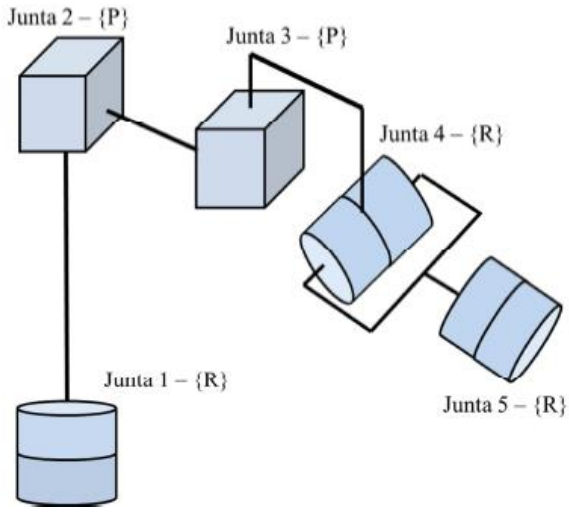

Figura 2.18 - Representación simplificada del robot RP5GDL.

El RP5GDL se desarrolló como una estructura mecánica rígida, dando preferencia a la utilización de piezas comerciales para su construcción. Posee una conFiguración cilíndrica cuya junta base es rotacional seguida de dos juntas prismáticas. Este tipo de conFiguración

reduce el acoplamiento dinámico [Sarmanho, 2014]. En el trabajo más adelante, se confirma que esta estructura le confiere al robot la posibilidad de hacer trabajos de manipulación.

La estructura mecánica del manipulador se puede ver a continuación en la Figura 2.19 [Sarmanho, 2014]. Se pueden diferenciar claramente las 5 juntas: 3 rotacionales y 2 prismáticas.

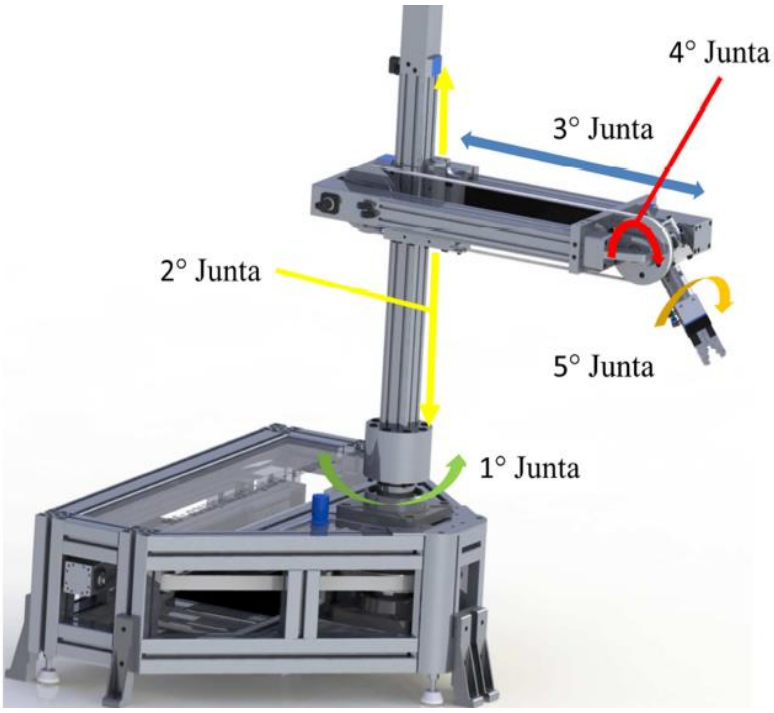

Figura 2.19 - Robot cilíndrico RP5GDL, vista de las juntas.

El sistema robótico presentado, usa como placa de procesamiento prototipo la llamada dSpace (placa DS-1104, cuyo fabricante es precisamente la compañía dSpace) que se puede comunicar directamente con Matlab y la herramienta de simulación y desarrollo de interfaces gráficas Simulink® [MathWorks, 2011]. La placa mencionada usa comunicación serial con protocolo propietario y en base al estándar RS-485. Al mismo tiempo, la comunicación de dicha placa de control con los actuadores y sensores se lleva a cabo mediante el protocolo CAN y SPI respectivamente.

En Leonardelli, 2015, la última mejora realizada al sistema fue la integración de la programación hecha en Matlab a través de Simulink, con un sistema de automatización propietario de Rockwell Automation®. El objetivo inicial de esta mejora fue lograr la programación *Online* de las trayectorias del robot, es decir, permitir que la placa dSpace escriba y lea datos de la trayectoria del robot en tiempo real. Una vez cumplido este objetivo, Leonardelli, agrega un OPC como interfaz de comunicación para el intercambio de datos con el software RSLogix 5000 y un controlador programable industrial de la gama *CompactLogix*, ambos desarrollados por Rockwell Automation® y ampliamente usados en la industria de modo comercial. Mediante esta integración, el RP5GDL puede ser programado usando GRAFCET (mediante el software RSLogix 5000) para ejecutar tareas como pick-and-place, movimiento lineal hacia un punto determinado, movimiento de paletización o alguna otra variante programable. A continuación, se presenta a modo gráfico la programación propuesta por Leonardelli, 2015 (Figura 2.20).

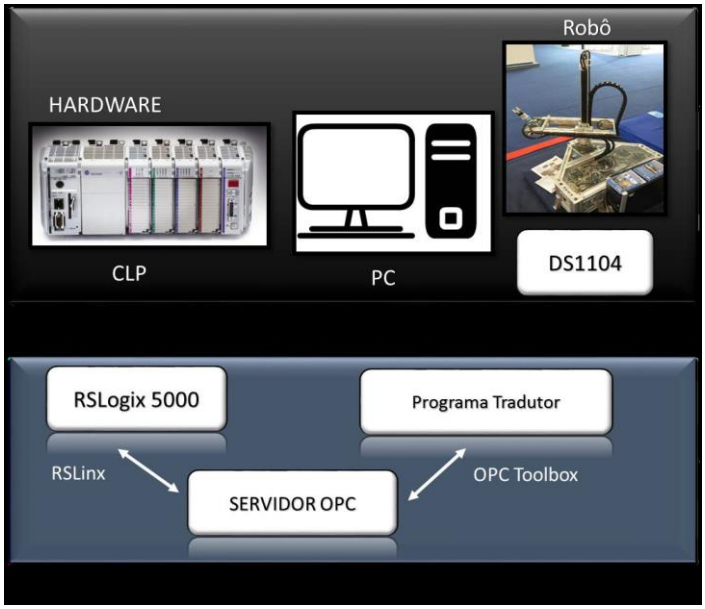

Figura 2.20 - Esquema de programación propuesto por Leonardelli.

2.9.1 Modelo Cinemático directo del robot RP5GDL

El algoritmo sistemático de Denavit-Hartenberg fue aplicado al robot para hallar el modelo cinemático directo que nos permita representar el sistema de coordenadas de cada junta y enlace. Este trabajo propuesto por Missiaggia, 2014 y revisado y adaptado por Sarmanho, 2014, fue la base de sobre la cual se pudo hacer la representación inicial del sistema de coordenadas completo. A continuación, se muestra una imagen descriptiva de las juntas que representan el sistema robótico, Figura 2.21, la Figura fue extraída de Missiaggia, 2014.

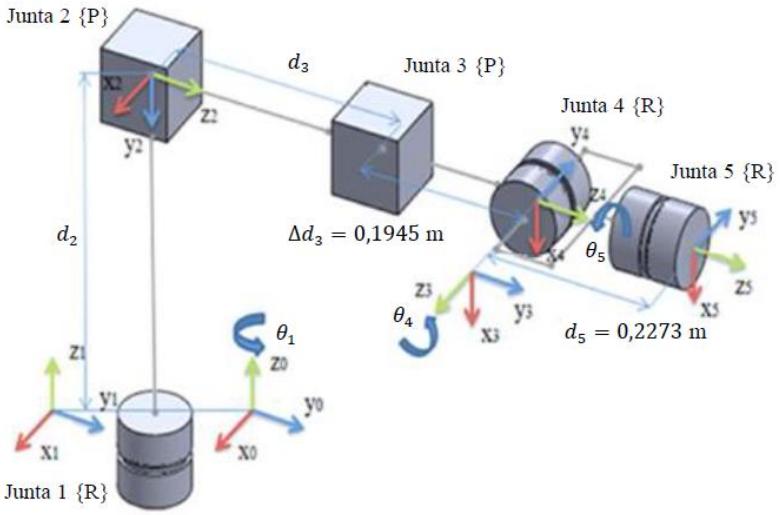

Figura 2.21 - Robot cilíndrico RP5GDL, vista de los ejes de coordenadas de juntas y enlaces.

Según las mediciones hechas y plasmadas en Missiaggia, 2014, se obtiene la tabla 2.1 de parámetros de Denavi-Hartenberg:

Tabla 2.1 - RP5GDL, Parámetros de Denavit-Hartenberg

Eje	α [rad]	a [m]	θ [rad]	d [m]
1	0	0	θ_1	0
2	$-\pi/2$	0	0	d_2
3	$\pi/2$	0	$\pi/2$	0,1945+d_3
4	$-\pi/2$	0	θ_4	0
5	0	0	θ_5	0,2273

Según la tabla y multiplicando matrices de transformación, se presentó también lo que es la ecuación de posición y orientación del efectuador según el sistema de coordenadas del robot presentado en la Figura 2.21 cuyo origen se encuentra en el punto (x_0, y_0, z_0) [Missiaggia, 2014]:

$$\begin{bmatrix} -c\theta_5 s\theta_1 s\theta_4 - c\theta_1 s\theta_5 & s\theta_1 s\theta_4 s\theta_5 - c\theta_1 c\theta_5 & -s\theta_1 c\theta_4 & -0{,}2273\, c\theta_4 s\theta_1 - (d_3 - 0{,}1945) s\theta_1 \\ c\theta_5 c\theta_1 s\theta_4 - s\theta_1 s\theta_5 & -c\theta_1 s\theta_4 s\theta_5 - c\theta_5 s\theta_1 & c\theta_1 c\theta_4 & 0{,}2273\, c\theta_1 c\theta_4 + (d_3 + 0{,}1945) c\theta_1 \\ -c\theta_4 c\theta_5 & c\theta_4 s\theta_5 & s\theta_4 & 0{,}2273\, s\theta_4 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.59)$$

En el momento de planear la manipulación de objetos, también fue importante tomar en cuenta los valores límite de desplazamiento que tienen cada una de las juntas. En el trabajo de Sarmanho, 2014 se hace un estimado según el modelo geométrico del robot. Medina, 2015, por otro lado, añade a esta tabla los valores considerando el amortecimiento de todas las juntas, tabla 2.2. Los valores mencionados se muestran a continuación: [Medina, 2015]

Tabla 2.2 - RP5GDL, valores límite de desplazamiento de las juntas.

Junta	Valores límite
1	-2,236 hasta 2,236 [rad]
2	0,02 hasta 0,430 [m]
3	0,02 hasta 0,280 [m]
4	-1,623 hasta 1,623 [rad]
5	-2,260 hasta 2,260 [rad]

Del resultado de los límites de desplazamiento se obtiene el volumen de trabajo. Para el caso presentado en esta disertación, el volumen de trabajo se limita al frontal, Figura 2.23 y superior Figura 2.22. Ambas figuras fueron extraídas de Sarmanho, 2014:

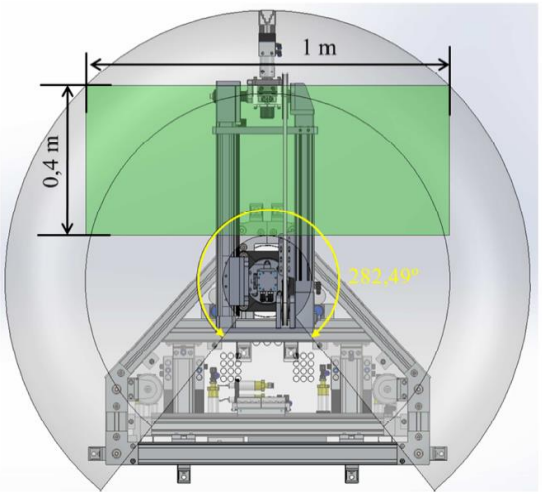

Figura 2.22 - Robot cilíndrico RP5GDL, Vista superior, espacio de trabajo del plano x-y.

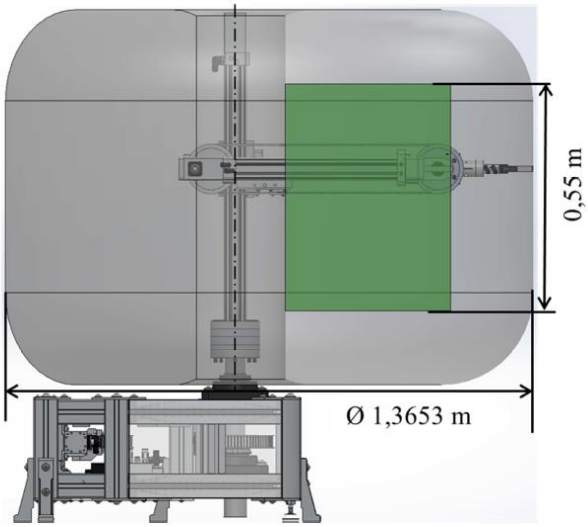

Figura 2.23 - Robot cilíndrico RP5GDL, vista frontal, espacio de trabajo del plano y-z.

2.9.2 Cinemática inversa del robot RP5GDL

En Missiaggia, 2014, se puede ver el trabajo que se hizo para el desarrollo de un método eficiente para la planeación de trayectorias del robot RP5GDL. El texto refiere el uso de polinomios de séptimo grado para la generación de puntos sobre los cuales una trayectoria deberá ser definida. Estas funciones de séptimo grado o *splines* obtenidas por la metodología propuesta por Missiaggia, fue usada para la generación de las trayectorias que minimizan el *jerk*. Se presentan las ecuaciones de cinemática inversa haciendo uso de la matriz de posición y orientación del actuador final o efectuador. Las ecuaciones que dependen geométricamente de los valores de desplazamiento de cada junta son las pertenecientes al GDL 1, 2 y 3, mientras que las del GDL 4 y 5 son independientes del movimiento de las anteriores. Las ecuaciones presentadas a continuación (2.60), (2.61) y (2.62) para los GDL 1,2 y 3, fueron adaptadas para una visualización más sencilla [Missiaggia, 2014]:

$$\theta_1 = \sin^{-1}\frac{x}{\sqrt{x^2+y^2}} \tag{2.60}$$

$$d_2 = z - 0{,}2273 * \sin\theta_4 \tag{2.61}$$

$$d_3 = \frac{y}{\cos\theta_1} - 0{,}1945 - 0{,}2273 * \cos\theta_4 \tag{2.62}$$

Mientras que los GDL 4 y 5 pueden ser provistos en radianes sin ninguna dependencia con las demás variables.

2.10 Estado del Arte

En variados trabajos presentados a lo largo de los años en el área de visión estereoscópica, los más importantes resultados se han mostrado en el ramo de los robots de asistencia. Por ejemplo, en el trabajo de [Natarajan, Durrant et al., 2011], un robot de asistencia es capaz de reconocer objetos para la manipulación en ambientes de mucha textura o con escenas bastante cargadas. Se combina el reconocimiento de patrones mediante color y forma, lo que permite generar una "ventana" de objetos deseados y se evita el cálculo de los mapas de disparidades de toda la escena, así se provee al robot no sólo de las coordenadas para la manipulación, sino de un objeto reconstruido en 3D del que no sólo se tiene la coordenada central en los tres ejes de coordenadas, sino que además las medidas de sus dimensiones que pueden describir su volumen. Sin duda que, para el efector final del robot, este es un dato interesante si se trata de manipular objetos que antes no habían sido cargados en memoria como patrones. También es de resaltar, que se hace uso de regiones segmentadas de modo que la iluminación del medio circundante no afecta en gran medida el cálculo de las disparidades. En este caso, el algoritmo de disparidad es mejorado proporcionándole una mejor robustez al sistema no importando las condiciones de luz variables.

Por otro lado, en el trabajo de [Kanellakis, Kyritsis et al., 2015] se trabaja con manipulación de objetos en movimiento donde el procesamiento de las imágenes se debería hacer en tiempo real y con el menor costo de procesamiento posible. La técnica usada es la de estimación de la posición 3D usando vectores que registran las posiciones en cada toma de la cámara y así pueden predecir el lugar donde se encuentra un punto después de un tiempo transcurrido tras haber observado por primera vez. En este trabajo de bajo costo, se hace uso del software libre OpenCV para las rutinas más comunes de procesamiento de imágenes en visión estereoscópica que son el armado de la disparidad por medio del método SGBM (*Semi Global Block Matching*, por sus siglas en inglés) y el cálculo de momentos para la detección del centro de masa una vez que el objeto pudo ser aislado de la nube de disparidades. Lo más resaltable del trabajo es que los resultados se proponen para brazos robóticos del tipo industriales y que pueden ser obtenidos desde arreglos sencillos con cámaras comunes y procesadores de bajo costo.

[Luu y Tran, 2015] y [Hu, Li et al., 2015] por otro lado, presentan trabajos realizados con la particularidad de que las cámaras en un arreglo estereoscópico se montan sobre un objeto móvil o tienen movilidad por servomotores. A este arreglo se le llama "eye-to-hand" o "Hand-eye", respectivamente, y es usado para que una vez observado un objeto cerca, el robot se mueva hasta quedar justo en frente del objeto y luego de reconocerlo por detección de color procede al cálculo de las coordenadas para la siguiente manipulación. Lo interesante de estos trabajos es que las cámaras poseen movilidad por si mismas emulando la posibilidad de los ojos de un ser vivo de captar el objeto deseado cuando se mueven los globos oculares o el cuello. En el primer caso, el trabajo de Luu y Tran, 2015, el cálculo de la disparidad se ve

seriamente afectado por la iluminación por la que se usaron mínimos cuadrados para mejorar el error en el cálculo de la disparidad. Por otra parte, en el trabajo de Hu, Li et al., 2015, la aplicación es mucho más interesante ya que se trata de una mano biónica controlada por impulsos nerviosos. El sistema estereoscópico fue montado usando cámaras comerciales y la mano biónica es usada para manipular objetos detectados y que podrían inclusive captarse en movimiento. Se presentan algoritmos mejorados de detección de coordenadas de objetos en movimiento y los cálculos matemáticos que describen las trayectorias y velocidades que deben tener cada uno de los dedos de la mano para poder captar un objeto.

En el área de los robots móviles también es muy conocido el uso de visión estereoscópica, tal es el ejemplo de los múltiples trabajos realizados por la compañía ya extinta Willow Garage que llegaron a desarrollar robots móviles de asistencia y que además son bastante conocidos en esta área de visión computacional. En el trabajo referenciado [Rusu, Holzbach et al., 2009], presentan un robot móvil para manipulación de objetos, que hace uso de una densa nube de puntos de profundidades producida por el cálculo de disparidades, la cual les permitió reconstruir objetos en 3D usando filtros de mallas de superficies y la proyección de iluminación texturizada, creando modelos que no sólo describen la posición, sino también la forma, y hasta el modo en el que pueden ser manipulados. Los modelos y la manipulación de los mismos pudieron ser recreados usando el software OpenRave (*Robotics Virtual Enviroment*) que desde 2013 fue lanzado como entorno de simulación de trayectorias de robots para manipulación. En este caso, el robot no depende de un objeto patrón, sino que recrea la escena con la probabilidad de poder manipular cualquier objeto. Está claro que la decisión de qué objeto va a ser manipulado constituye un paso posterior de toma de decisiones según el requerimiento.

Al mismo tiempo otro caso interesante es presentado por [Azad, Asfour y Dillman, 2007], en el que los objetos reconocidos por un robot móvil pueden ser de texturas variables y de formas complejas. Este robot puede detectar objetos que se encontrarían en una cocina común. Se habla del reconocimiento 6D de objetos patrones. En la forma práctica se dice que para cada eje hay dos dimensiones a medirse respecto de todos los puntos del objeto, la traslación y la rotación, de ahí que se maneje el 6D. El trabajo permite reconocer objetos según la textura o por segmentación global de color o de forma. Ya que trabaja con un sistema en tiempo real y en escenas arbitrarias, el robot debe ser capaz de recrear los escenarios 3D de la manera más confiable posible. Los resultados muestran desviaciones menores al 1% en comparación de formas, desde los objetos patrón a los objetos reconocidos como el similar en la escena.

También existen bastantes trabajos realizados usando la llamada visión activa. Esta denominación se refiere a sensores que miden distancias usando métodos que pueden alcanzar el objeto como los sensores de láser o ultrasonido. Las cámaras, por otro lado, constituyen elementos pasivos de sensado. En un trabajo realizado por [Karaoguz, Dankers et al., 2010], se analizan métodos de disparidad y “tamaño cercano o familiar”, usando estadística para determinar el más efectivo. Este trabajo se desarrolla para determinar el mejor método de detección de profundidades para el después permitir al robot una manipulación segura y eficaz. Los resultados muestran que la visión activa es bastante eficaz en distancias de corto y mediano alcance. Se encontró evidencia de que la visión activa en combinación con la visión pasiva estereoscópica provee mejores resultados y que los algoritmos planteados en OpenCV funcionan mejor en el medio y largo rango. Se demostró que hasta el rango largo propuesto (hasta 20 cm) OpenCV muestra buen desempeño en la estimación.

Entre los métodos también existen algunas mejoras planteadas como es el caso del trabajo presentado por [Hirschmüller, 2001]. En este trabajo se intenta resolver el problema que existe en la búsqueda de correspondencia en los bordes en sistemas que trabajan en tiempo real. Se plantea que los puntos de correspondencia en los bordes que, generalmente son desechados por inconsistencias, sean tomados en cuenta y así generar con ellos el verdadero borde buscado. También el uso de filtros para evitar errores generales de iluminación podrá mejorar el procesamiento de la información en tiempo real. Hirschmüller,2001 trabajó tomando en cuenta que, en algunos casos, cuando se hace el cálculo de la disparidad, por problemas de iluminación o ruido, los pixeles pueden estar ocluidos.

Finalmente, se exploró una tesis de postgrado completa presentada por [Medina, 2015]. En este trabajo se integra un robot manipulador robótico cilíndrico, un sistema de visión computacional, de una sola cámara, que identifica las piezas a ser manipuladas (posición y ángulo de giro en relación al sistema de coordenadas cartesianas del robot) informando al algoritmo de planeación de trayectorias del robot los valores de coordenadas necesarias para generar la trayectoria de forma que este pueda atrapar una pieza en una determinada posición y moverla hasta otra posición predeterminada .

Inicialmente, para lograr una homografía confiable, se hizo uso de la calibración de la cámara con un patrón tablero de ajedrez que se montó en el robot. Seguido a esto se logró el reconocimiento de los objetos haciendo cálculos de momentos y comparándolos con los momentos del objeto patrón. En el trabajo no se maneja la información de la forma o las dimensiones del objeto como dato obtenido del reconocimiento, sin embargo, se maneja su orientación. La orientación es obtenida desde la llamada “look-up table”, la cual permite al usuario, según la observación de los momentos calculados del objeto visto, aproximar un ángulo de orientación para la manipulación. El sistema de visión finalmente es acoplado al sistema del robot para la planeación y generación de trayectorias necesarias para la manipulación usando las coordenadas (x, y) halladas.

3. METODOLOGÍA DE RECONOCIMIENTO DE COORDENADAS A PARTIR DE UN SISTEMA DE VISIÓN ESTEREOSCÓPICO Y PLANEACIÓN DE TRAYECTORIAS

3.1 Descripción general, etapas del trabajo

El sistema de visión computacional presentado se propone como medio de adquisición de datos de coordenadas en tres ejes. El propósito de la adquisición de coordenadas se centra en la manipulación de objetos que pueda realizar el robot neumático de control automático para lograr un determinado fin. Generalmente, en la industria, este fin es el de transporte, selección, o acomodo para el despache. Sin embargo, fuera de los posibles usos, la experimentación de las posibilidades de reconocimiento son el tema principal del presente.

Todos los algoritmos implementados en el presente trabajo fueron desarrollados haciendo uso del software libre OpenCV *(Open Source Computer Vision Library*) en versión 3.0 y por medio de la interfaz en C++. Las etapas por las que pasó el trabajo se pueden observar en la Figura 3.1.

La primera parte, la calibración de cada cámara y la estereoscópica, es un proceso importante que requiere de varias muestras de la tabla de ajedrez mencionada en el apartado 2.5.2. En la metodología de implementación según los algoritmos de Zhang, 2000 y Brown, 1971, es posible hallar los valores de distorsión y distancias focales de los ejes x y y (de la cámara), lo cual permitirá relacionar los sistemas de coordenadas de la imagen de la cámara y del mundo real. El último paso de este proceso es el de rectificación, donde los pixeles de las filas de las imágenes derecha e izquierda se alinean para dejar los puntos específicos de observación en la misma fila o en la misma altura descrita por el eje y. En la práctica esto simplifica la parte de la triangulación. Se repite la metodología de calibración hasta que el valor de error medio sea menor a 1 pixel. El software programado cuestiona al usuario si es necesario “recalibrar” el sistema en caso de que el error medio de diferencia de pixeles en ambas imágenes sea mayor a 1 pixel.

La siguiente parte está relacionada a la detección de contornos mediante la aplicación de filtros a la imagen para destacar sus bordes. Un contorno es escogido (cambiará de color a verde) si el *target* y aquel contorno cumplen las condiciones de similitud previstas. Estas condiciones de similitud se establecen mediante el cálculo de momentos de los objetos, si los momentos de ambos contornos tienen valores cercanos y si la cantidad de puntos del borde (descritos por vectores de puntos) es similar a la del *target*, entonces el objeto es reconocido como el buscado. El sistema fue programado de modo que el entorno o fondo de la imagen, aun cuando esté muy cargado, no sea un obstáculo para la detección. Una vez el objeto es reconocido se procede al cálculo de su centro de masa con el uso del cálculo de momentos de segundo orden. Finalmente, en esta parte no se previó el reconocimiento de imágenes obstruidas, ya que la comparación de momentos deja de tener sentido cuando el centro de masa cambia.

En la parte final se halla la tercera coordenada situada en el centro de masa calculado en la parte anterior. Se crea un pequeño cuadro de 40 x 40 pixeles alrededor del centro de masa para estimar la profundidad de ese segmento tomando una media de la disparidad de todos los pixeles dentro del cuadro.

Una vez que ya se tienen las tres coordenadas, éstas son enviadas como datos al software de planeación de trayectorias del robot mediante el uso del protocolo DDE como interfaz de comunicación entre el software de reconocimiento estereoscópico y el software de programación de trayectorias del robot. Esta programación se añadió al sistema de visión como una rutina final con la intención de transmitir datos de manera determinística y lograr que el robot se mantenga en movimiento de manera autónoma cuando los objetos sean reconocidos.

Ya en el Matlab, las trayectorias son originadas tomando el punto inicial *home* del robot y el punto de coordenadas x, y y z hallados y trasmitidos por DDE. Mientras que el software comercial RSLogix5000® Los movimientos son programados usando GRAFCET.

Todos los detalles de esta metodología se describen a continuación a partir del apartado 3.3, mientras que los estadios se pueden observar en la figura 3.1.

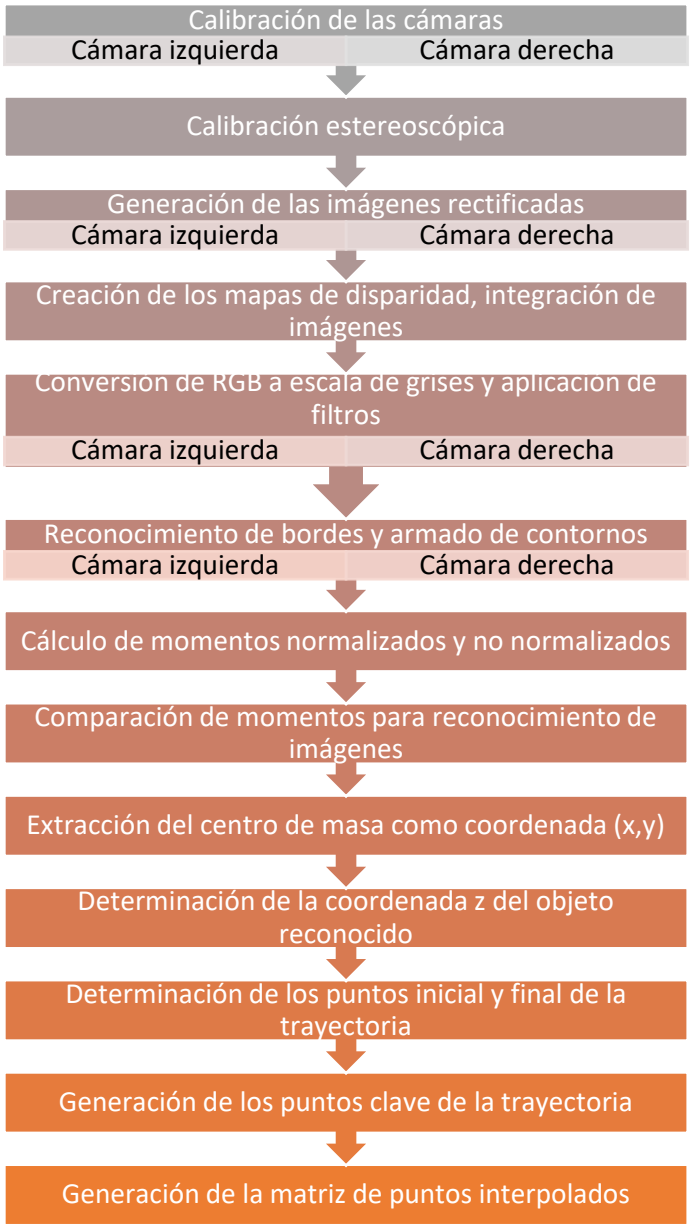

Figura 3.1 - Metodología de implementación del sistema estereoscópico

3.2 Especificaciones técnicas y estructura física de instalación

Para el sistema de visión computacional se usaron dos cámaras IP, sin cables, que comúnmente se usan para aplicaciones de seguridad y que poseen lentes *wide-angle* o apertura panorámica de visión.

El propósito de usar cámaras *inalámbricas* fue el de poder mover este sistema de visión a diferentes vistas del robot, a fin de lograr que los objetos observados queden siempre enfocados en el centro o próximos al centro de la imagen. También se puede mencionar que las cámaras poseen libertad de movimiento en los ejes x y y, lo que facilita la alineación en los ejes para la rectificación. Abajo en la tabla 3.1 se puede observar las características técnicas de las cámaras utilizadas:

Tabla 3.1 - Características técnicas de las cámaras usadas en el sistema estereoscópico

Características de las cámaras	
Marca y modelo	D-link DCS-5222L
Sensores ópticos	1/4” Megapixel progresivo sensor CMOS
Iluminación Infrarroja	Distancia 26 pies, aplicable sólo cuando existe poca iluminación
Otros sensores	Sensor infrarojo (PIR) pasivo, detector de movimiento
Accesorios	Micrófono y parlante
Apertura del lente	F1.9
ConFigurables	Tamaño de imagen, resolución y frame rate, brillo, saturación, contraste, vista espejo, vista invertida
Compresión de Video	JPEG, H.264 a 25 fps
Resoluciones	1280 x 720
	800 x 448
	640 x 360
	480 x 272
	320 x 176
Movimiento horizontal y vertical	Rango: -170° to 170°
Zoom Digital	10x
Interface de Red	IEEE 802.11n
	10/100 Ethernet
Protocolo de Red	IPv4, IPv6, TCP, UDP, ICMP, DHCP Client, NTP Client (D-Link), DNS Client, DDNS Client (D-Link), SMTP Client, FTP Client, HTTP
	ConFiguración accesible via web browser
Alimentación	12 V/1.25 A, 50/60 Hz
Consumo	10,5 watts
Dimensiones	4,46 x 4,46 x 4,92 pulgadas
Peso	0,75 lb (340 gramos)
Temperatura de operación	32 a 104 ºF (0 a 40 ºC)

Un esquema de montaje ejemplo del sistema, se puede observar en la Figura 3.2:

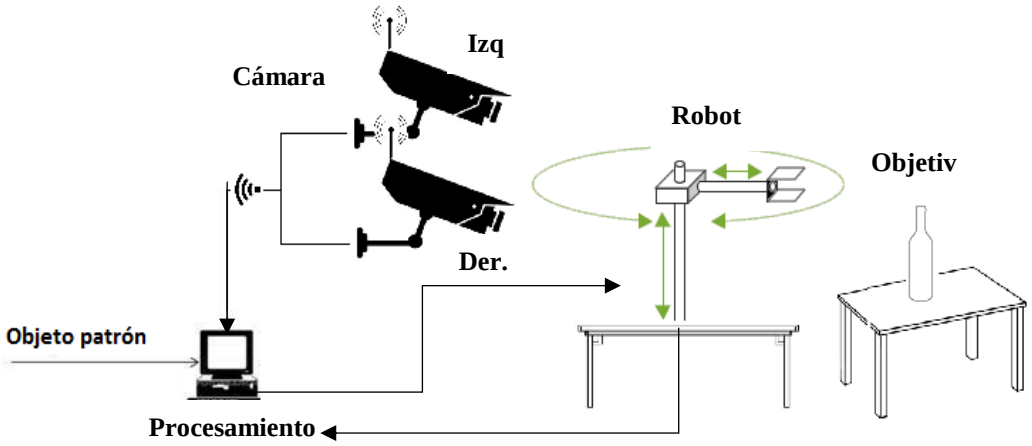

Figura 3.2 - Esquema de montaje del sistema estereoscópico

3.3 Calibración de las cámaras

Como se vio en el capítulo 2, apartado 2.5, la calibración trata de hallar las distorsiones geométricas correspondientes a los parámetros físicos de las cámaras al igual que los parámetros de rotación y traslación. La calibración estereoscópica, de forma más explícita, es el proceso por el cual es posible calcular la relación geométrica de las dos cámaras en el espacio. Esta calibración depende de la matriz de rotación R y el vector de traslación T entre las dos cámaras.

En este trabajo, se hizo uso de las cámaras descritas en el apartado 3.2 y para la rutina de calibración de las mismas, se usó en primera instancia, el objeto "tablero de ajedrez" mostrado en la Figura 3.3:

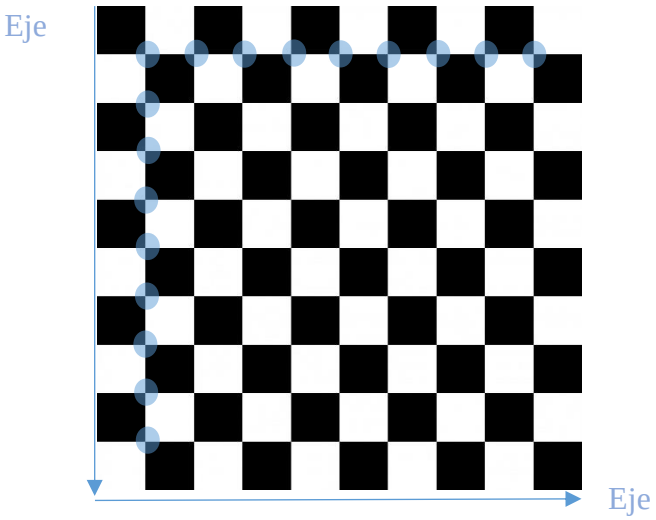

Figura 3.3 - Patrón de objeto "tablero de ajedrez" usado para la calibración de ambas cámaras

Se puede ver que el objeto cuenta con 10x10 cuadros en blanco y negro. En la realidad, se imprimió el objeto con cada cuadro de 38x38mm. Este dato fue importante en la

resolución de las ecuaciones de calibración que, a pesar de tomarse como constante, puso la base en unidades espaciales para calcular el valor de la proyección de un punto en el plano real.

Para el reconocimiento de los "cuadros" blancos y negros, se debe especificar primeramente la cantidad de esquinas de cuadros negros que se tienen en el tablero, siendo la esquina relevante la que se conecta con otro cuadro negro. En el ejemplo de la Figura 3.3, se ven 9 esquinas por lado (esquinas interiores rodeadas por un cuadro negro). Es decir, el algoritmo de reconocimiento de esquinas de OpenCV usa los pixeles conectados entre la esquina de un cuadro negro y otro, detectando el cambio en el gradiente de color de las esquinas y posteriormente, este proceso se afina eligiendo a nivel de pixeles dónde están verdaderamente los puntos. Además, por un tema de simplificación se transformó la imagen "tablero" a una escala de grises para manejar solamente un canal. En la Figura 3.4 se puede ver el resultado de este reconocimiento.

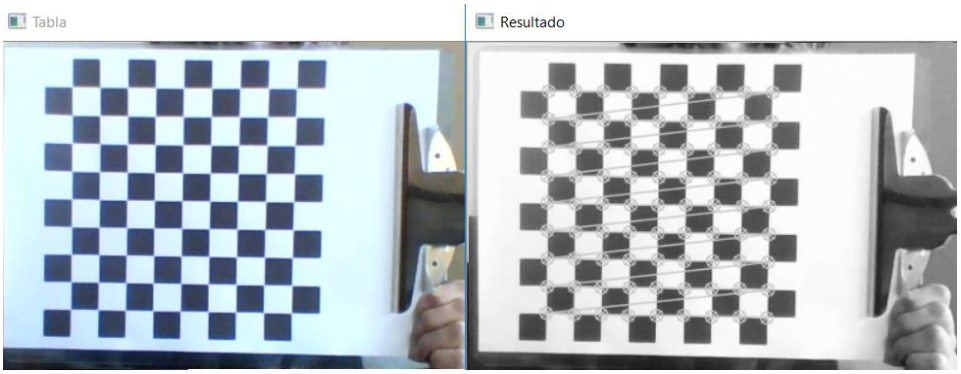

Figura 3.4 - Reconocimiento de esquinas del objeto de calibración usando OpenCV

Se debe mencionar que tal como se plantea en la Figura 3.3, el origen de coordenadas está situado en la izquierda superior del cuadro y es de donde se empiezan a reconocer las esquinas. Las líneas graficadas entre punto y punto denotan el orden en el que fueron tomados todos los puntos.

Como se mencionó en el apartado 2.6.4 acerca de la calibración, el objeto tablero debería tener más de 3x3 cuadros y además hacen falta más de 2 vistas por eje para resolver las 10 incógnitas pertenecientes a los parámetros intrínsecos y extrínsecos de la cámara.

Inicialmente, se plantearon 9 vistas de 9x9 cuadros y se logró obtener mediante la función *CalibrateCamera()*, primeramente, los parámetros de la ecuación M y las distorsiones. En el apartado 2.5, Parámetros extrínsecos e intrínsecos del modelo de cámara Pinhole, se había mencionado la matriz M Ecuación (3.1) y las distorsiones: tres radiales (k_1, k_2, k_3) y dos tangenciales (p_1, p_2).

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

Lo cual permite hallar la Homografía del sistema. Se había mencionado en el apartado 2.6.3, que la homografía H era la combinación de la ecuación de parámetros propios de la cámara M y de las rotaciones y la traslación W, de modo que $H = MW$.

Desde el punto de vista práctico, esta homografía finalmente es la que permite hallar el mapeo del punto del mundo real al del plano de la imagen o viceversa. El último paso de este trabajo es agregar a los valores de las coordenadas de la imagen la corrección de los valores usando las distorsiones.

Se tienen, como se vio, cinco coeficientes de distorsión que se requieren para la corrección de las coordenadas del punto en el plano de la imagen x_v y y_v. OpenCV usa el procedimiento planteado por [Heikkila y Silven, 1997] para coordenadas corregidas en pixels como parámetros intrínsecos de la cámara. En la resolución de los parámetros de la matriz M, f_x y f_y como distancias focales y c_x y c_y como puntos principales (usualmente el centro de la imagen) finalmente se obtienen los valores de coordenadas deseados.

Aun cuando el algoritmo de calibración es bastante consistente, la tarea de hallar la matriz M y las distorsiones depende de un cierto grado de práctica, ya que las vistas en diferentes rotaciones y traslaciones de la tabla de ajedrez deben proveer información relevante para hallar las 10 incógnitas planteadas. Para poder iniciar, los valores de distancia focal f_x y f_y o los puntos principales c_x y c_y deberán iniciarse en ceros. Como se verá en el capítulo de resultados, los valores de estos coeficientes van a diferir (de manera más bien sutil) de intento en intento.

Para poder comprobar que el trabajo fue bien realizado, sin embargo, se hace uso de la homografía y los parámetros de distorsión para corregir la imagen que muestra la cámara. Se puede ver esto en la Figura 3.5:

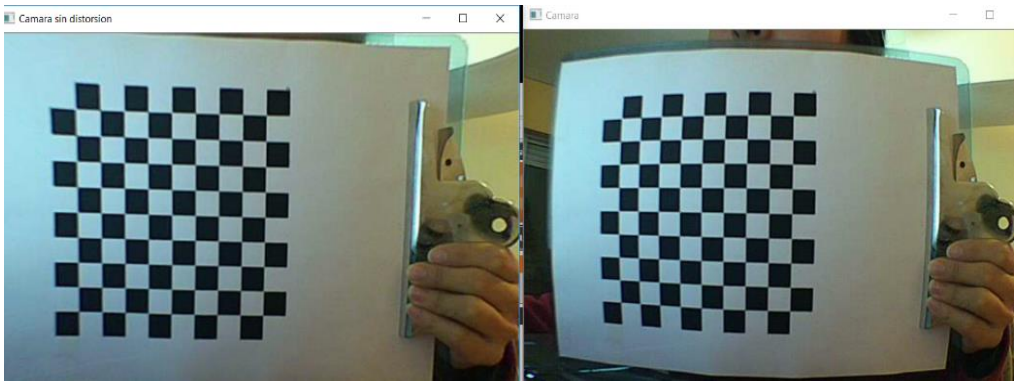

Figura 3.5 - A la derecha la imagen de la cámara sin cambios, a la izquierda la imagen de la cámara sin distorsión

Se puede ver en la imagen de la izquierda que los cuadros están alineados y las esquinas del soporte están perpendiculares. En la imagen sin distorsión se tomaron en cuenta los valores de la matriz M hallada y el valor de las aproximaciones de las distorsiones.

La calibración se debe repetir hasta que el valor del resultado “RMS” o valor de la media cuadrática del error de re-proyección (*Root Mean Square* por sus siglas en inglés) en un caso exitoso esté entre 0,1 y 1,0 pixeles para una buena calibración. Este valor en el modo práctico, es el cálculo de cuán alejado (en pixeles) esté un punto proyectado en la imagen de su posición real.

Finalmente, también se debe tomar en cuenta que, por ser dos cámaras, la matriz W es la resultante de la rotación y la traslación relativas entre ambas cámaras. A este proceso se le

conoce como calibración estereoscópica y es un procedimiento que será necesario abordar para poder llevar este sistema a un entorno consistente. La calibración estereoscópica se detalla en el siguiente apartado.

3.4 Calibración Estereoscópica

El arreglo estereoscópico consta de dos cámaras correctamente alineadas horizontalmente (o verticalmente según el caso), con una separación entre ambas. Esta separación es conocida como el "baseline" o línea base del arreglo y para poder estimar distancias, esta línea base debe ser conocida. En el caso presentado, se montaron las cámaras con una línea base $T_b = 130 \pm 2\, mm$.

En el montaje de este modelo se tuvo que tener muy en cuenta la alineación horizontal de las cámaras ya que por fabricación es posible que, a pesar de ser del mismo lote y serie, la distancia llamada A en la Figura 3.6, no sea la misma de cámara a cámara.

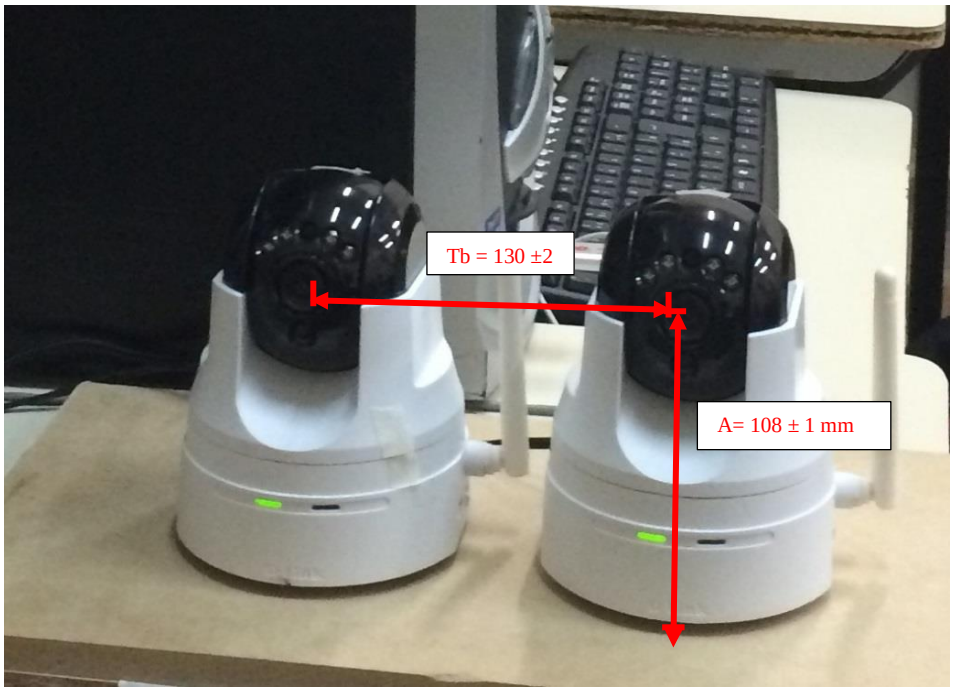

Figura 3.6 - Arreglo de cámaras en el sistema estereoscópico y compensación para alineación en el eje y

Figura 3.7 - Alineación de cámaras según la imagen mostrada

En la Figura 3.7 se puede observar que las cámaras están aproximadamente alineadas. Se observa que la alineación se hizo con un objeto que ocupa el centro de la imagen para evitar el efecto de las distorsiones. Para la alineación se rotaron los lentes con el software embebido de movimiento (cada cámara de seguridad posee la opción de rotación de lentes).

Este paso fue necesario para poder lograr un sistema estable antes de hacer la calibración estereoscópica. Este paso fue fundamental para que el resultado de la calibración sea aceptable y sea posible llegar a valores coherentes de disparidad de la cual se trata en el apartado 3.7.

El algoritmo de Levenberg-Marquardt, mencionado en el apartado 2.5.4, usado en la función de OpenCV (StereoCalibrate), ayuda a encontrar los mínimos locales del error de la re-proyección de las esquinas del tablero de ajedrez en las dos cámaras. Dado que, este algoritmo de estéreo calibración permite hallar tanto la rotación como la traslación entre cámaras izquierda y derecha, la alineación horizontal es necesaria, ya que la traslación sólo se puede calcular en un eje. En particular, el arreglo tomado fue el caso alineado horizontal ya que es más simple de alinear y medir.

Al igual que en el caso de la calibración de cámaras por separado, la función de calibración estereoscópica tiene un valor de media cuadrática del error de re-proyección que se puede utilizar como parámetro para saber si el cálculo de la matriz de rotación y traslación entre cámaras es aceptable. Al igual que en el otro caso, un valor entre 0,1 y 1 de error sería aceptable para una buena calibración. Sin embargo, el error de la calibración previa de cada cámara en particular, incide en este valor por la propagación de errores en el cálculo final.

El resultado de la calibración estereoscópica se detalla en el apartado de implementación experimental, sin embargo, se puede acotar en esta sección que el proceso de calibración requiere del uso de varias imágenes dispuestas en distintas posiciones de traslación y rotación (como se vio antes, por lo menos 8). En específico, por problemas de ruido e iluminación, se requiere de varias imágenes que puedan, en lo posible, cubrir el área útil de captación del lente de la cámara.

Otro punto muy importante es la forma de lectura de puntos de las esquinas del tablero. En el algoritmo de detección de esquinas mencionado en el apartado 3.3, se vio que

las esquinas se leen en el orden correspondiente a la rotación de la imagen del tablero mostrado. Para que la calibración estereoscópica dé valores aceptables, los vectores de puntos cartesianos x y y (en pixeles) que describen la posición de cada esquina de la tabla, deberán ser compatibles entre cámaras. Es decir, si en la cámara de la derecha los puntos se leyeron de izquierda a derecha y de arriba hacia abajo, en la cámara de la izquierda debería haberse dado el mismo caso.

Finalmente, el orden de las cámaras influye en la calibración, desde la vista frontal a la escena, estás son designadas como izquierda y derecha. La cámara "fija" que se usó como eje y desde la cual se miden las rotaciones y traslaciones, es en sistema armado, la cámara derecha.

3.5 Rectificación de imágenes estereoscópicas

Una vez las cámaras por separado se calibraron, el sistema estereoscópico también puede ser calibrado para obtener los pares de puntos epipolares necesarios para el cálculo de disparidad.

La rectificación, como se comentó en el apartado 2.6.5, es el proceso de alineación de pixeles entre la imagen de la izquierda y la derecha con el fin de simplificar el cálculo de la correspondencia entre los puntos de ambas imágenes. Se quiere lograr vistas en las que las filas de las dos cámaras estén alineadas para que la correspondencia estereoscópica sea más confiable y requiera menos procesamiento.

Dadas las matrices de rotación y traslación (R, T) entre dos imágenes estereoscópicas, el algoritmo de Bouguet para la rectificación de las mismas, minimiza la cantidad de cambios en las re-proyecciones de las dos imágenes, de modo que también se minimizan las distorsiones. Este algoritmo también trata de maximizar el área de visión de la imagen.

Los parámetros necesarios para poder implementar el algoritmo son:

1. Matriz intrínseca de la cámara izquierda, M1
2. Matriz intrínseca de la cámara derecha, M2
3. Coeficientes de distorsión de la cámara izquierda, D1
4. Coeficientes de distorsión de la cámara derecha, D2
5. Matriz de rotación entre las cámaras, R
6. Matriz de traslación entre las cámaras, T

Todos estos datos, son valores que se hallaron en los anteriores apartados. Para poder rectificar la imagen se debería lograr que las líneas epipolares sean paralelas, y que los planos de las cámaras sean coplanares rotándolos la mitad del valor de rotación total R de cada lado.

Para una rectificación horizontal, la función usada en OpenCV permite hallar las matrices P1 y P2, que son matrices de proyección 3x4 en el nuevo sistema de coordenadas rectificadas correspondientes a la cámara de la izquierda y la derecha respectivamente. Ecuaciones (3.2) y (3.3):

$$P1 = \begin{bmatrix} f & 0 & cx_1 & 0 \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.2}$$

$$P2 = \begin{bmatrix} f & 0 & cx_2 & T_b * f \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.3}$$

Donde T_b es la línea base horizontal entre las cámaras.

Otro producto del uso del algoritmo de rectificación es la matriz Q de 4x4 que representa el mapa de profundidades, ecuación (2.45).

Un ejemplo del resultado de la rectificación de imágenes estereoscópicas se muestra en la Figura 3.8:

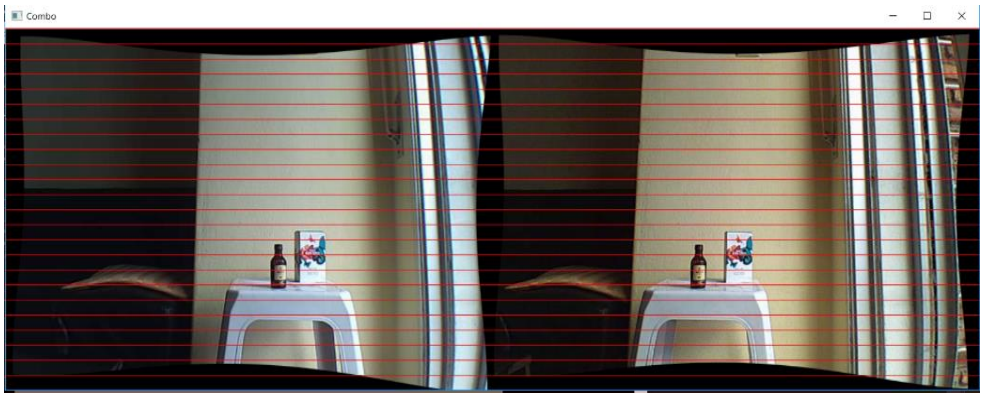

Figura 3.8 - Arreglo de cámaras en el sistema estereoscópico, ejemplo de rectificación

Como se puede apreciar en las imágenes izquierda y derecha, el producto de la rectificación son dos imágenes cortadas de ambos lados cuyos pixeles horizontales están alineados. En lo sucesivo, solamente será necesario usar estas imágenes transformadas para poder calcular las disparidades y en la detección de contornos y reconocimiento.

Se puede observar también que las imágenes no contienen más distorsión exceptuando en los costados o márgenes donde después del cálculo, el error se propaga. Todo el procedimiento de calibración anteriormente citado, deberá ser lo bastante consistente para evitar estas distorsiones.

3.6 Detección de contornos y reconocimiento de imágenes

En el apartado anterior se presentaron la calibración y la rectificación como parte del trabajo inicial. Otra parte importante es la detección de contornos y reconocimiento de imágenes.

Inicialmente, se tomó como base la detección de formas para el reconocimiento de patrones. Esta tarea también se podría haber llevado a cabo por medio de la detección de colores, sin embargo, por razones de eficiencia y flexibilidad se planteó hacer el reconocimiento a partir de las formas de los objetos.

Como se vio en el apartado 2.7, [Canny, 1986], planteó el algoritmo para detección de bordes que es desde entonces el referente en el área. Para una correcta detección de bordes en el programa, la imagen deberá contener la menor cantidad de ruido posible y contar con una

buena iluminación. Además, por la simplicidad, se pasó la imagen a escala de grises para trabajar en un solo canal.

El primer paso para el reconocimiento será entonces la aplicación de un filtro "suavizante" en la imagen de entrada. El filtro mencionado permite que el ruido de la imagen se haga más imperceptible y depende del tamaño del kernel usado. En todos los casos se usaron kernel 3x3 para imágenes normales y 5x5 para imágenes en las cuales no exista muy buena iluminación.

En algunos casos también fue necesario la implementación de transformaciones morfológicas para mejorar el contraste y resaltar los bordes. Se puede observar este resultado en la Figura 3.10, a la izquierda abajo.

A fin de evitar formas geométricas muy definidas se usó el modelo de una botella para el reconocimiento. Usando el algoritmo de reconocimiento de Canny en OpenCV, se pudieron detectar los bordes analizando los gradientes.

En la Figura 3.9 se puede observar a la derecha, la imagen real tomada, en medio los bordes detectados usando Canny, y a la izquierda, los bordes detectados, pero con la opción de relleno que permite ver más claramente la forma que se encierra con los bordes.

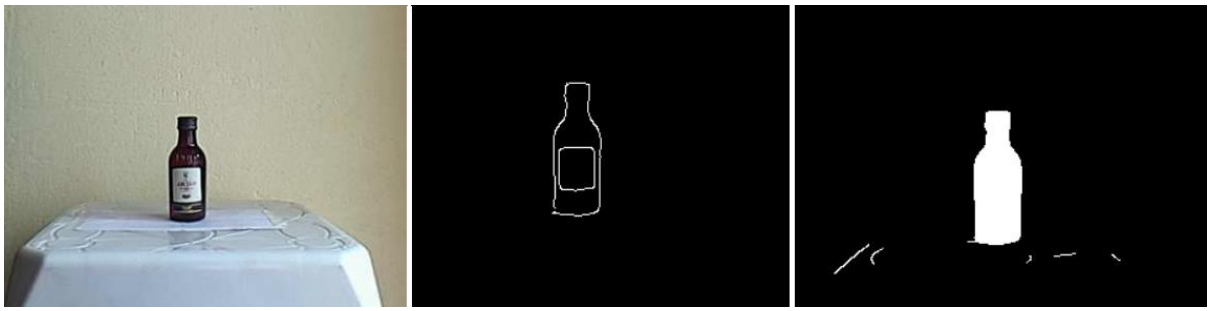

Figura 3.9 - Detección de bordes de las formas propuestas, usando el algoritmo planteado por Canny

Una vez el trabajo se mostró aceptable, se usó el algoritmo de captación de contornos en OpenCV. Los bordes resultan del análisis de gradientes en la imagen, son pixeles desconectados unos de otros que aparecen en la imagen casi como usando un filtro. Sin embargo, se hace necesario guardar en bases de datos estos valores con un cierto orden y jerarquía para determinar un contorno.

Los contornos almacenados en vectores, permiten formar la "imagen" del objeto deseado, y eventualmente también del entorno. Por tanto, para reconocer qué parte de la imagen corresponde al objeto deseado y qué parte no, es que se usa el cálculo de momentos. Como se puede intuir, es necesario primero, guardar en una base de datos o un archivo, el valor de los puntos del contorno "patrón" o deseado para después poder utilizarlo.

Como se vio en el apartado 2.8, los momentos se desarrollan a partir del análisis de imágenes como Figuras planas en dos dimensiones. La base matemática de los momentos invariantes en Figuras planas fue planteada por [Hu, 1962]. Estas funciones resumen varios aspectos de la forma de los objetos y su tamaño, en mecánica serían análogas a los momentos de inercia de los cuerpos.

Se define el momento de orden $(p+q)$ de un contorno de tal forma que los momentos calculados son (Apartado 2.8): Ecuación (3.4)

$$m_{pq} = \sum_{i=1}^{n} \sigma(x,y) x^p y^q \tag{3.4}$$

En el programa desarrollado en OpenCV se hizo uso de las funciones *moments()* y *matchshapes()*, para cálculo de momentos y comparación de los mismos. Fundamentalmente la comparación, que es la función más útil para el reconocimiento, fue tomada en cuenta a manera de cálculo usando: Ecuación (3.5)

$$I_3(A,B) = \max_{i=1\ldots7} \frac{|m_i^A - m_i^B|}{|m_i^A|} \tag{3.5}$$

Donde:

$$m_i^A = sign(h_i^A) \cdot log(h_i^A)$$

$$m_i^B = sign(h_i^B) \cdot log(h_i^B)$$

Además, h_i^A y h_i^B son los momentos de Hu, invariantes, del contorno A y el B. Este cálculo permite hacer una comparación un poco más exacta de los momentos de los contornos.

Usando los momentos de primer orden, se encontraron los centros de masa, donde las coordenadas están dadas por los valores $\bar{x}$ y $\bar{y}$. El cálculo de momentos respecto al centro de masa del objeto permite obtener valores invariantes respecto a la traslación de los objetos. Ecuación (3.6) y ecuación (3.7)

$$\bar{x} = m_{10}/m_{00} \tag{3.6}$$

$$\bar{y} = m_{01}/m_{00} \tag{3.7}$$

En el capítulo 4, Implementación experimental, se hizo uso de estos valores (transformados usando la homografía) en coordenadas del mundo real, permitiendo saber inicialmente, las coordenadas x y y necesarias para definir el punto de fin de movimiento del robot.

Finalmente, para el cálculo de la orientación del objeto, se usaron los valores de momentos de segundo orden como describe la ecuación (3.8).

$$\tan 2\theta = \frac{\mu_{11}}{\mu_{20} - \mu_{02}} \tag{3.8}$$

En otros trabajos, la comparación de momentos para hallar la orientación se llevó a cabo por la llamada “look up table” [Medina, 2015] y [Grassi, 2005]. Sin embargo, este método se tomó de ejemplo, más no fue implementado en esta oportunidad.

Las pruebas de la metodología planteada, se puede ver en la Figura 3.10.

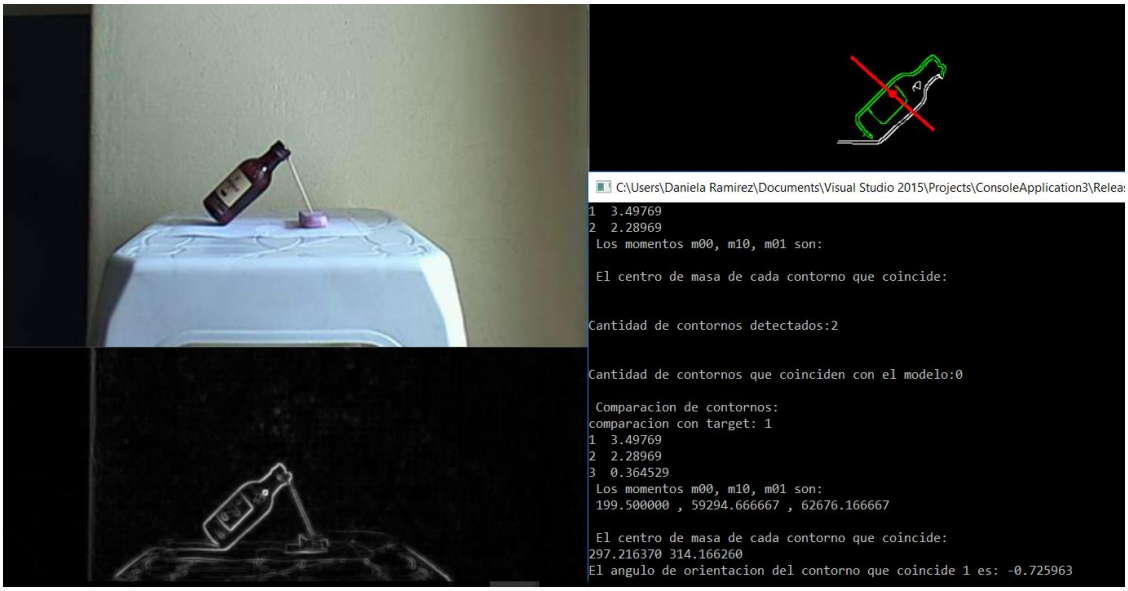

Figura 3.10 - Reconocimiento de contornos, centro de masa y orientación de un objeto.

En la Figura se puede observar a la izquierda arriba, el objeto patrón a ser reconocido. A la izquierda abajo se puede ver la imagen en escala de grises y con una transformación morfológica de gradiente, la cual exagera la pigmentación de los bordes.

Arriba a la derecha el contorno detectado en color verde. La respuesta de la función de comparación usando la ecuación (3.5), permitió al programa obtener números entre 0 y 100 que describen el porcentaje de no coincidencia. Por ejemplo, un valor cercano al 0, mostraría que los contornos comparados son exactamente iguales. Se puede ver que por temas de iluminación sólo la mitad del contorno es aceptado como “comparablemente igual”.

Por último, en la misma Figura de los contornos detectados, se puede ver un punto rojo y una línea que describen respectivamente el centro de masa y el ángulo de rotación del objeto medido desde el eje x positivo.

Abajo a la derecha también se pueden ver los valores de los resultados, todos los cuales se discutirán en el capítulo 5.

3.7 Correspondencia estereoscópica y disparidades

Como se vio en el apartado 2.6.6, La correspondencia en sistemas estereoscópicos se trata de la formación de pares de puntos que en dos vistas de diferentes cámaras resultan ser el mismo.

En OpenCV existen varias opciones de algoritmos para poder implementar las llamadas correspondencias estereoscópicas. En particular, se usó la función StereoSGBM(), porque implementa un algoritmo más robusto y versátil. Para poder llevar a cabo este cálculo, en la práctica se debe contar primeramente con las cámaras calibradas y las imágenes rectificadas además sin distorsión.

Los parámetros de entrada necesarios para el cálculo son:

1. Disparidad mínima: que es el valor mínimo posible de disparidad entre dos pixeles que corresponden al mismo punto de cada imagen. En el caso ideal este valor podría ser 0, sin embargo, las rutinas anteriores de rectificación pueden mover los pixeles a fin de alinearlos. Lo mejor en la práctica sería poder ajustar este parámetro en vivo según las profundidades ya conocidas.
2. Número de disparidades: Es la diferencia entre la disparidad máxima y la mínima.
3. Tamaño de la ventana de búsqueda: la correspondencia es calculada por una ventana que se deberá definir antes de empezar el algoritmo. Para cada característica en la imagen de la izquierda, debe corresponder alguna de la derecha en la misma fila. Después de la rectificación, cada fila es una línea epipolar, así que es la misma en ambas imágenes. Al mismo tiempo ambos puntos deberán tener suficiente textura para poder ser detectados y por supuesto no estar ocluidos.
4. Valores para controlar el suavizado de la disparidad, a manera de filtros.

Este que sería la vena central del cálculo de profundidades, es un paso fundamental y no tan sencillo. Se requiere de bastante memoria de procesamiento cuando se trata de imágenes con resoluciones mayores a los 640x480 pixeles. También se ve afectado por la calibración inicial y por los valores aproximados de distorsión que pueden ser fácilmente afectados por la iluminación.

Hay que tomar en cuenta también que el algoritmo no es tan sencillo de implementar y que deberá hacerse bastante trabajo en el proceso a modo de prueba y error. Sin embargo, una vez que el mapa de disparidades se muestra estable (con pocos pixeles negros que muestran la imposibilidad del cálculo de disparidad) es confiable para el cálculo de profundidad.

En la Figura 3.11 se puede observar el mapa de disparidad hallado usando dos imágenes, una izquierda y otra derecha, de un grupo de objetos. Las imágenes ya fueron rectificadas usando también previamente la calibración estereoscópica de ambas cámaras.

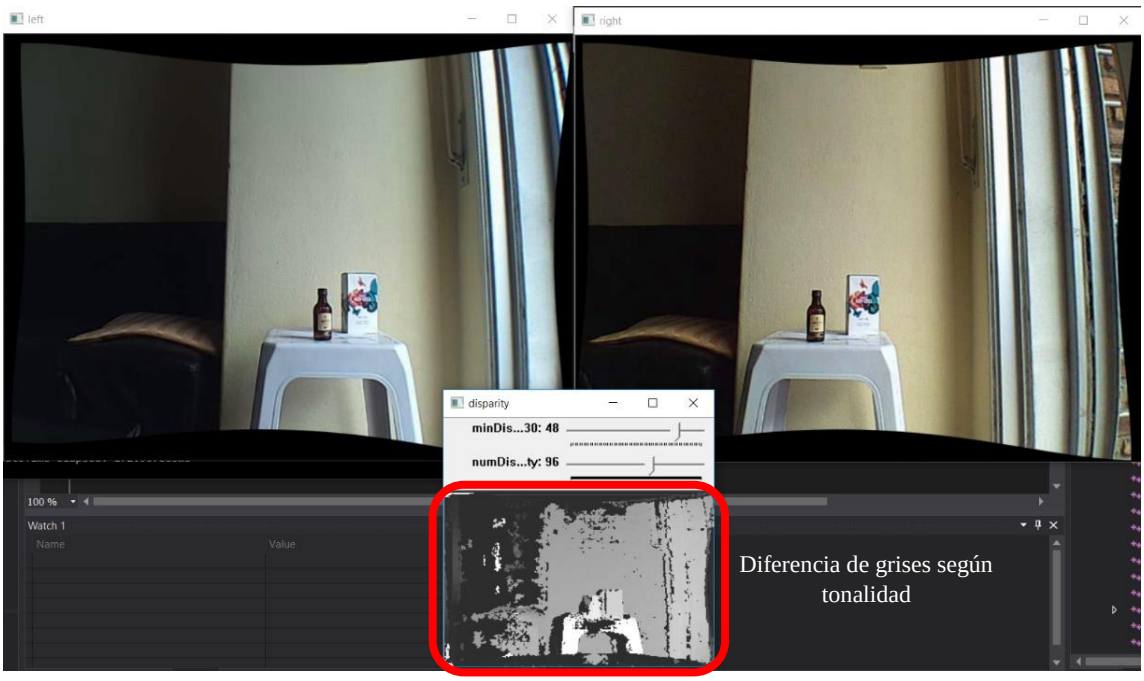

Figura 3.11 - Mapa de disparidades de dos imágenes estereoscópicas

En la Figura se pueden observar diferentes colores en grises. Según el resultado del algoritmo de correspondencia, los grises menos saturados o más claros, muestran zonas más cercanas al punto de observación, mientras que los grises más saturados muestran puntos más alejados. Los pixeles que se observan negros son valores para los cuales no se encontraron valores de disparidad aceptables.

3.8 Determinación de la trayectoria del robot

Una vez se tienen las coordenadas de posición del objeto, el sistema pasa a determinar la trayectoria del robot mediante la transmisión de las coordenadas del punto final y del punto de inicio. Para el manejo de coordenadas en tiempo real, el control del robot como se dijo antes, se maneja desde la placa dSpace a través de la conexión entre Matlab y el software ControlDesk®. Este último software, propietario de la compañía dSpace, se usa para monitoreo e intercambio de datos de comando con la placa dSpace DS-1104. Leonardelli, 2015, desarrolló el *programa traductor* que es utilizado para comunicar el control del robot con los comandos de movimiento. El programa corre continuamente actualizando la posición del robot a lo largo de la trayectoria y escribiendo comandos para el movimiento deseado. Esta ejecución continua se lleva a cabo en Matlab y requiere de algunas entradas de comando para su funcionamiento.

Dígase previamente que los puntos de trayectoria son generados usando el polinomio de séptimo grado de libertad propuesto por Missiaggia, 2014, compilando el control desarrollado en Simulink por Sarmanho, 2014.

Como parte del trabajo de programación en línea, por otra parte, Leonardelli, 2015, logra establecer una conexión entre el sistema de control y la entrada de datos de posición en tiempo real por medio de programación GRAFCET que se implementa desde RSLogix5000 haciendo uso del OPC propietario de Rockwell Automation®, Rslinx®.

La programación GRAFCET permite colocar diferentes bloques de movimiento en la secuencia, de modo tal que, para un determinado movimiento, la planeación de trayectoria se pueda dar de manera mucho más sencilla. Existen por defecto, sin embargo, tres tipos de movimiento estándar programados [Leonardelli, 2015]: Movimiento lineal (entre dos puntos), *pick and place*, y paletización.

En el presente trabajo, se pensó desde un inicio en el uso del bloque de movimiento lineal, ya que, por ser el bloque más simple, con él se puede demostrar la efectividad del sistema al lograr que la garra del robot atrape el objeto propuesto. Sin embargo, según se ve en los siguientes capítulos de implementación y resultados, se tuvieron que hacer algunas otras adecuaciones para un movimiento sin colisiones.

4. IMPLEMENTACIÓN EXPERIMENTAL DEL SISTEMA

En este capítulo son presentados los aspectos más relevantes de la implementación experimental del sistema de visión computacional y las trayectorias generadas en el robot neumático RP5GDL, ubicado en el laboratorio de mecánica y control (LAMECC) de la Universidade Federal do Rio Grande do Sul (UFRGS).

El sistema tiene principalmente dos partes, una la parte del robot neumático desarrollado en el laboratorio LAMECC y la otra la parte del sistema de visión computacional conformado por las dos cámaras IP inalámbricas que se montaron en un punto de observación frontal a los objetos de los cuales se obtienen las coordenadas del punto final de movimiento del robot.

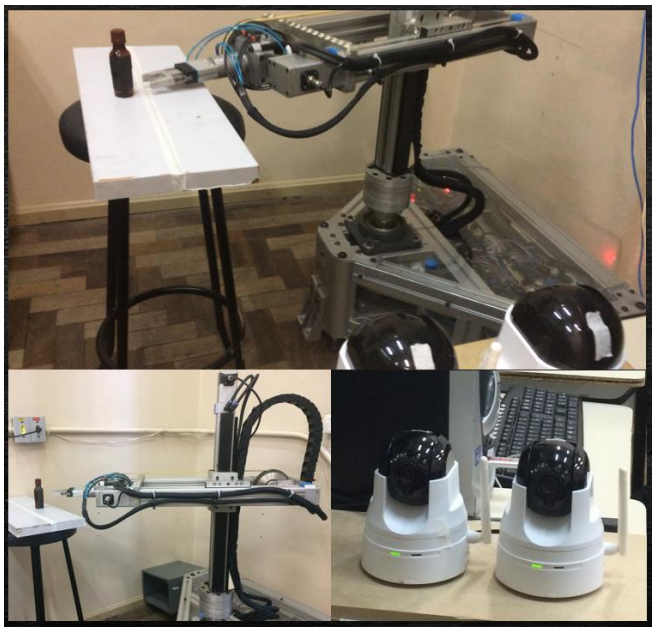

Figura 4.1 - Arreglo de cámaras montado en el laboratorio LAMECC para pruebas del sistema

La primera parte, el robot neumático, posee las siguientes características importantes y a tomar en cuenta en el momento de la implementación:

1. 5 grados de libertad con actuadores neumáticos. En el trabajo de Sarmanho, 2014 los resultados muestran diferentes valores de precisión de posicionamiento según juntas. En el caso del presente trabajo se enfoca principalmente en el error de posicionamiento del 3er GDL, ya que el 4to y 5to no afectan de manera relevante en la manipulación de los objetos ya que estas juntas se usan sólo para atrapar el objeto en alguna orientación aproximada.
   Según los resultados observados en el trabajo de Sarmanho, 2014, se puede evidenciar un error de entre +0,01m y -0,02m (+10 mm a -20 mm), dependiendo del tipo de recorrido de la junta. Sarmanho, después haciendo un análisis de los resultados, propone que la diferencia podría llegar a los ± 0,0012 m o ±1,2 mm. Necesarios claramente para la manipulación de objetos en el ambiente industrial.
2. La garra de sujeción tiene una abertura máxima de 20 mm según las mediciones hechas en el laboratorio.

3. Posee una diferencia sistemática entre el valor deseado y el valor real de posición de:
    a. $DifS_{GDL1} = 0{,}15\, rad$
    b. $DifS_{GDL2} = 0{,}011\, m$
    c. $DifS_{GDL3} = 0{,}006\, m$
4. Donde DifS: Se refiere a la diferencia sistemática y DGL*X*: se refiere al grado de libertad. Estos valores promedio fueron observados en diferentes experimentaciones de movimiento haciendo uso del software ControlDesk y algunas pantallas programadas para monitoreo de valores de posición. Todos los movimientos del robot se hicieron de modo manual.
5. La masa a ser levantada no debería superar los 590 gramos para evitar fallos en el control del robot. En los casos en los que el robot debe manipular más carga, los errores de posición se incrementan [Sarmanho, 2014], por tanto, para llegar a una manipulación segura el objeto debería tener una masa inferior a este margen. Este valor de masa fue integrado en el modelado de control ya hecho por Sarmanho, 2014. Por lo que el valor máximo fue tomado de este estudio anterior más no ensayado en el laboratorio.
6. La presión mínima de aire de alimentación al sistema debería de ser de 6.5 bares, dado que el sistema de compensación de fricción está calibrado con ese valor mínimo. Esta alimentación, sin embargo, no debería sobrepasar los 7.5 bares para evitar que los sensores de presión muestren valores erróneos de medición. [Sarmanho, 2014]. En laboratorio, usando el software ControlDesk se comprobó que los sensores de presión marcaban variaciones en la medición si el valor de presión no estaba en el rango mencionado, es decir, perdían sensibilidad y la respuesta en voltaje al sistema se tornaba constante entre 0 y 5 Volts. Más importante aún, la cantidad de presión mínima es indispensable para el correcto control de las válvulas y las respuestas de los actuadores.
7. El robot debe encontrarse comunicando cada enlace con la placa dSpace. En este caso todos los leds de comunicación deberían indicar un estado favorable. Sarmanho, 2014 diseñó las placas de control de los diferentes enlaces de modo que cada GDL tenga su placa correspondiente. Cuando un paquete enviado falla en la entrega o en la recepción, el led indicador de la placa se apaga. El protocolo manejado de comunicación entre las placas y la tarjeta dSpace usa el estándar RS485.

La segunda parte que corresponde al sistema de visión computacional, es un sistema nuevo implementado a manera de experimentación y que tiene las siguientes características:

1. Se trata de dos cámaras inalámbricas IP con direcciones de red estáticas (ver apartado 3.2), que se conectan con la computadora procesadora de imágenes a través de una red wifi de protocolo 802.11n en la frecuencia inalámbrica más común que es la 2.4 GHz. La velocidad de transmisión de datos es de 130 Mpbs. El sistema de visión se pensó inicialmente para proveer datos de posición en tiempo real al sistema de manera versátil, lo que requiere el uso de video. Sin embargo, en experimentaciones de reconocimiento, se lograron los siguientes resultados de procesamiento:
    a. Hasta 700 ms de procesamiento para reconocer la profundidad de un punto. Tomando en cuenta que el video tiene 25 *frames* por segundo, no es posible

hacer la implementación en modo video (tiempo real). Será necesario el uso de fotografías.

2. El software donde se programaron todas las rutinas de procesamiento es OpenCV versión 3.0. Este software de uso libre no necesita licencia y fue usado con la API (por sus siglas en inglés para Application programming interface) desarrollada en C++, usando como entorno de desarrollo el Visual Studio 2015 con licencia gratuita "community 2015". Se intentó mantener la filosofía de "programación libre" desde la fabricación del robot RP5GDL, ya que, también fue programado con Matlab para garantizar la libertad de codificación y el acceso de usuarios para el testeo. Para hacer más simple la interacción, se usó C++ por la compatibilidad de programación con la plataforma de DSpace y Matlab.
3. El sistema físico de montaje de las cámaras fue hecho a medida según las dimensiones planteadas en el apartado 3.2. Estas cámaras fueron montadas en el laboratorio en una mesa soporte que está orientada de modo que los objetos siempre quedan localizados en el plano frontal de las cámaras.
4. La iluminación del sistema es la misma que se usa en el laboratorio, no se usó ningún otro sistema de iluminación ya que, para el caso, los algoritmos se implementaron con diferentes filtros que mejoran el contraste y además porque las cámaras también tienen opción de visión nocturna. En la visión nocturna las cámaras mandan imágenes en escala de grises, es decir, imágenes de un solo canal.
5. Las cámaras son originalmente cámaras de seguridad. Estas cámaras tienen lentes de fabricación menos comercial que normalmente manejan el "wide-angle", es decir, las vistas que proporcionan son vistas de ángulos más amplios que en algunos casos se pueden configurar hasta ser panorámicos. Sin embargo, esta característica no fue útil al momento de la calibración ya que los lentes agregan una mayor distorsión a las imágenes. Esto se tratará más adelante en este mismo capítulo.
6. Las cámaras proporcionan imágenes en diferentes resoluciones y formatos, sin embargo, se eligió el formato JPEG de resolución 640x480 para evitar sobrecargar los algoritmos con imágenes de más resolución, ya que la presentada en un solo canal presenta ya una matriz de 307 200 puntos a tratar. Por otro lado, el formato JPEG presenta una compresión de 10:1 lo cual permite imágenes claras de calidad estándar que no ocupan demasiada memoria.

A continuación, en los apartados siguientes, se tratará primeramente el sistema de visión estereoscópico, seguido de la generación de trayectorias en el sistema del robot y finalmente se tratarán las pruebas experimentales resultantes de la combinación de ambos sistemas para la manipulación de objetos deseados.

4.1 Sistema de visión estereoscópico

Inicialmente la implementación del sistema empieza con la calibración de las cámaras. Se requieren los valores de la matriz de valores intrínsecos y las distorsiones de cada una de las cámaras por separado. Esto con el objetivo de encontrar la relación entre los puntos $P_w(x, y)$ del plano del mundo real en milímetros y $P_c(x_c, y_c)$ del plano de la cámara en pixeles. Se inicia el software de reconocimiento y se presenta la siguiente pantalla, Figura 4.2:

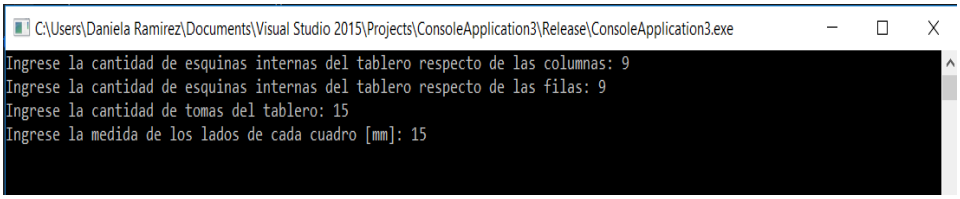

Figura 4.2 - Programa de calibración para cámaras izquierda y derecha

Se ingresan los valores de las esquinas del tablero de ajedrez que se va a usar para la calibración, la cantidad de tomas de imágenes que se van a hacer en tiempo real y la medida de cada cuadro del tablero en milímetros.

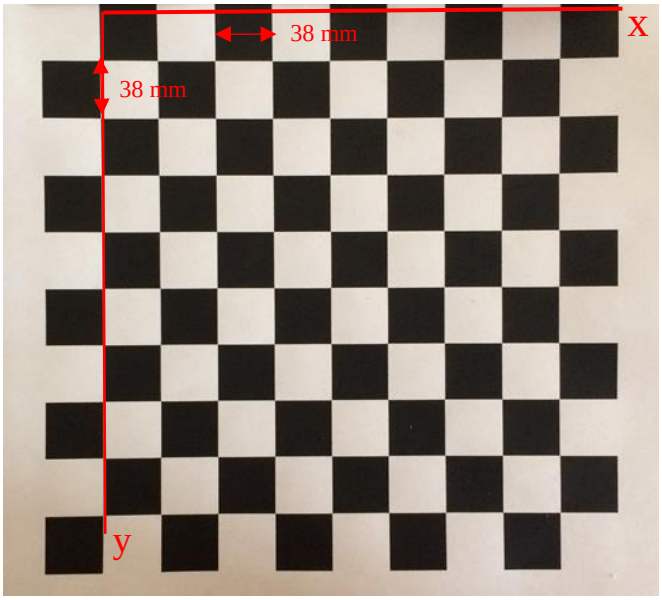

Figura 4.3 - Programa de calibración para cámaras izquierda y derecha

El tablero de la Figura 4.3 fue el que se usó como patrón para hallar las matrices de parámetros intrínsecos de las cámaras y los parámetros de distorsión. En la Figura 4.4 se puede observar el orden de los puntos leídos por el algoritmo de detección.

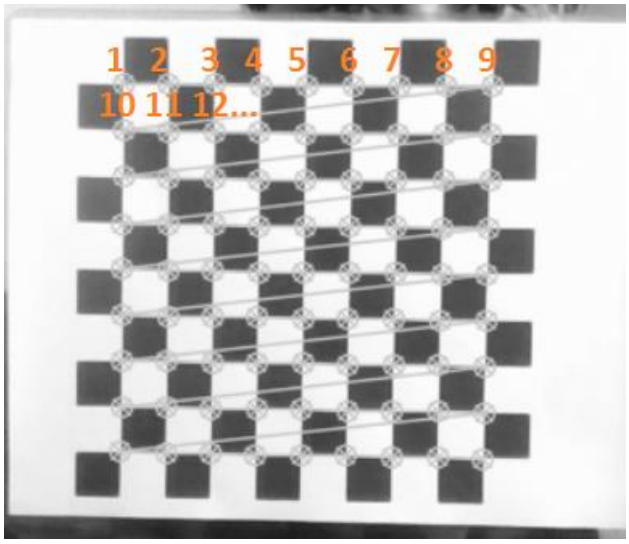

Figura 4.4 - Orden de puntos leídos para la calibración

Para el tablero usado se forma un vector con 81 pares de coordenadas en pixeles que guardan la información de todas las esquinas interiores encontradas en el tablero. El orden que se aprecia en la Figura 4.4, debe ser la misma para ambas cámaras ya que en la calibración estereoscópica se saca una media de los puntos correspondientes a cada esquina y se logra calcular la correcta rotación y traslación entre las cámaras.

Una vez captadas las imágenes para la calibración, el algoritmo muestra la media cuadrática del error calculado en el proceso. Este valor deberá estar entre 0,1 y 1 pixeles para una buena calibración. En la experimentación se determinó que hacen falta al menos 15 imágenes de muestra de los tableros por cámara para lograr una calibración que oscile entre los 0,3 y 0,9 de error.

Si el valor de RMS es mayor a 1, el algoritmo pregunta si se desea realizar una nueva calibración que toma como base los valores anteriores de la matriz intrínseca para lograr un mejor resultado. Véase el mensaje en el último renglón de la pantalla impresa en la Figura 4.5.

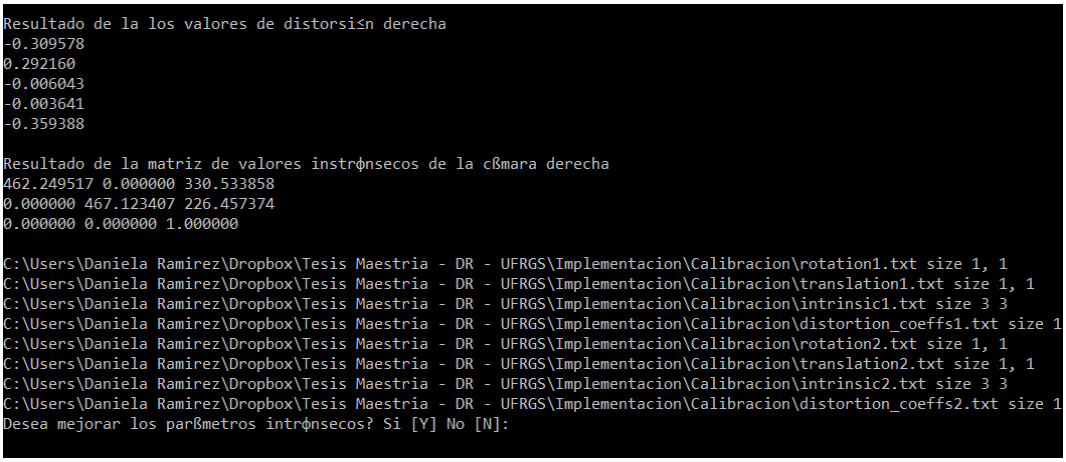

Figura 4.5 - Cálculos de los valores intrínsecos de las cámaras, algoritmo de mejora que repite la calibración

Finalmente, los valores de la calibración quedan guardados en formato .txt (Figura 4.5) ya que, en sucesivos tratamientos de datos, estos valores serán consultados una y otra vez al usar las imágenes de las cámaras sin distorsión y con valores de homografía válidos.

En la siguiente imagen, Figura 4.6, se puede ver el resultado de la calibración de cada cámara. Se observan arriba las imágenes con distorsión y abajo las imágenes sin distorsión.

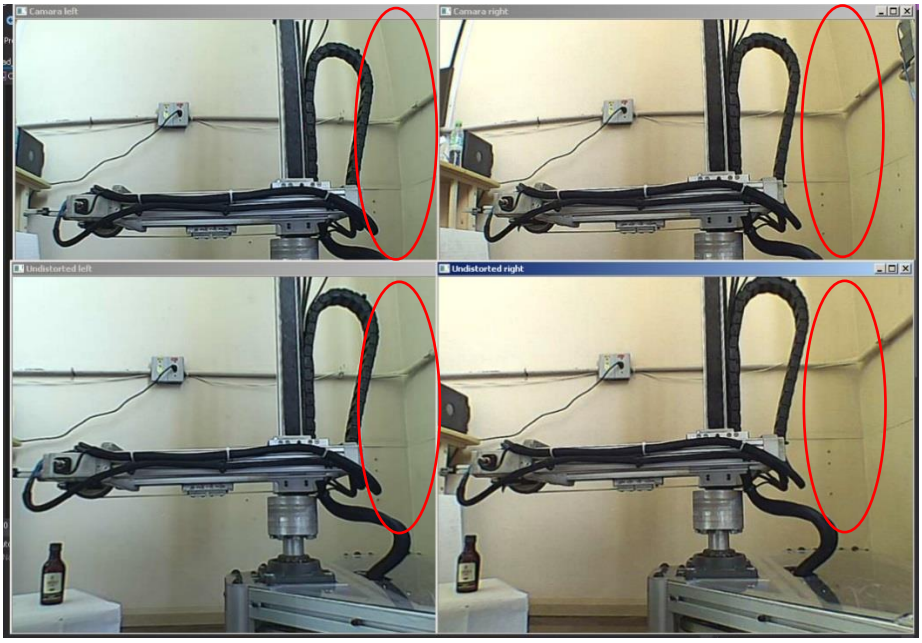

Figura 4.6 - Comparación de imágenes con y sin distorsión

Después de lograr la calibración de cada cámara en particular, se hace la calibración estereoscópica usando los datos antes obtenidos. El programa hace el cálculo automáticamente sin participación del usuario.

El arreglo de las cámaras es el que se puede ver en la Figura 4.7. Como se dijo en la sección 3.3, la separación entre las cámaras es de 130 ± 2 mm. La cámara fija es la cámara izquierda, es desde este punto que se mide la traslación y el ángulo de rotación.

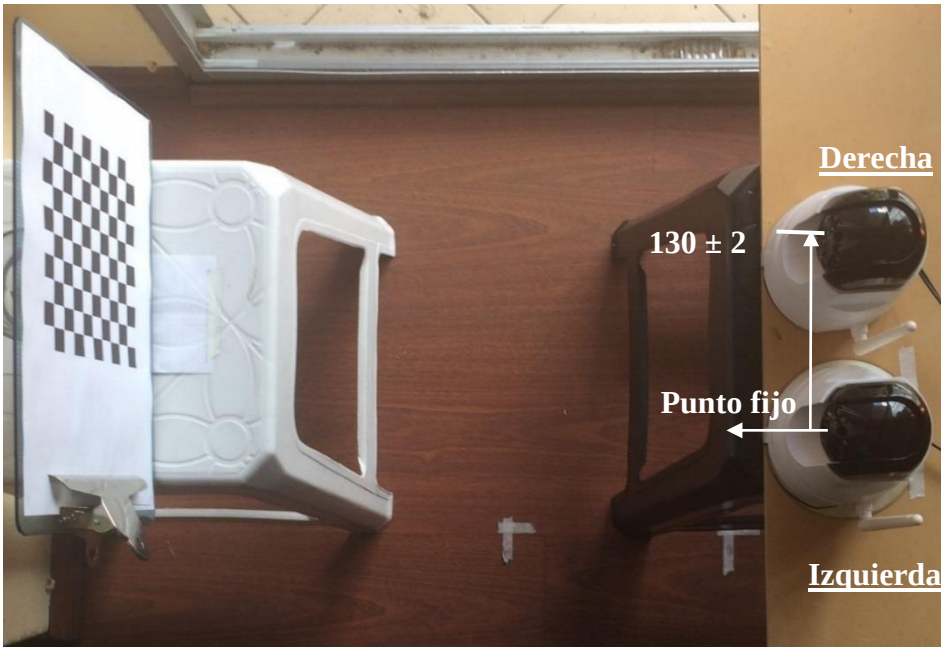

Figura 4.7 - Arreglo estereoscópicos de cámaras, montaje planteado como banco de pruebas

El resultado de la calibración proporciona valores de la matriz de rotación R y la matriz de traslación T, medidas respecto del punto de referencia fijo que está en la cámara izquierda. Al mismo tiempo también son calculadas la matriz fundamental F y la esencial E.

En el apartado 2.6.3 se había mencionado que la matriz fundamental F relaciona las dos cámaras en coordenadas de pixel y la matriz esencial E contiene toda la información acerca de la geometría de las dos cámaras relativa una a la otra.

En la implementación, la calibración del sistema estereoscópico mostró valores que oscilaban con RMS de entre 1 y 20. En la teoría este valor debería oscilar entre 0,1 y 1 como valor máximo. Sin embargo, el valor mínimo logrado de 0,93 cuenta con la consistencia suficiente para poder emparejar los pixeles de lado a lado haciendo las líneas epipolares paralelas.

En la Figura 4.8 se puede observar el resultado de la calibración cuando el algoritmo muestra valores cercanos a 1. A la izquierda de la Figura 4.8 se puede observar la cámara izquierda con la imagen rectificada y a la derecha la imagen rectificada de la cámara derecha. Puede observarse también los espacios negros que rodean las imágenes resultado de la rectificación estereoscópica para mejorar la correspondencia entre los pixeles de las filas. En esos espacios oscuros no fue posible encontrar una correspondencia lógica por lo que los pixeles quedan en color negro.

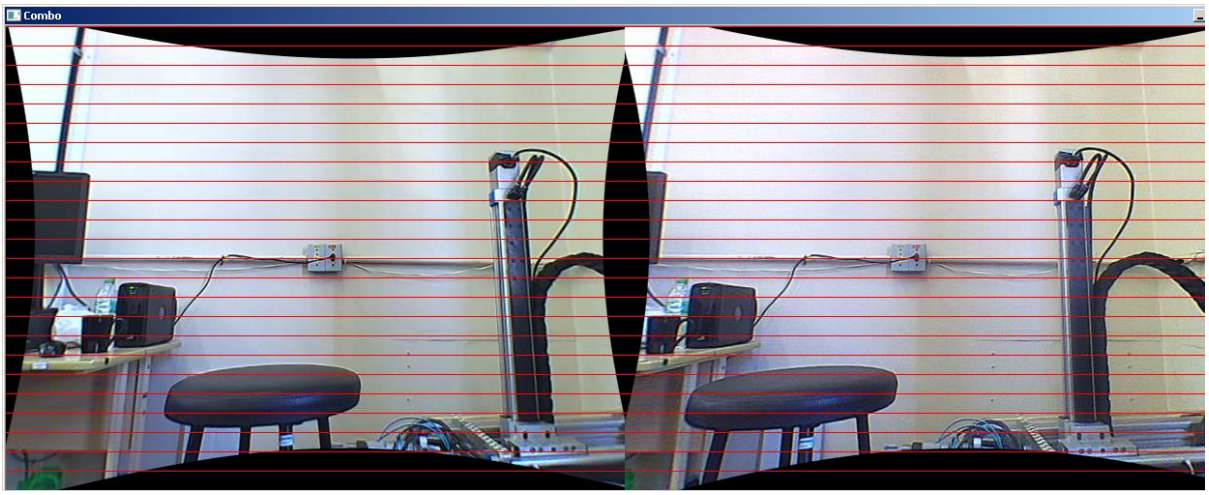

Figura 4.8 - Imágenes estereoscópicas después de la calibración y rectificación (cámaras izquierda y derecha)

En la Figura 4.8 se pueden observar las imágenes rectificadas usando el algoritmo de Bouguet, refiérase al apartado 3.5 para ver los detalles de rectificación. En los bordes se puede ver, como se dijo, el resultado de las inconsistencias de los cálculos. Es posible recortar la resolución de las imágenes, que originalmente se muestran en formato de 640x480 pixeles, para evitar el cálculo de la disparidad en los bordes donde no se tendrán buenos resultados. En la implementación, sin embargo, no se hizo el recorte de las imágenes.

Una vez rectificadas las imágenes, el sistema reconoce el objeto deseado según el *target* previamente escogido. En el caso, el objeto *target* es una pequeña botella escogida por sus dimensiones y color. En el reconocimiento se detectan los contornos por medio del método Canny discutido en el apartado 3.6. Además, se aplican transformaciones morfológicas para mejorar el contraste entre un borde y un área encerrada.

El programa de reconocimiento busca en la carpeta correspondiente el archivo .txt con la información del contorno *target* a ser buscado, en caso de fallar la lectura se le pide al usuario que use una imagen con el objeto y se calculan los contornos. En este momento, el programa saltará al módulo de detección de objeto *target.* Manipulando las barras se puede llegar a definir el contorno deseado que se guardará en la base de datos.

Una vez se tienen guardados los contornos del objeto (un ejemplo se puede ver en la Figura 4.9), o se tiene datos de contornos en los archivos, con una confirmación del usuario (presionando algún botón), inicia la fase de reconocimiento. En este caso, el reconocimiento se hace mediante el cálculo de momentos de los contornos disponibles en las imágenes.

En la Figura 4.10 se puede observar el reconocimiento de objetos y los valores de momentos calculados. Se toman dos criterios de selección:

1. Usando la ecuación (3.5) de comparación de momentos normalizados, se hallan valores aceptables de modo tal que si:

$$I_3(A,B) < 0{,}41, \text{ el objeto coincide con el target}$$

$$I_3(A,B) > 0{,}41, \text{ el objeto no coincide con el target}$$

Este valor límite fue hallado después de algunas experimentaciones donde se observaron las coincidencias de detección que abajo se muestran en la tabla 4.1. En cada caso la cantidad de muestras se pueden observar a la derecha de la tabla 4.1. Cada muestra consistió en una fotografía tomada a un grupo de objetos entre los cuales se encontraba un objeto que *target,* un ejemplo se puede observar en la Figura 4.10.

Tabla 4.1 - Porcentajes de coincidencia según el valor de comparación I_3 (A,B)

Valor $I_3(A,B)$	Coincidencias	Fallas	Muestras
Hasta 0,1	3%	97%	10
Hasta 0,25	15%	85%	7
Hasta 0,35	68%	32%	8
Hasta 0,5	65%	35%	8
Hasta 1	27%	73%	9

2. Un valor cercano al 70% de coincidencias, sin embargo, no es del todo aceptable. Por esto se hizo uso de un método simple de selección por cantidad de puntos detectados. Por cada grupo de contornos detectado que describe una forma, se hace la suma de ±20 puntos. Si el objeto está dentro del rango de coincidencia menor al 0,41 y además la cantidad de puntos está dentro del valor máximo y mínimo de puntos admitidos, el objeto es específicamente el deseado. Con este método se pudo mejorar las coincidencias hasta un 90%. Los contornos que a la comparación den valores menores a 1, tengan al menos la misma cantidad (±20 o el valor que se elija como variable) de puntos correspondientes a los contornos. Por ejemplo, si el contorno patrón tiene 40 puntos que describen la forma, el contorno encontrado deberá tener mínimo 20 puntos o máximo 60 puntos para ser aceptado.

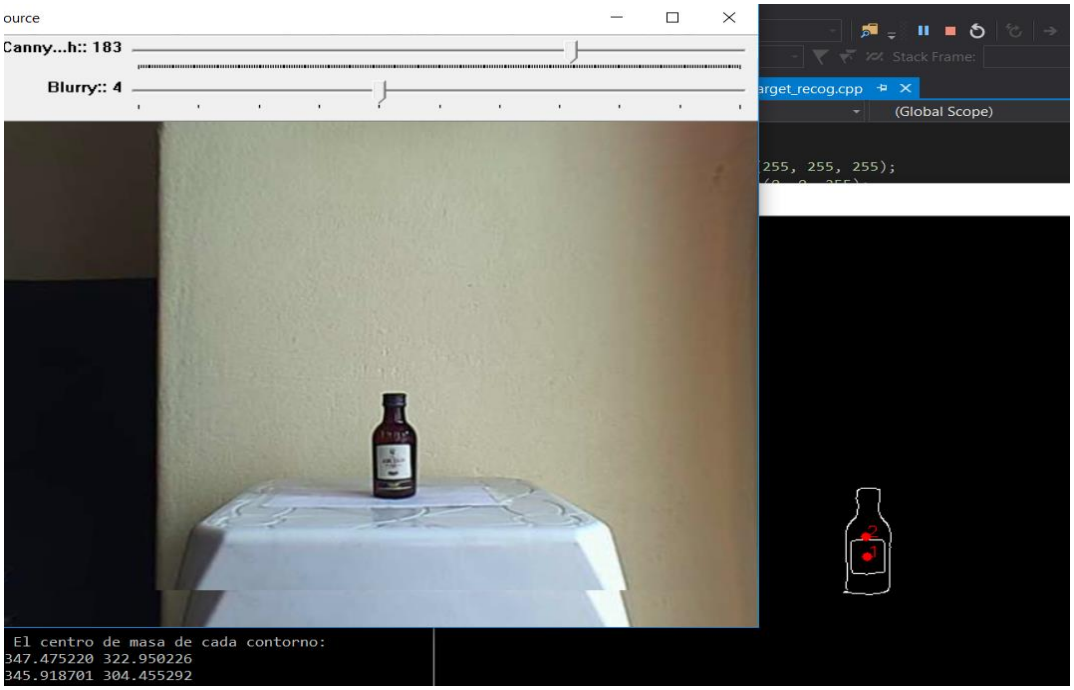

Figura 4.9 - Objeto patrón o target, se pueden observar dos contornos definidos

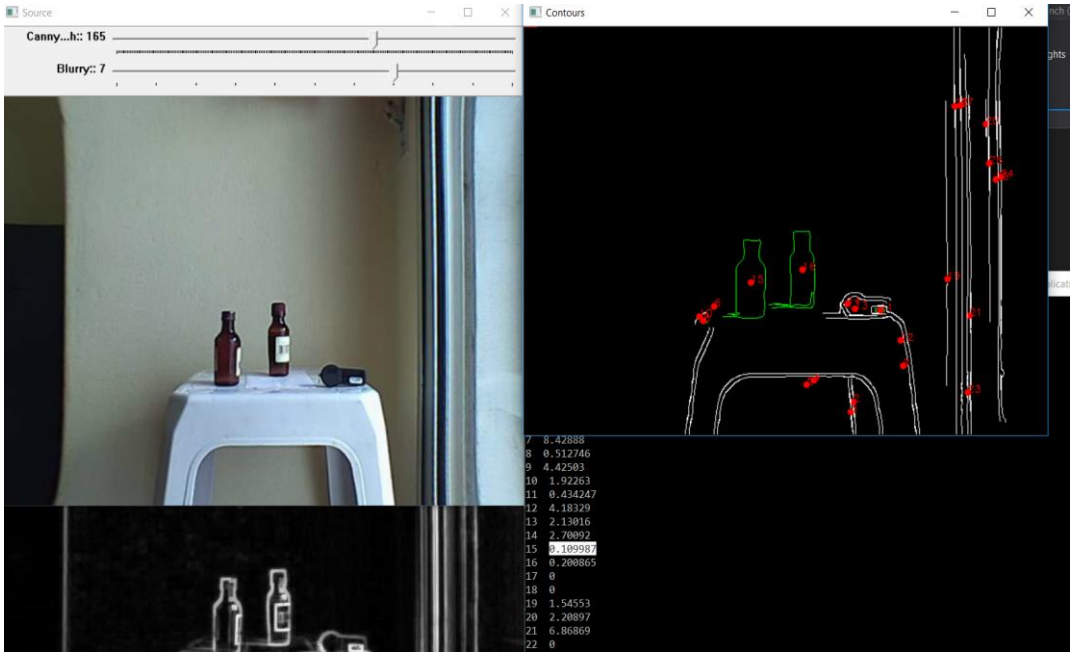

Figura 4.10 - Detección de objeto target desde un entorno que tiene otros objetos

Se implementaron estas dos condiciones ya que, en la práctica, se observa que muchos contornos que no corresponden a la forma deseada pueden dar como resultado un número menor a 0,41 en la comparación ya que se asemeja la posición del centro de masa aleatoriamente.

Una vez el algoritmo encuentra contornos que cumplen las dos condiciones se calcula el centro de masa con los momentos de primer orden y se imprime la coordenada. Figura 4.11

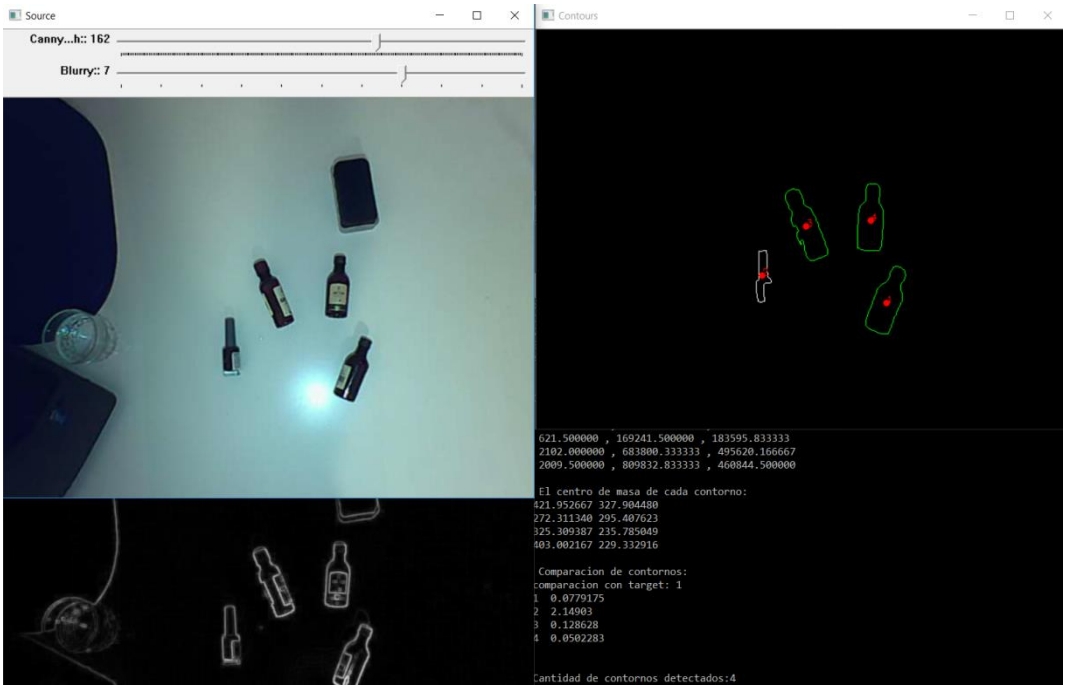

Figura 4.11 - Impresión de coordenadas de centro de masa de objetos cuyos contornos son coincidentes en la comparación

Las coordenadas halladas del centro de masa en pixeles se transforman después en coordenadas del mundo real usando la homografía armada con las matrices de rotación y traslación halladas en el punto de calibración estereoscópica.

En la Figura 4.12 se puede observar un ejemplo de contorno *target* reconocido con éxito con el entorno que también enfoca el robot. Cuando es reconocido el objeto se marca en color verde. En la Figura 4.12 a la derecha abajo se puede ver el centro de masa y la línea describiendo la orientación del objeto.

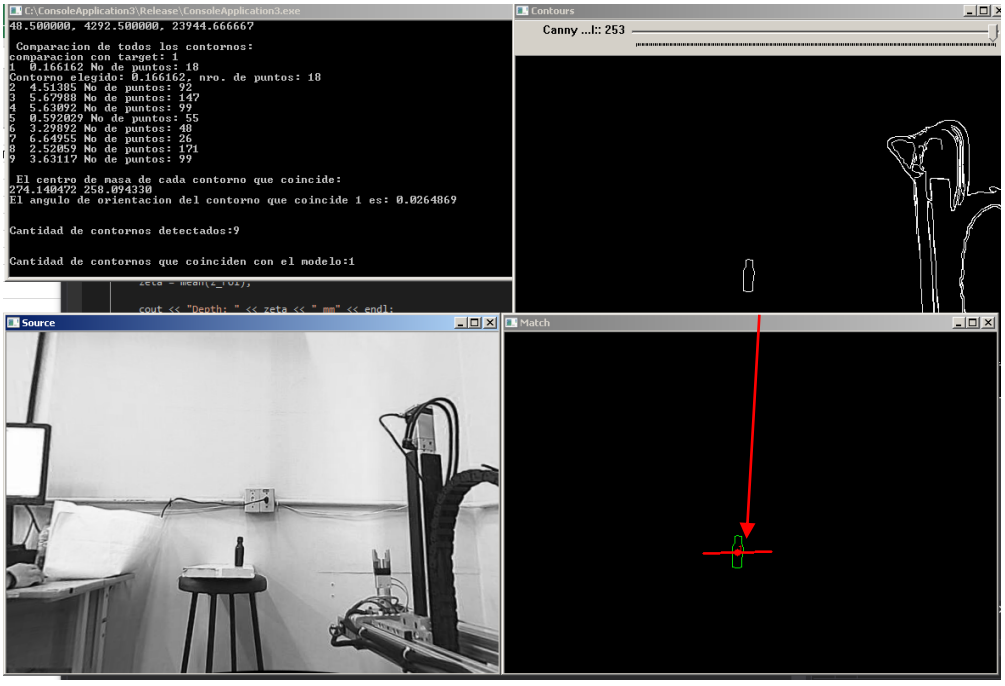

Figura 4.12 - Detección y reconocimiento de contornos

Las coordenadas del centro de masa $(\bar{x}, \bar{y})$ y la orientación de la botella sirven como base para el siguiente paso que es el cálculo de las coordenadas del mundo. También, estos datos se usan para la elección de la ventana de cálculo de la nube de profundidades que determinan la coordenada z.

Para el cálculo de las coordenadas del mundo, se requiere que los sistemas cartesianos del robot y de la cámara estén en el mismo punto origen. El punto $(0,0)$ origen del sistema cartesiano del robot, será bastante diferente al origen del sistema cartesiano de las cámaras, sin embargo, es posible relacionarlos tomando en cuenta la traslación de un punto en las imágenes de la cámara y el punto origen en el sistema cartesiano del robot.

En la Figura 4.13 se puede observar el punto inicial escogido. La detección de las esquinas del tablero de esta imagen nos permite, con la función de calibración, hallar la traslación y rotación del primer punto respecto del centro de la imagen. Este trabajo se realizó con la cámara izquierda por tener una posición más favorecida con respecto a los objetos que serán observados.

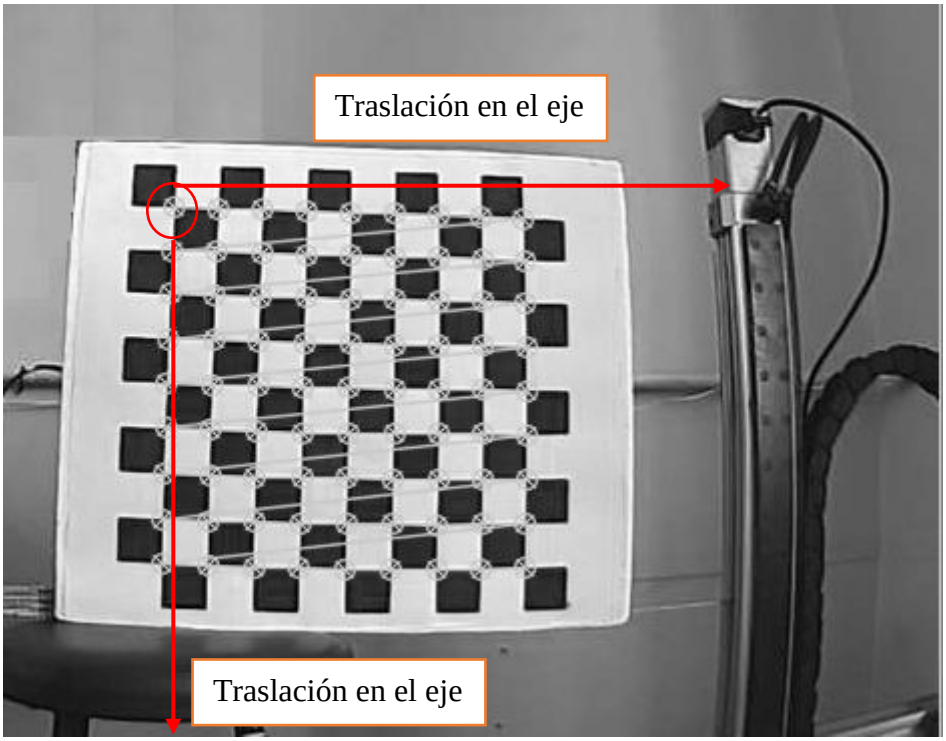

Figura 4.13 - Punto de inicial para relación entre sistemas de coordenadas de robot y cámaras

Se hizo una medición para conocer la traslación en coordenadas de mundo del punto mostrado en la Figura anterior. La medición en el caso de estudio y en la última experimentación dio (0,54 , 0,55) m. Se puede ver fácilmente que esta medición va a tener un margen de error. En las experimentaciones se encontró un valor sistemático que fue compensado según el error constante entre la medición de la posición de un objeto y el resultado del cálculo (ver ecuación de la homografía 2.26). Por ejemplo, si el valor de la coordenada del objeto en la imagen daba (0,1 , 0,1), el valor compensado final en el mundo sería (0,1+s1, 0,1+s2). En la experimentación se encontraron los valores compensados finales:

- Para el eje x $s1 = 0{,}112$ m

- Para el eje y $s2 = 0{,}08$ m

A continuación, en la tabla 4.2 se muestra la relación entre el sistema cartesiano del robot y el de las cámaras donde:

Tabla 4.2 - Relación de coordenadas del robot y de las cámaras

Cámaras eje	Robot eje
x →	Y
y →	Z
z →	X

Se hace uso de la ecuación de homografía (2.26) que despejada nos da Ecuación (4.1):

$$\widetilde{Q'} = [sMW]^{-1}\tilde{q} \tag{4.1}$$

Donde M, matriz de parámetros intrínsecos de la cámara izquierda y W, matriz de rotación y traslación del punto escogido, fueron halladas previamente en la calibración de la cámara. Por otro lado, $\tilde{q}$ es el vector de puntos (x, y) que se obtiene del centro de masa y s el valor mencionado de corrección que en el caso de estudio corresponde a $(0{,}112\ , 0{,}08)$.

El valor del vector $\widetilde{Q'}$ con coordenadas del mundo nos permite finalmente generar la ventana de inspección en la nube de profundidades.

En la etapa de la correlación o correspondencia estereoscópica, con la confirmación del usuario (presionando cualquier botón) se inicia el cálculo de la correlación usando como base el algoritmo StereoSGBM (Semi Global Block Matching por sus siglas en inglés), mencionado en el apartado 3.6.

El algoritmo toma como base las imágenes rectificadas para el cálculo de disparidad, que adaptada de la ecuación (2.29):

$$d = x_l - x_r - (c_{x\,izq-}\, c_{x\,der}) \tag{4.2}$$

Como base de deben declarar dos parámetros, uno es la disparidad mínima que en el caso ideal sería 0, pero por la rectificación llegó a ser de 12 pixeles en la implementación, y segundo el dato de número de disparidades, que sería la diferencia entre la disparidad máxima y la disparidad mínima donde podría encontrarse una coincidencia, en este caso se puso una ventana de 5 pixeles por punto de interés. Se debe tomar en cuenta que en la implementación se evidenció que estos valores pueden variar según la iluminación. Como ayuda inicial y para corroboración de mediciones, como se observa en la Figura 4.14 se optó por ingresar estos valores por barras deslizadoras hasta obtener una medición de profundidad y una imagen reconstruida coherente.

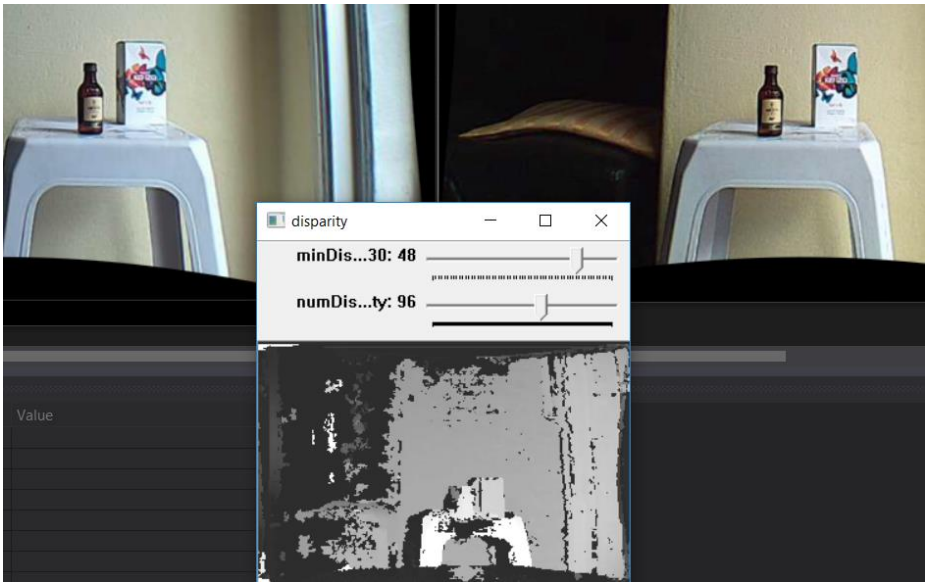

Figura 4.14 - Mapa de disparidades de un objeto observado por dos cámaras.

En este punto se está en condiciones de hallar los valores de las profundidades como se muestra en la Figura 4.15. El mensaje muestra el tiempo de procesamiento que toma el algoritmo en armar el mapa de disparidades y el valor de la coordenada Z del objeto más cercano al plano de observación.

Figura 4.15 - Resultados de profundidad del objeto observado marcado por el cuadro rojo

El valor de la profundidad es calculado mediante un barrido por la ventana de inspección mostrada en la figura 4.15. Se halla la media de la medida de profundidad de cada

uno de los pixeles que están dentro de la ventana. La misma, se construye con el punto del centro de masa previamente hallado con 40 x 40 pixeles de área alrededor.

La nube de profundidades refiere al gris más claro como los objetos que están más cercanos a la cámara mientras que el gris más oscuro como objetos que se encuentran a mayor distancia de la cámara.

En este caso también es necesario relacionar los ejes x del robot y z de la cámara según el montaje, ya que la relación de distancia entre el origen de medición de las cámaras no será el mismo que el origen del eje x en el robot. Para el caso específico de este montaje, la diferencia fue de 0,945, es decir, las cámaras se encuentran montadas a esa distancia del origen del eje x del robot.

Una vez terminado este proceso, teniendo las coordenadas (x, y, z) del objeto reconocido en el sistema de coordenadas del mundo, al igual que su orientación, se transfieren estos datos al sistema de planeación de trayectorias del robot que se describe en el siguiente apartado.

4.2 Algoritmo de planeación de trayectorias

Las coordenadas (x, y, z) del mundo que describen la posición del objeto target que se requiere manipular, son transferidas al software RSLogix5000 mediante un módulo de comunicación que se desarrolló en C++ en el sistema de visión. Este módulo usa el método DDE (Dynamic Data Exchange), que Microsoft desarrolló en los años 90s como método de comunicación entre programas basados en sistema operativo Windows.

La transferencia implica que el sistema pasa los datos de las coordenadas de posición a manera de vector del modo:

$$P = [x, y, z, roll, pitch, yaw]^T$$

Este vector P contiene la información referida en el anterior apartado como los valores de las coordenadas del centro de masa del objeto reconocido como *target* y su orientación. La orientación $roll$ será siempre igual a $-\pi/2$ o 90° ya que la botella debe ser manipulada desde el frente tal como se observa en la Figura 4.16. La orientación $pitch$, corresponde a la medición de la orientación del objeto usando los momentos de segundo orden como ya fue explicado en el apartado 3.6

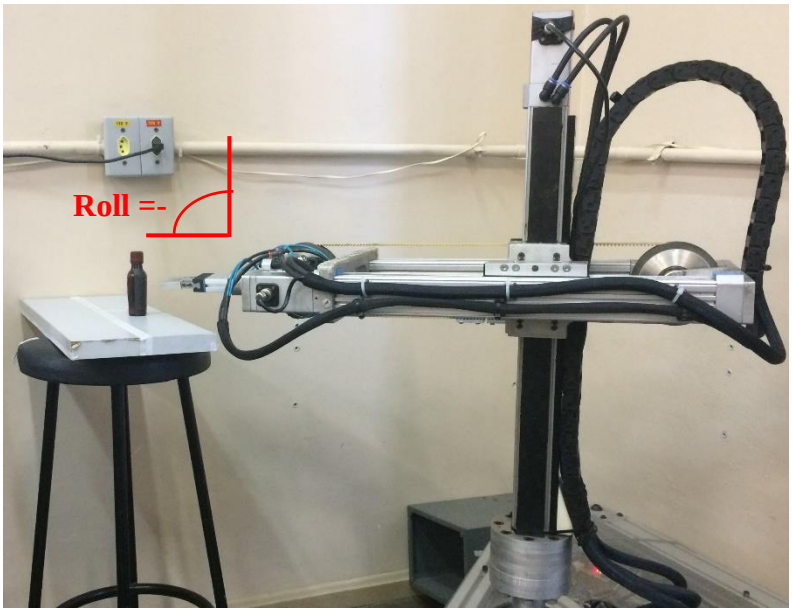

Figura 4.16 - Representación de la orientación Roll del robot RP5GDL

En el caso del robot RP5GDL, la orientación *yaw* siempre será cero ya que no se cuenta con la junta rotacional que logre este movimiento. Sin embargo, en Leonardelli, 2015, este valor es requerido ya que se considera una programación estándar capaz de usarse para programar movimientos de cualquier robot.

Después de transmitir el vector de posición al software RSLogix5000, se programaron en el mismo software, los movimientos requeridos del robot que lograrán la manipulación. Como se puede ver en la Figura 4.17, se usaron cuatro bloques de programación de movimientos lineales en el diagrama GRAFCET según los módulos sugeridos por Leonardelli, 2015.

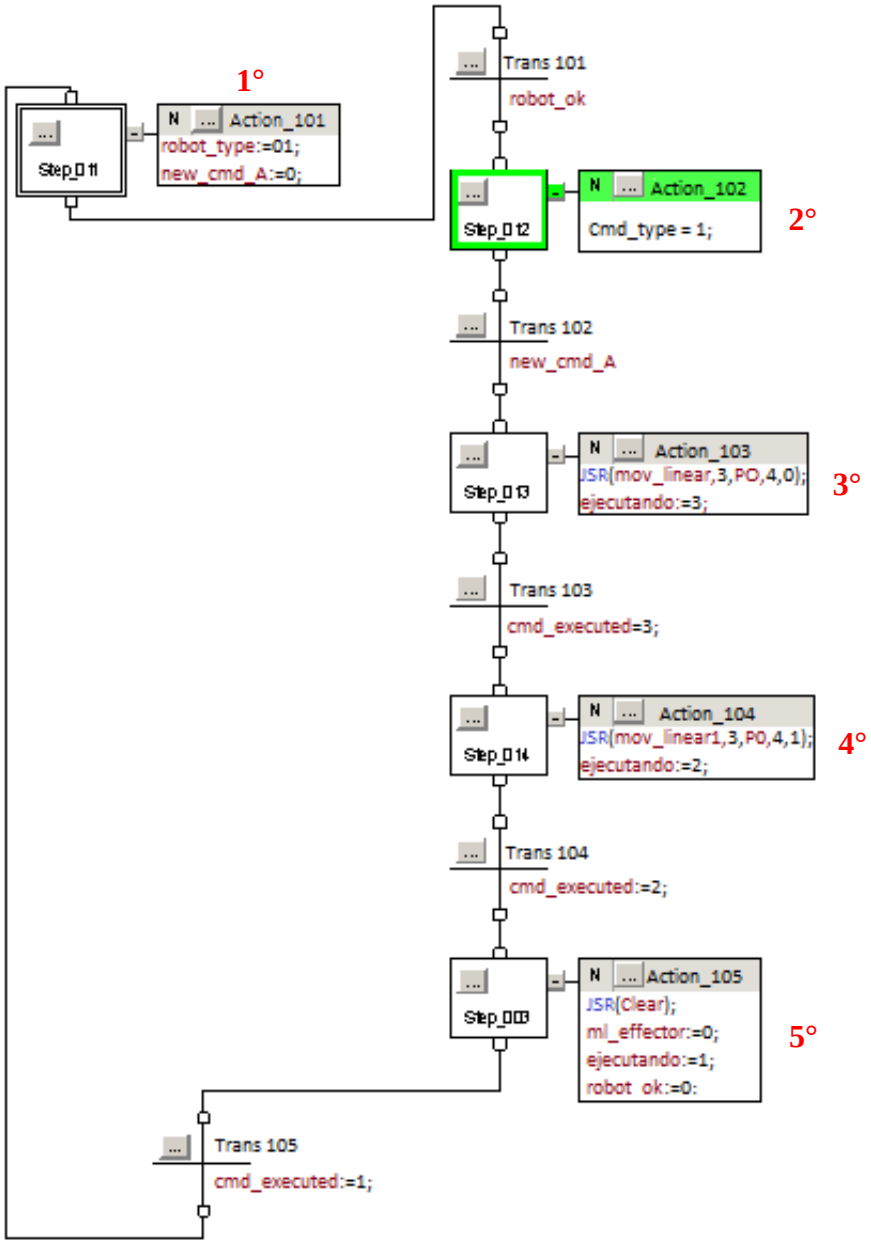

Figura 4.17 - Bloques de programación GRAFCET, desde el software RSLogix5000

Los bloques describen:

1. El primer bloque de confirmación de estado *ready* del robot: que considera que las variables de intercambio entre robot y placa de control dSpace han podido ser leídas y el robot se encuentra en el punto de inicio. En la aplicación este punto fue definido considerando la retracción de las juntas 2 y 3, y la junta 1 de modo que se dé espacio al objeto de ser posicionado en frente del robot. Para la aplicación se programó este punto en el vector de posición $P_i = [-1{,}2,\ 0{,}07,\ 0{,}03, 1{,}57, 0\ ,0]^T$.
2. El segundo bloque espera la confirmación de *start* del movimiento, este dato es comunicado desde Matlab.
3. El tercer bloque mueve el robot hasta la posición $P_1 = [x,\ y, 0{,}03, -\pi/2\ , pitch\ ,0]^T$. Y abre la garra mediante la escritura en el comando

según el formato propuesto por Leonardelli, 2015: Rutina(datos de entrada) = Rutina (tipo movimiento, punto final, tiempo de ejecución, estado de la garra). Se observa que la coordenada z se mantiene en 0,03 como en el punto inicial. Esto se debe a que el primer movimiento posicionará las juntas 1,2,4 y 5 sin mover la 3 para evitar colisionar con el objeto de manipulación.

4. El cuarto bloque recibe la confirmación de que el anterior punto fue alcanzado y se dispone a mover el robot hacia el punto $P_1 = [x,\ y, z, -\pi/2, pitch, 0]^T$.
5. El quinto bloque recibe la confirmación de que el anterior punto fue alcanzado y procede a cerrar la garra. En este momento el objeto es capturado.

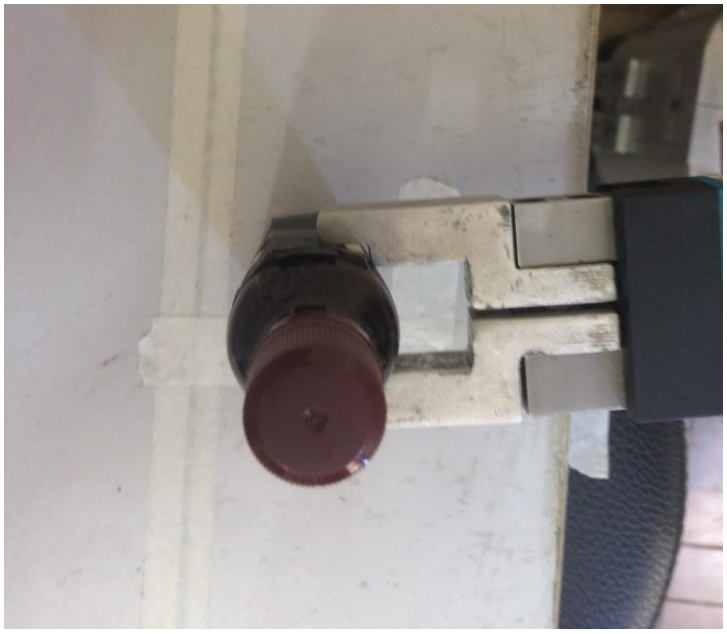

Figura 4.18 - Objeto target capturado por la pinza del robot RP5GDL

La planeación de trayectorias para cada punto usa el método de *splines* propuesto por Missiaggia, 2014. Para cada punto provisto por el sistema primeramente se hace el cálculo de la cinemática inversa en el bloque de movimiento lineal propuesto por Leonardelli, 2015, usando las ecuaciones descritas en el apartado 3.8. Tal como se observa en la Figura 4.19.

```
gl_01_0:=-ASIN(x/SQRT(x*x+y*y))+0.15; //.0.15 es un valor de compensación por tener un offset en el GL1
gl_02_0:=z-0.2273*SIN(ang4rad);
gl_03_0:=(y/COS(gl_01_0-0.15))-0.1945-0.2273*COS(ang4rad); //tambien se debe corregir 0.15
gl_04_0:=ang4rad;
gl_05_0:=ml_Ponto.Pitch;
gl_06:=ang6rad;
```

Figura 4.19 - Cálculo de la cinemática inversa del robot RP5GDL

Además, en este bloque se hicieron algunos cambios como se puede observar en la Figura 4.19 para el "gl_01_0" o GDL 1, se añadió al final la compensación de 0,15 tratando de corregir el error sistemático del control que desvía la posición deseada en 0,15 rad. Esta condición fue descrita en las consideraciones iniciales tomadas en el apartado 4 de la implementación.

Finalmente, con las correcciones y los cálculos necesarios se generan las trayectorias mediante *splines*, las que, por su parte describen las siguientes gráficas respecto de las posiciones deseadas y las alcanzadas, Figura 4.20.

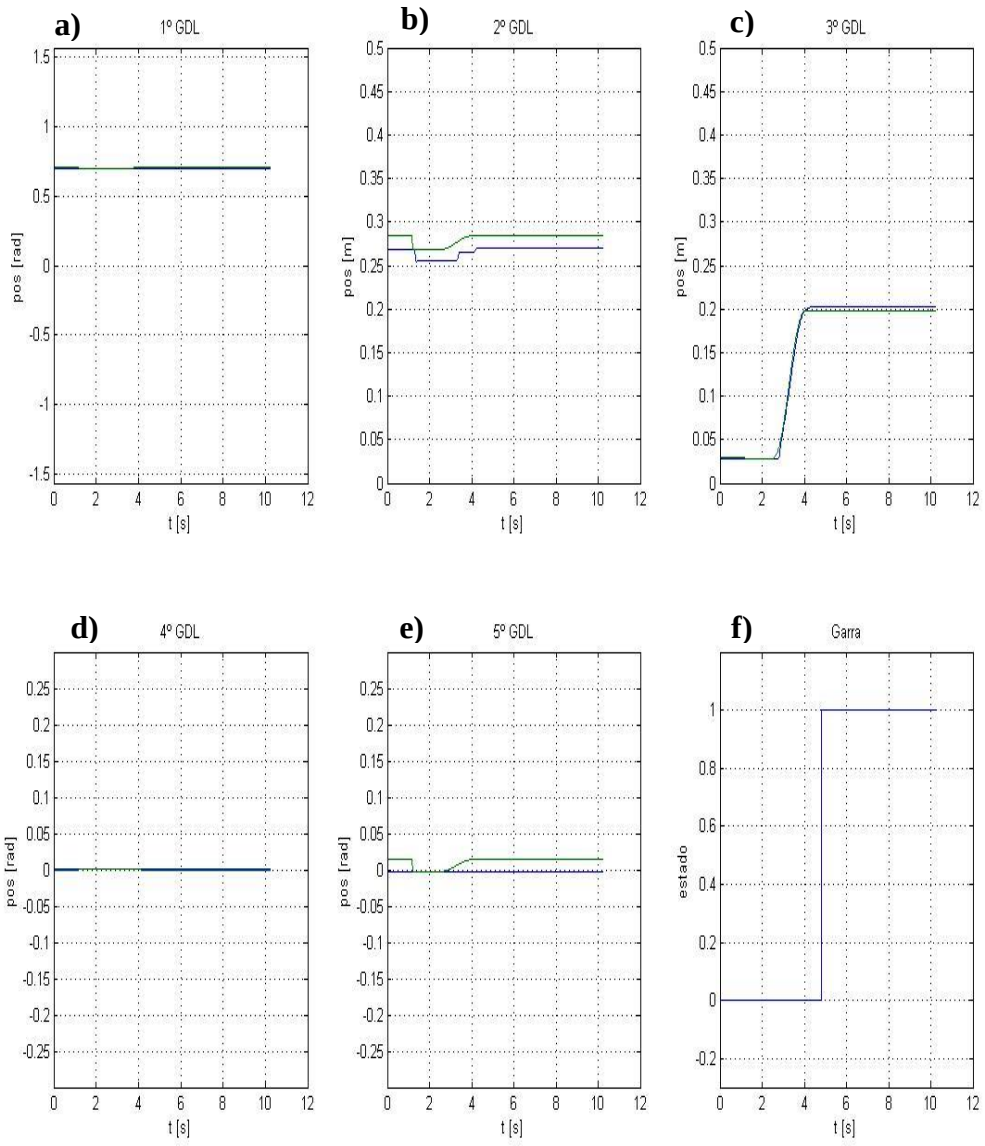

Figura 4.20 - Posición respecto del tiempo de cada una de las juntas en la trayectoria hacia el punto inicial. a) Gráfica de posición del GDL1, b) Gráfica de posición del GDL2, c) Gráfica de posición del GDL3, d) Gráfica de posición del GDL4, e) Gráfica de posición del GDL5, f) Gráfica de estado de la garra

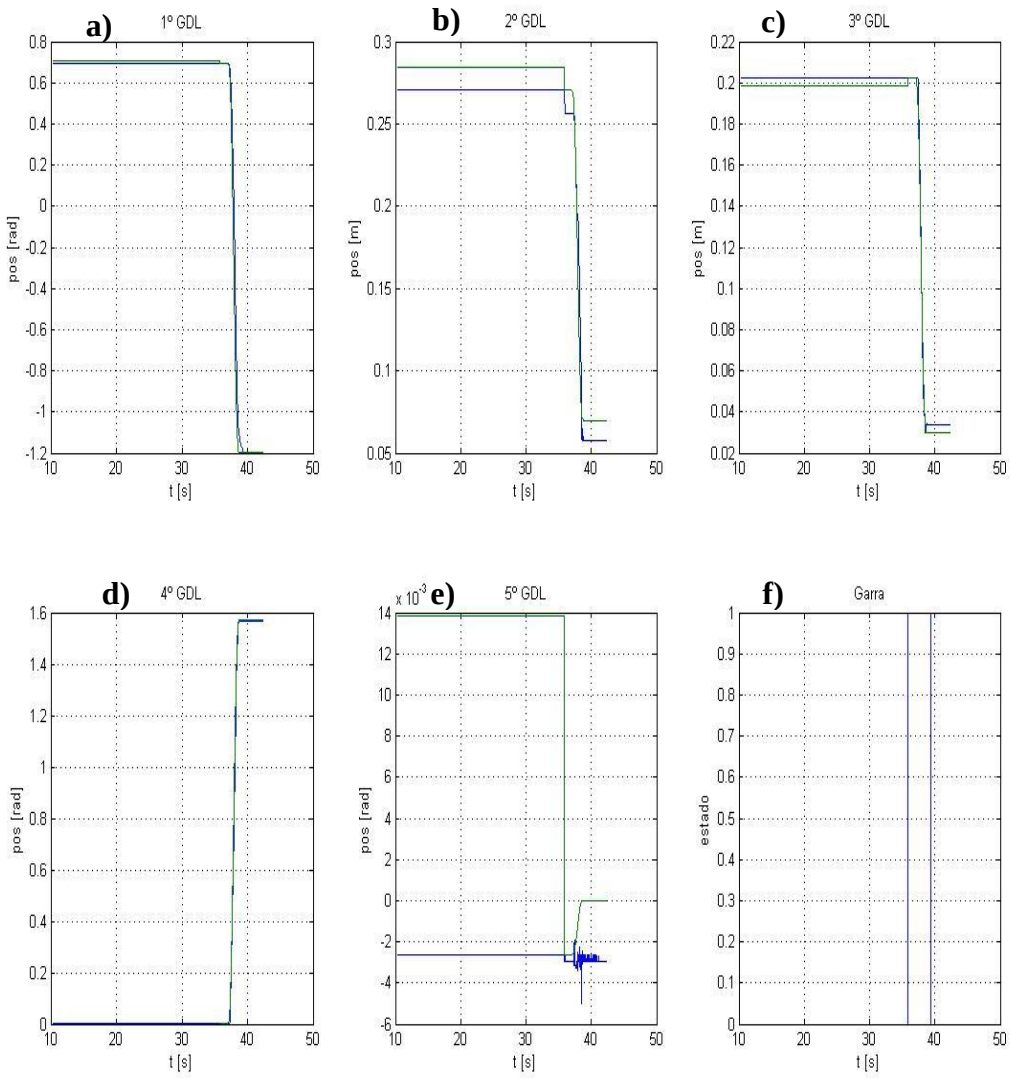

Figura 4.21 - Posición respecto del tiempo de cada una de las juntas en la trayectoria hacia el centro del objeto a) Gráfica de posición del GDL1, b) Gráfica de posición del GDL2, c) Gráfica de posición del GDL3, d) Gráfica de posición del GDL4, e) Gráfica de posición del GDL5, f) Gráfica de estado de la garra

Se observa un error de régimen estacionario en los GDL 2 y 3, que se mantienen sistemáticos. Para poder lograr la manipulación del objeto, desde el sistema de visión, estos valores tuvieron que ser compensados.

En el caso del GDL 5, se observa un error relativamente importante, sin embargo, al tratarse de la orientación $pitch$ del objeto, en la experimentación no se observaron inconvenientes de manipulación ya que está en menos de 3 rad.

5. RESULTADOS

El sistema de visión fue implementado al robot por medio de la transmisión de coordenadas en tres dimensiones y la orientación del objeto reconocido, con el objetivo de que cualquier *target* pueda ser manipulado de manera satisfactoria generando las trayectorias necesarias en el robot. En este apartado se muestran los resultados de este trabajo según como el sistema planeado inicialmente.

Se pueden ver los resultados preliminares de medición de las coordenadas a diferentes distancias del lente de la cámara (tomada como coordenada z) en las gráficas de las Figuras 5.1, 5.2 y 5.3. Estas gráficas plasman los resultados de 28 muestras tomadas con el sistema estereoscópico.

Un objeto era centrado en el espacio de trabajo del robot de modo que pudiera ser manipulado. Cada muestra implica la medición de las coordenadas del centro de una pieza en los tres ejes usando las cámaras en el sistema estereoscópico y su correspondiente medición en el mundo real usando una cinta métrica (tomando en cuenta el error de medición). De los 28 intentos de sujeción, sólo el 36% se trataron de botellas capturadas por la garra.

Se observa las diferencias de medición con las cámaras y la medición en el mundo real con una cinta métrica en la Figura 5.1 para el eje x.

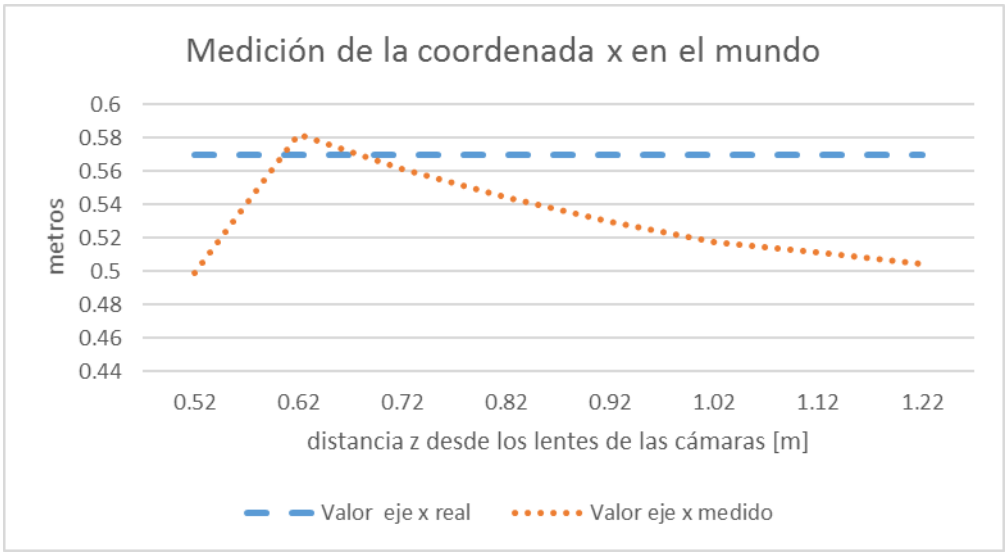

Figura 5.1 - Gráfica de la medición hecha por el sistema de visión vs la medición hecha en campo de la coordenada x en el mundo

De la coordenada y en la Figura 5.2:

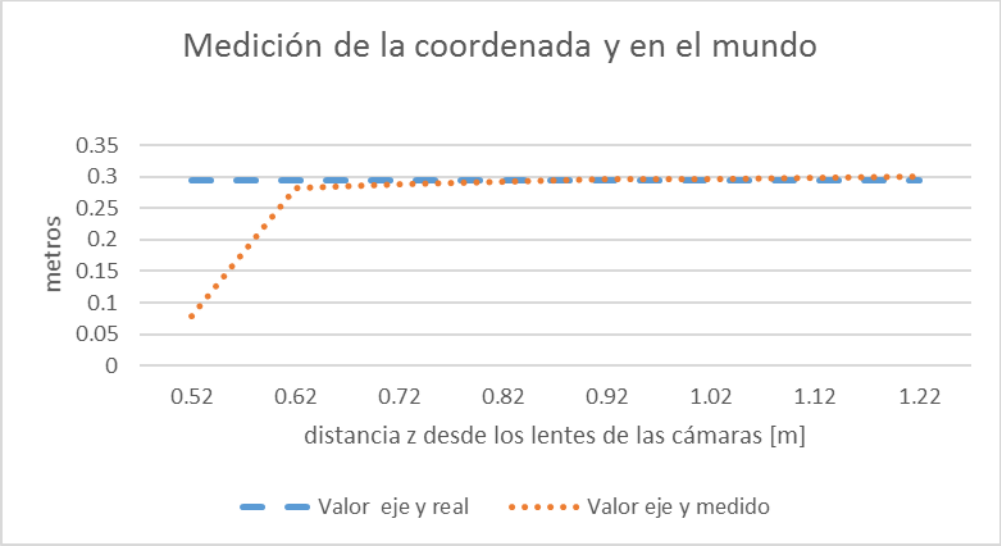

Figura 5.2 - Gráfica de la medición hecha por el sistema de visión vs la medición hecha en campo de la coordenada y en el mundo

De la coordenada *z* en la Figura 5.3:

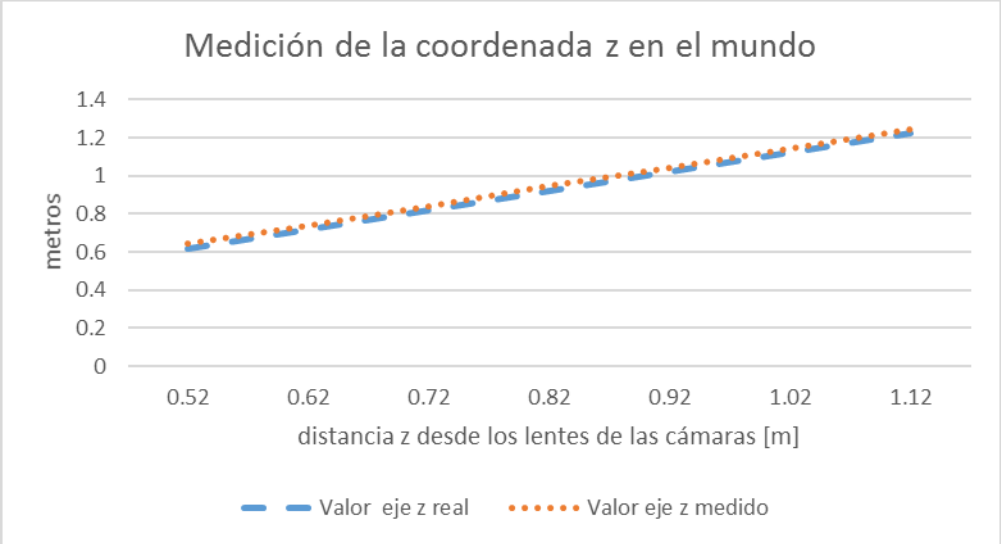

Figura 5.3 - Gráfica de la medición hecha por el sistema de visión vs la medición hecha en campo de la coordenada z en el mundo

Se puede observar que en el eje x, existe una diferencia de medición importante. Como se ve en la Figura 5.4:

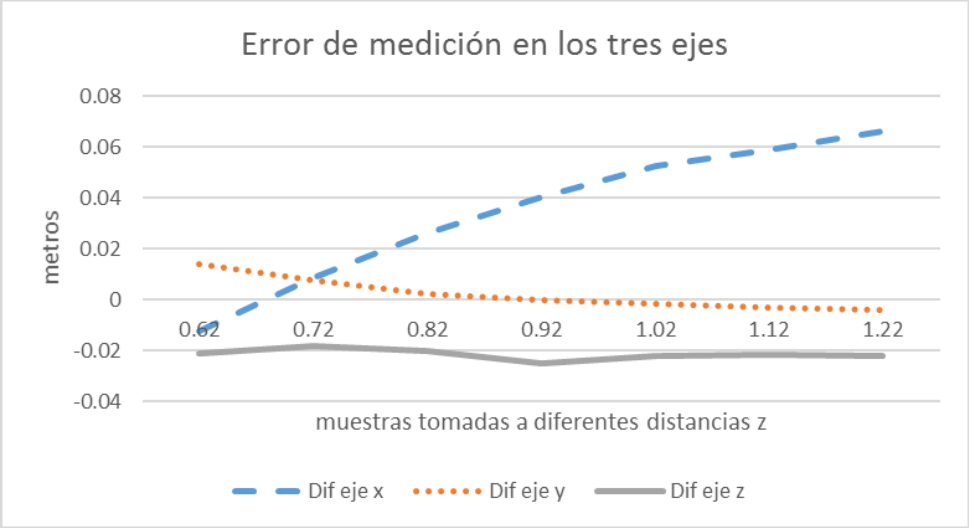

Figura 5.4 - Gráfica de las diferencias de mediciones entre lo observado por las cámaras y lo medido

Dado que esta diferencia de medición es aproximadamente en algunos casos hasta de 0,07 m o 70 mm, no se puede pensar en que el sistema pueda ser capaz de manipular un objeto. Tomando en cuenta que para la manipulación la tolerancia es de ± 0,009 m o 9 mm (calculado desde el tamaño máximo de la garra abierta y el diámetro de la botella), este valor no es aceptable. Eso se puede notar en los resultados, ya que de 28 botellas sólo fueran capturadas 10, el 36% mencionado anteriormente. Estas botellas se encontraban a distancias no muy lejanas de las cámaras que están entre los 0,65 y 0,75 m.

Para poder mejorar estos resultados y dado que la diferencia en la medición en el eje x, describe una función más o menos uniforme y curva hacia mayores valores, es que se intentó corregir esta medición con un polinomio de tercer grado, el cual corrige el valor de la distancia z medida desde el objeto hacia las cámaras. Ecuación (5.1):

$$Pl_{xc} = 0{,}0440x^3 - 0{,}2732x^2 + 0{,}5171x - 0{,}2384 \quad (5.1)$$

Los resultados de esta corrección se pueden ver a continuación. Para el caso de la medición de la coordenada x en el mundo y la entregada por el sistema de visión, la Figura 5.5 muestra que la precisión es mucho mayor que la observada en la Figura 5.1:

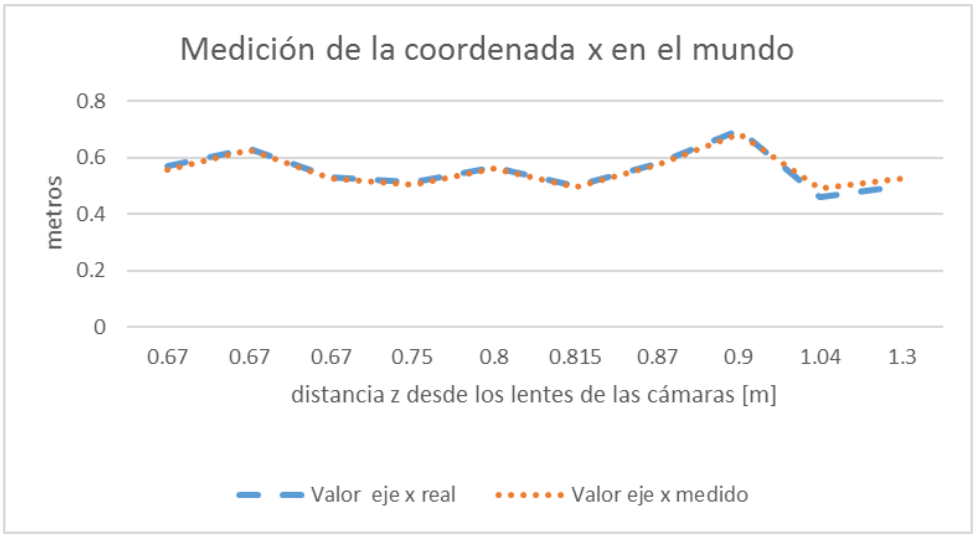

Figura 5.5 - Gráfica de la medición hecha por el sistema de visión vs la medición hecha de la coordenada x en el mundo

Para el caso de la medición de la coordenada *y* en el mundo y la entregada por el sistema de visión, la Figura 5.6 muestra los resultados:

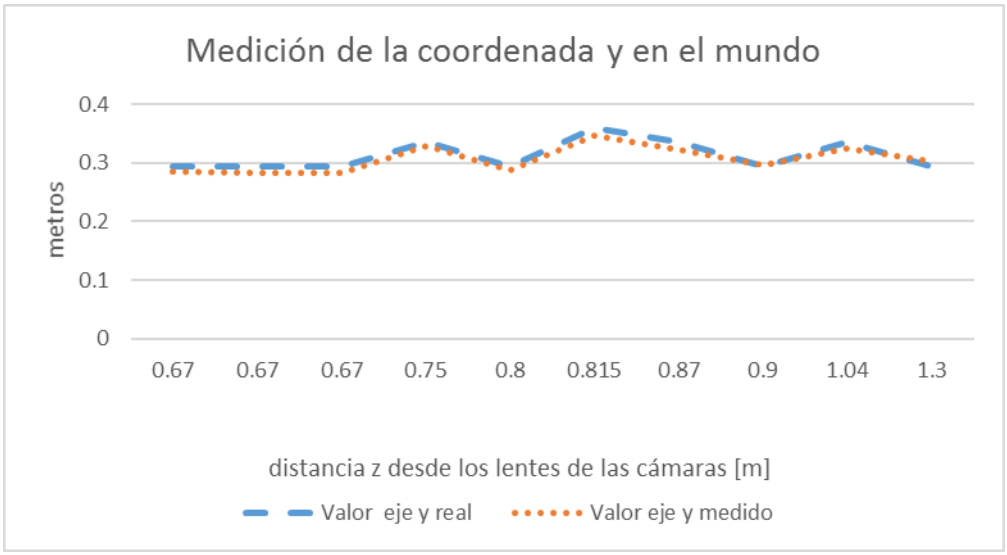

Figura 5.6 - Gráfica de la medición hecha por el sistema de visión vs la medición hecha en campo de la coordenada y en el mundo

Para el caso de la medición de la coordenada *z* en el mundo y la entregada por el sistema de visión, Figura 5.7 muestra los resultados:

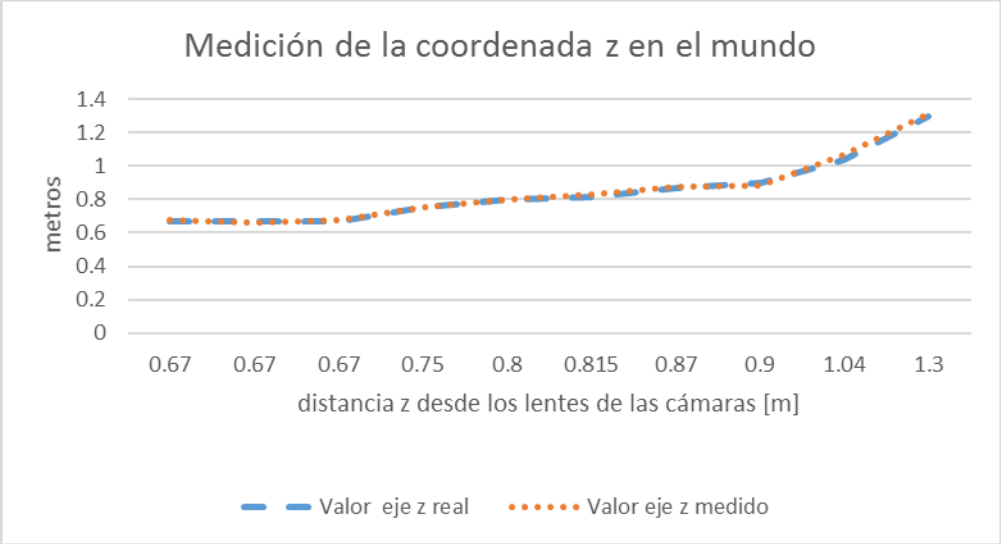

Figura 5.7 - Gráfica de la medición hecha por el sistema de visión vs la medición hecha en campo de la coordenada z en el mundo

A continuación, se pueden ver los errores de medición en los tres ejes en la Figura 5.8. Estos resultados contemplan la corrección de medición en la coordenada del eje x.

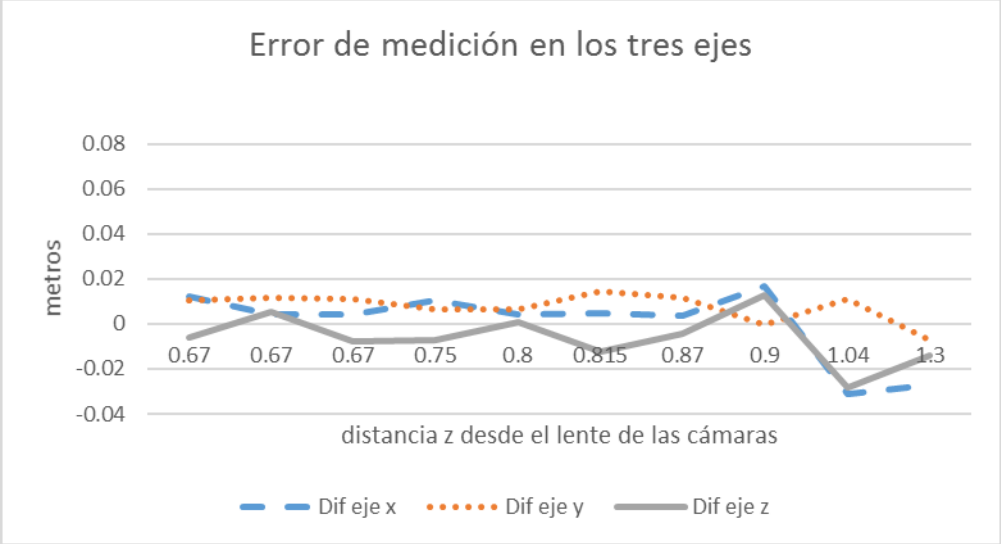

Figura 5.8 - Gráfica de las diferencias de mediciones entre lo observado por las cámaras y lo medido después de la corrección con el polinomio de tercer orden aplicado al eje x

Con estas nuevas condiciones se verificó que de 10 botellas se pudieron capturar 7, es decir, que en el 70% de los casos se logró manipular el objeto *target* deseado. La precisión promedio de medición de las coordenadas en los tres ejes es de:

Tabla 5.1 - Precisión de medición del sistema de visión de las tres coordenadas (x,y,z)

Eje x [m]	Eje y [m]	Eje z [m]
±0,012	±0,009	±0,009

Lo que le da una precisión al sistema de visión de: ± 0,010 [m], respecto de la medición de posición.

6. CONCLUSIONES Y SUGERENCIAS PARA FUTUROS TRABAJOS

Después de ver los resultados obtenidos en el anterior apartado y tomar en cuenta los objetivos específicos tratados en el apartado del primer capítulo, se puede decir que con certeza es posible manipular objetos usando dos cámaras en un sistema esteroscópico y un robot neumático de 5 grados de libertad, programándolo mediante GRAFCET. Sin embargo, por lo observado será necesario tomar en cuenta el tamaño del objeto en relación a la garra ya que por lo menos serán necesarios 10 mm por lado para poder atrapar el objeto con seguridad.

Por otro lado, la medición de la coordenada *z* representa un hallazgo importante en el área de procesos de fabricación, ya que demuestra que con precisión bastante aceptable un robot es capaz de observar un objeto a la distancia y poder calcular su posición en tercera dimensión. Lo cual se trabaja actualmente en los robots de asistencia como en [Natarajan, Durrant et al., 2011] y [Rusu, Holzbach et al., 2009], los cuales no requieren una gran precisión. En el área de manufactura, esta característica es muy útil si se trata de objetos montados en líneas de producción que deben ser empacados, seleccionados o transportados como en [Kanellakis, Kyritsis et al., 2015].

Respecto a los inconvenientes encontrados, se pudo observar que la iluminación es un elemento crucial en la calibración, la medición de coordenadas y el armado de la nube de profundidades.

Como sugerencia para futuros trabajos:

- El uso de segmentación de colores e histogramas en combinación con el procedimiento presentado será útil para lograr mayores precisiones que las obtenidas, a fin de evitar los problemas de iluminación y mejorar/facilitar el reconocimiento de piezas.
- El uso de una tercera y/o cuarta cámara sería un gran aporte ya que permitiría cubrir completamente el volumen de trabajo del robot y ayudaría a distinguir correctamente objetos aun cuando estos estuvieran ocluidos.
- El uso de múltiples cámaras en un sistema estereoscópico permite la comparación de disparidades entre pares de cámaras lo que mejora y corrige el valor de disparidad encontrado, ya que es posible usar valores calculando una media. Sin embargo, una desventaja podría ser que la calibración resulte en un proceso más largo y con mucho más error propagado en las rectificaciones.

REFERENCIAS

Azad, Pedram; Asfour, Tamim; Dillmann, Ruediger. **Stereo-based 6D Object Localization for Grasping with Humanoid Robot Systems. Conference on Intelligent Robots and Systems.** San Diego CA, Estados Unidos, Oct. 29 - Nov.2, 2007

Bradski, Gary; Kaehler, Adrian. **Learning OpenCV**. O'Reilly Media, Sebastopol CA Estados Unidos de América, 2008.

Brown, Duane. **Close-range camera calibration**. Photogrammetric Engineering (p. 855–866), 1971.

Canny, John. **A computational Approach to Edge Detection**. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-8(6):679–698, Nov, 1986.

Cyganek, B.; Siebert, J.P. **An Introduction to 3D Computer Vision Techniques and Algorithms**. First Edition, John Wiley & Sons Inc., The Atrium, Southern Gate, Chichester, W. Sussex, UK, 2009

Grassi, M.V. **Desenvolvimento e aplicacao de um sistema de visao para robo industrial de manipulacao.** Disertación de potgrado, Universidade Federal Do Rio Grande do Sul, Programa de Postgrado en Ingeniería Mecánica, Porto Alegre, Brasil, 2005

Heikkila, Janne; Silven, Olli. **A four-step camera calibration procedure with implicit image correction.** Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (p. 1106), 1997.

Hirschmüller, Heiko. **Improvements in Real-Time Correlation-Based Stereo Vision**. Centre for Computational Intelligence Montfort University, Leicester, Reino Unido, 2001.

Hu, Ming-Kuei †. **Visual Pattern Recognition by Moment Invariants**. IRE Transactions on information theory, 1962.

Hu, Yingbai; Li, Zhijun; Li, Guanglin; Yuan, Peijiang; Yang, Chenguang; Song, Rong. **Development of Sensory-Motor Fusion-Based Manipulation and Grasping Control for a Robotic Hand-Eye System**. Presentación en Journal IEE, miembros IEEE, 2168-2216, 2015.

Kanellakis, Christoforos; Kyritsis, George; Tsilomitrou, Ourania; Manesis, Stamatis. **A low-cost stereoscopic uP-based vision system for industrial light objects grasping**. 23rd Mediterranean Conference on Control and Automation, Torremolinos, España, Junio 16-19, 2015.

Karaoguz, Cem; Dankers, Andrew; Rodemann, Tobias; Dunn, Mark. **An analiysis of depth Estimation within Interaction Range.** International Conference on Intelligent Robots and Systems, Taipei, Taiwan, Oct. 18-22, 2010.

Leonardelli, Pablo. **Uso de Diagrama de Blocos Funcionais como Linguagem de Programação para Robô Cilindrico de 5 Graus de Liberdade.** Disertación de postgrado, Universidade Federal Do Rio Grande do Sul, Programa de Postgraduación en Ingeniería Mecánica, Porto Alegre, Brasil, junho 2015.

Luu, Trang Hieu; Tran, Thanh Hung. **3D Vision for Mobile Robot Manipulator on Detecting and Tracking Target**. 15th Conference on Control, Automation and Systems, Busan, Korea, Oct. 13-16, 2015.

Medina, Betania. **Sistema de visao computacional aplicado a um robo cilíndrico acionado pneumaticamente.** Disertación de postgrado, Universidade Federal Do Rio Grande do Sul, Programa de Postgraduación en Ingeniería Mecánica, Porto Alegre, Brasil, febrero 2015.

Missiaggia, Leonardo. **Planejamento otimizado de trajetória para um robô cilíndrico acionado pneumáticamente.** Disertación de potgrado, Universidade Federal Do Rio Grande do Sul, Programa de Postgraduación en Ingeniería Mecánica, Porto Alegre, Brasil, febrero 2014.

Natarajan, Saravana.; Durrant, Danijela; Leu, Adrian; Gräser, Axel. **Robust Stereo-Vision Based 3D modelling of Real-World Objects for Assistive Robotic Applications**. International Conference on Intellligent Robots and Systems, San Francisco CA, Estados Unidos, Sept. 25-30, 2015.

IFR, International Federation of Robotics: **http://www.ifr.org/industrial-robots/statistics/**. Acceso marzo de 2016

IITD, Department of Computer Science and Engineering at Indian Institute of Technology: http://www.cse.iitd.ernet.in/~pkalra/col783/canny.pdf. **Canny Edge Detection Resume.** Acceso septiembre 2016.

MathWorks: **https://www.mathworks.com/**. Acceso enero de 2017

Pérez, Luis; Rodríguez, Iñigo; Rodríguez, Nuria; Usamentiaga, Rubén; García, Daniel. **Robot Guidance Using Machine Vision Techniques in Industrial Environments: A Comparative Review.** www.mdpi.com/journal/sensors, Sensors 16,335, España, marzo 2016.

Powell, Victor. Online: **Explained Visually Project**. http://setosa.io/ev/image-kernels/. Acceso septiembre 2016.

Rusu, Radu; Holzbach, Andreas; Diankov, Rosen; Bradski, Gary; Beetz, Michael. **Perception for Mobile Manipulation and Grasping using Active Stereo, 9th International Conference on Humanoid Robots**. Paris, Francia. Dic. 7-10, 2009

Sarmanho, Carlos. **Desenvolvimento de um robô pneumático de 5 graus de liberdade com controlador não linear com compensação de atrito.** Tesis de doctorado, Universidade Federal Do Rio Grande do Sul, Programa de Postgraduación en Ingeniería Mecánica, Porto Alegre, Brasil, septiembre 2014.

Sobel, I.; Feldman, G. **A 3x3 Isotropic Gradient Operator for Image Processing**. Presentado para el Stanford Artificial Intelligence Project (SAIL), 1968.

Spong, Mark; Hutchinson, Seth; Vidyasagar, M. **Robot Modeling and Control**. First Edition, John Wiley & Sons Inc., Hoboken NJ USA, 2006.

Zhang, Zhengyou**. A flexible new technique for camera calibration**. IEEE Transactions on Pattern Analysis and Machine Intelligence (p. 1330–1334), 2000.

Zisserman, Andrew; Hartley, Richard. **Multiple View Geometry in Computer Vision**. Cambridge University Press, ISBN:0521540518, segunda edición, 2004.