



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA.
ESCUELA DE INGENIERÍA ELECTRÓNICA CONTROL Y REDES
INDUSTRIALES.

"EVALUACIÓN DE ALGORITMOS DE TRACKING 3D PARA LA
SIMULACIÓN DE UN BRAZO ROBÓTICO, MEDIANTE KINECT".

TESIS DE GRADO.

Previo a la obtención de título:

"INGENIERO EN ELECTRÓNICA, CONTROL Y REDES INDUSTRIALES"

AUTOR: LUIS GUIDO ILBAY LLANGARÍ.

TUTOR: ING. MSC. FRANKLIN SAMANIEGO

Riobamba-Ecuador

2015

A Dios, por haberme regalado un motivo de superación y una oportunidad de lograrlo. A mis hermanas, el soporte y fin emocional de este y todos mis proyectos. A mi hermano, por la motivación que su presencia gesta en mí.

A mis tíos que han estado conmigo toda la vida y han puesto su atención y esperanza en mi carrera.

A mis amigos.

A mi tutor de tesis, todas las gracias por su tiempo invertido en este trabajo.

Nunca han dicho lo mucho que dejaron
de soñar mientras yo soñaba. Y más
noble aún el deseo de intentarlo
olvidar. Anega el orgullo en sus
nombres míos. Y serán siempre la
tierra pequeña que un día se desesperó
por retoñar. Más la mente callada no
dejará sin fuerzas sus manos, y si aún
después llueve, habrá un techo cercano,
que para vuestro bien, perdurará.

A Luis y Luz. Mis padres.

FIRMAS DE RESPONSABLES Y NOTA

NOMBRE **FIRMA** **FECHA.**

Ing. Gonzalo Samaniego.Ph.D.

DECANO DE LA FACULTAD
DE INFORMÁTICA Y ELECTRÓNICA.

Ing. Alberto Arellano.

DIRECTOR DE LA ESCUELA DE
INGENIERÍA ELECTRÓNICA, EN
CONTROL Y REDES INDUSTRIALES.

Ing. Franklin Samaniego.

DIRECTOR DE TESIS.
.....

Ing. Alberto Arellano.

MIEMBRO DEL TRIBUNAL
.....

COORDINADOR
.....

SISBIB - ESPOCH.

NOTA DE LA TESIS:

TEXTO DE RESPONSABILIDAD

"Yo, **LUIS GUIDO ILBAY LLANGARÍ**, soy responsable de las ideas, doctrinas y resultados, expuestos en esta tesis; y, el patrimonio intelectual de la tesis de grado pertenece a la **ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**".

.....

Luis Guido Ilbay Llangarí

ÍNDICE DE ABREVIATURAS

CMOS	Semiconductor Complementario de Óxido Metálico.
cm	centímetros.
ESPOCH	Escuela Superior Politécnica de Chimborazo.
fps	Frames Por Segundo.
GDL	Grados de Libertad.
HSV	Hue, Saturación, Value.
IR	Infrarrojo.
m	metros.
mW	mili Watts.
mm	Milímetros.
PC	Computador Personal.
RGB	Red, Green, Blue.
ROI	Región de Interés.
V.A.	Visión Artificial.
3D	Tres Dimensiones.
2D	Dos Dimensiones.

ÍNDICE GENERAL.

AGRADECIMIENTO.

DEDICATORIA.

FIRMAS DE RESPONSABLES Y NOTA.

RESPONSABILIDAD DEL AUTOR.

ÍNDICE DE ABREVIATURAS.

ÍNDICE GENERAL.

INTRODUCCIÓN.

1	MARCO REFERENCIAL.....	33 -
1.1	ANTECEDENTES.....	33 -
1.2	JUSTIFICACIÓN DEL PROYECTO DE TESIS.....	19 -
1.3	OBJETIVOS.	21 -
1.3.1	OBJETIVO GENERAL.....	21 -
1.3.2	OBJETIVOS ESPECÍFICOS	21 -
1.4	HIPÓTESIS.....	21 -
2	MARCO TEÓRICO.	27 -
2.1	ESTUDIO DEL ARTE	27 -
2.2	VISIÓN ARTIFICIAL	26 -
2.2.1	IMAGEN DIGITAL.	28 -
2.2.2	ADQUISICIÓN DE IMÁGENES.	31 -
2.2.3	PROCESAMIENTO DE IMÁGENES.....	32 -
2.2.4	HISTOGRAMA DE UNA IMAGEN.....	36 -
2.2.5	SEGMENTACIÓN.....	39 -
2.2.6	ILUMINACIÓN DE LA ESCENA.	41 -
2.3	BRAZO ROBÓTICO.....	42 -
2.3.1	ESPACIO DE TRABAJO.	43 -

2.3.2	CINEMÁTICA DEL ROBOT	- 44 -
2.4	SOFTWARE.	- 50 -
2.4.1	LINUX UBUNTU 14.04 LTS.	- 50 -
2.4.2	LENGUAJE C++.....	- 50 -
2.4.3	OPENCV.	- 51 -
2.4.4	CMAKE.....	- 52 -
2.4.5	MATLAB.....	- 52 -
2.5	KINECT.....	- 55 -
2.5.1	MENÚ DE COMPONENTES.....	- 56 -
2.5.2	VISIÓN DEL KINECT.	- 57 -
2.5.3	LIBFREENECT.....	- 60 -
2.6	TRACKING DE OBJETOS.....	- 61 -
2.6.1	ALGORITMOS DE TRACKING DE OBJETOS.....	- 62 -
2.7	RESUMEN COMPARATIVO.	- 74 -
3	SELECCIÓN DEL ALGORITMO MAS ADECUADO.....	- 26 -
3.1	INTRODUCCIÓN.	- 26 -
3.2	MÉTODOS USADOS.	- 76 -
3.2.1	MEDICIÓN.	- 77 -
3.3	DESARROLLO DE LAS APLICACIONES.....	- 78 -
3.4	TRATAMIENTO DE LA VISIÓN DEL KINECT.	- 79 -
3.5	OBTENCIÓN DEL DATO DE PROFUNDIDAD.....	- 80 -
3.5.1	FÓRMULA MATEMÁTICA DE TRANSFORMACIÓN.	- 82 -
3.6	INSTALACIÓN DEL SOFTWARE REQUERIDO.	- 84 -
3.6.1	PROCESO INSTALACIÓN DE LIBFREENECT.....	- 84 -
3.6.2	INSTALACIÓN DE OPENCV	- 85 -
3.7	PRUEBA DEL ALGORITMO CAMSHIFT.....	- 87 -
3.7.1	RESUMEN DE RESULTADOS	- 90 -

3.8	PRUEBA DE ALGORITMO DE COINCIDENCIA DE COLORES.....	- 91 -
3.8.1	TABLA DE RESULTADOS.....	- 94 -
3.8.2	CONCLUSIONES	- 96 -
3.9	PRUEBA DE ALGORITMO DE UMBRALIZACIÓN.....	- 96 -
3.9.1	PROCESO DE IMPLEMENTACIÓN.....	- 96 -
3.9.2	MEDICIÓN RESULTADOS.	- 101 -
3.9.3	CONCLUSIONES DE RESULTADOS.....	- 103 -
3.10	COMPROBACIÓN DE LA HIPÓTESIS.	- 103 -
3.10.1	ESTUDIO COMPARATIVO DE RESULTADOS.....	- 104 -
4	SIMULACIÓN DEL BRAZO ROBÓTICO.....	- 112 -
4.1	ESPACIO REAL DE TRABAJO PROYECTADO.	- 112 -
4.1.1	CALIDAD DE INFORMACIÓN.....	- 112 -
4.1.2	PROPÓSITOS DE LA INVESTIGACIÓN.....	- 113 -
4.2	ESTUDIO MATEMÁTICO - GEOMÉTRICO DEL BRAZO ROBÓTICO	- 113 -
4.2.1	GRÁFICAS DE RESULTADOS.	- 117 -
4.3	SIMULACIÓN DEL BRAZO ROBÓTICO EN TIEMPO REAL.	- 119 -
4.3.1	ESPACIO REAL DE VISIÓN DEL KINECT	- 119 -
4.3.2	ESLABONES DEL BRAZO ROBÓTICO.....	- 122 -
4.3.3	PROGRAMACIÓN.....	- 122 -
4.3.4	RESULTADO FINAL.....	- 123 -

CONCLUSIONES

RECOMENDACIONES

RESUMEN

ABSTRACT

BIBLIOGRAFÍA

ÍNDICE DE TABLAS.

TABLA 2-1 :TABLA COMPARATIVA DE LOS ALGORITMOS.....	- 74 -
TABLA 3-1 : MEDIDAS TOMADAS POR COINCIDENCIA DE COLORES.	- 94 -
TABLA 3-2: RESULTADOS PRUEBA ALGORITMO DE UMBRALIZACIÓN.-	101
-	
TABLA 3-3: PRUEBAS ALGORITMO CAMSHIFT	- 104 -
TABLA 3-4. PRUEBAS ALGORITMO UMBRALIZACIÓN	- 106 -
TABLA 3-5: PRUEBA ALGORITMO DE COLORES	- 109 -

ÍNDICE DE ILUSTRACIONES.

FIGURA 2-I : PROCESO GENERAL DE UN SISTEMA DE VISIÓN ARTIFICIAL.	- 27 -
FIGURA 2-II: LONGITUD DE ONDA DE COLORES VISIBLES.....	- 29 -
FIGURA 2-III: FORMATO DE IMAGEN HSV	- 31 -
FIGURA 2-IV: DILATACIÓN BINARIA.....	- 33 -
FIGURA 2-V: EROSIÓN BINARIA	- 34 -
FIGURA 2-VI: APERTURA.....	- 35 -
FIGURA 2-VII: IMAGEN TRAS PROCESO DE CIERRE.....	- 35 -
FIGURA 2-VIII : IMAGEN EN ESCALA DE GRISES.	- 37 -
FIGURA 2-IX: HISTOGRAMA DE LA IMAGEN.....	- 38 -
FIGURA 2-X: BRAZO ROBÓTICO TIPO SCARA	- 42 -
FIGURA 2-XI: ESPACIO DE TRABAJO DE UN BRAZO ROBÓTICO.....	- 43 -
FIGURA 2-XII: DIFERENCIA CINEMÁTICA INVERSA Y DIRECTA	- 44 -
FIGURA 2-XIII: BRAZO ROBÓTICO 3 GRADOS DE LIBERTAD.....	- 46 -
FIGURA 2-XIV: RESOLUCIÓN BRAZO 3 GDL.....	- 47 -
FIGURA 2-XV: BRAZO ROBÓTICO 4 GRADOS DE LIBERTAD.....	- 48 -
FIGURA 2-XVI: INSTALACIÓN C++ EN UBUNTU.....	- 51 -
FIGURA 2-XVII : KINECT XBOX 360.....	- 55 -
FIGURA 2-XVIII: PROYECCIÓN DEL INFRARROJO DEL KINECT.....	- 58 -
FIGURA 2-XIX: RANGO NOMINAL DE VISIÓN DE PROFUNDIDAD DEL KINECT.....	- 59 -
FIGURA 2-XX: IMAGEN RGB Y DEPTH EN AUSENCIA DE LUZ.....	- 59 -
FIGURA 2-XXI : LOGO OPEN KINECT	- 60 -
FIGURA 2-XXII: DIAGRAMA DE ALGORITMO CAMSHIFT.	- 65 -
FIGURA 2-XXIII: DIAGRAMA DE ALGORITMO DE UMBRALIZACIÓN.	- 69 -
FIGURA 2-XXIV: DIAGRAMA ALGORITMO POR COINCIDENCIA DE COLORES.....	- 72 -
FIGURA 3-I: DESARROLLO APLICATIVO DE LAS EVALUACIONES.....	- 79 -
FIGURA 3-II: DOBLE VISIÓN DEL KINECT.	- 80 -
FIGURA 3-III : BITS DE INFORMACIÓN DEPTH	- 81 -
ILUSTRACIÓN 3-IV : RANGO DE BITS - PROFUNDIDAD DEL KINECT	- 81 -
ILUSTRACIÓN 3-V : DATOS BINARIOS PUROS DEPTH DEL KINECT	- 82 -

FIGURA 3-VI: SYNAPTIC DE UBUNTU 14.04.	- 84 -
FIGURA 3-VII : INSTALACIÓN DE PAQUETES NECESARIOS PARA EL OPENCV	- 85 -
FIGURA 3-VIII: COMPILACIÓN DEL PAQUETE OPENCV	- 85 -
FIGURA 3-IX: CONFIGURACIÓN DEL OPENCV.	- 85 -
FIGURA 3-X: CONFIGURACIÓN DE LA RUTA DE LAS LIBRERÍAS INSTALADAS.	- 86 -
FIGURA 3-XI: CÓDIGO DE GENERACIÓN DE BASH.	- 86 -
FIGURA 3-XII: CONFIGURACIÓN FINAL DEL PKG_ CONFIG_PATH.	- 86 -
FIGURA 3-XIII: INIZIALIZACION Y DEFINICIÓN DE PARÁMETROS	- 87 -
FIGURA 3-XV: CÓDIGO DE LA OBTENCIÓN DEL ROI.	- 88 -
FIGURA 3-XIV: OBTENCIÓN DE LA IMAGEN.	- 88 -
FIGURA 3-XVII: HISTOGRAMA DEL OBJETO SEGUIDO.	- 89 -
FIGURA 3-XVI: HISTOGRAMA DE LA IMAGEN Y EL OBJETO.	- 89 -
FIGURA 3-XVIII: OBJETO ENCONTRADO.	- 90 -
FIGURA 3-XIX: INICIALIZAR LA CÁMARA Y EL SENSOR.	- 91 -
FIGURA 3-XX: ESTABLECIMIENTO DEL COLOR.	- 91 -
FIGURA 3-XXI: COMPARACIÓN DEL PIXELADO POR COLOR.	- 92 -
FIGURA 3-XXII: CALCULAR LA POSICIÓN DEL OBJETO.	- 92 -
FIGURA 3-XXIII: SEGUIMIENTO DEL OBJETO.	- 93 -
FIGURA 3-XXIV: DISTANCIA REAL, OBTENIDA Y ERROR DE ÉL ALGORITMO.	- 95 -
FIGURA 3-XXV: DISTANCIA REAL VS ERROR ABSOLUTO.	- 95 -
FIGURA 3-XXVI: INICIALIZACIÓN DE LA CÁMARA Y EL SENSOR.	- 96 -
FIGURA 3-XXVII : LLAMADA A LOS DATOS DE LOS DOS VIDEOS.	- 97 -
FIGURA 3-XXVIII: IMAGEN DEPTH VS RGB	- 97 -
FIGURA 3-XXIX: TRACKBAR DE CONTROL DE PARÁMETROS.	- 97 -
ILUSTRACIÓN 3-XXX: TRACKBARS DE LOS PARÁMETROS.	- 98 -
FIGURA 3-XXXI : CONVERSIÓN DE FORMATO DE IMAGEN.	- 98 -
FIGURA 3-XXXIII : CÁLCULO DE LA POSICIÓN DEL OBJETO.	- 99 -
FIGURA 3-XXXII : IMAGEN UMBRALIZADA.	- 99 -
FIGURA 3-XXXIV : CÓDIGO CONDICIÓN DE MUESTREO EN EL DEPTH-	100
-	
FIGURA 3-XXXV : DISTANCIA REAL VS DISTANCIA OBTENIDA.	- 102 -

FIGURA 3-XXXVI: MEDICIÓN REAL VS ERROR ABSOLUTO.....	- 103 -
FIGURA 4-I : ESTABLECIMIENTO DE PARÁMETROS DEL MOVIMIENTO.-	114
-	
FIGURA 4-II : PUNTO INICIAL Y ESPACIO DE MOVIMIENTO.....	- 115 -
FIGURA 4-III : OBTENCIÓN DEL ÁNGULO DE GIRO.	- 115 -
FIGURA 4-IV : ÁNGULOS DE TODOS LOS ESLABONES.	- 116 -
FIGURA 4-VI : GRÁFICA DEL BRAZO EN MOVIMIENTO.	- 117 -
FIGURA 4-V : PUNTOS EN EL ESPACIO DE LAS ARTICULACIONES	- 117 -
FIGURA 4-VII: MOVIMIENTO EN EL PLANO Y,Z.	- 118 -
FIGURA 4-VIII : MOVIMIENTO EN EL PLANO Y,X.....	- 118 -
FIGURA 4-IX: VISTA DEL BRAZO EN LOS PLANOS (X,Y,Z)	- 119 -
FIGURA 4-X : VISIÓN HORIZONTAL DE LA CÁMARA KINECT.....	- 120 -
FIGURA 4-XI : VISIÓN VERTICAL DEL KINECT.....	- 121 -
FIGURA 4-XII : LIBRERÍAS NECESARIAS PARA TRIGONOMETRÍA.	- 122 -
FIGURA 4-XIII : IMAGEN FINAL BRAZO ROBÓTICO.	- 124 -

INTRODUCCIÓN.

La visión artificial es una rama de la inteligencia artificial, que por sus prestaciones y su potente versatilidad en la implementación de aplicaciones en muchos campos de la ciencia, ha ido ganando su espacio exclusivo dentro del desarrollo actual de la tecnología y las mejoras informáticas.

Desde el nacimiento mismo de la visión artificial esta busca una sola cosa: entender con precisión una escena captada. Entonces las aplicaciones se extienden por casi todas las ciencias existentes. El progreso de la visión artificial debe asemejarse a la del humano.

Una de las aplicaciones clásicas de esta rama de la ciencia es el seguimiento de un objeto. Es cierto que para el humano prestar atención a un objeto específico no representa mayor esfuerzo. De hecho, basado en su pericia y experiencia puede incluso apuntar medidas de distancia entre su punto de observación y el objeto sabiendo que estas medidas no serán exactas. Es allí cuando un sistema de visión por computador puede adelantar al ser humano en este trabajo peculiar.

El crecimiento de esta especialidad va de la mano con las innovaciones tecnológicas tanto de hardware como de software. Podríamos citar desde cámaras con mas resolución hasta librerías de visión artificial más potentes y con mejores funcionalidades.

El tracking en tres dimensiones hoy en día debe arrojar con mucha precisión la posición en tiempo real y en los ejes (X,Y,Z). El verdadero problema fue siempre el dato de profundidad de la escena pues las cámaras de décadas pasadas no estaban preparadas para lograrlo y las que ahora lo están, son inaccesibles para los investigadores de poco presupuesto.

En el 2011, al abrirse al público un controlador libre para computador del sensor KINECT, lanzado por Microsoft, se dio un gran salto en el avance de cámaras aplicadas a la visión artificial. Las posibilidades de estudio con este dispositivo son variadas y de una cobertura realmente atractiva.

Pero esta misma cantidad desmesurada de investigación particular provoca que a la hora de escoger un método, una librería o un algoritmo para un trabajo propio, sea una acción de muchas controversia. No podríamos asegurar que herramienta es la más adecuada para todos los trabajos, ya que esto depende de diversos factores y la misma subjetividad del investigador.

Llegado a este punto un investigador siempre anhela que su trabajo sea realizado con las herramientas que mejor resultados arrojen. Por lo que muchos optan primero en comparar o evaluar las distintas herramientas para asegurarse que su elección.

En la actualidad la potencia de los ordenadores ya están abiertas al público por un precio elevado, pero muchas veces costeable. Esto abre las posibilidades de que los trabajos propios tengan resultados fiables a nivel regional y quizá a nivel mundial.

En el presente trabajo se ha desarrollado el "tracking" de un objeto con el algoritmo idóneo, usando como único sensor al Kinect y optando por el Open Source como plataforma de desarrollo. El proyecto ha puesto hincapié en escoger un algoritmo de entre tantos que tenga las prestaciones necesarias y que se adapte al Kinect de la mejor manera.

Muchas herramientas se han desarrollado para el servicio de la visión artificial. Las más importantes son las librerías orientadas. Aunque existen una cantidad considerable de librerías una a sobresalido entre estas por su versatilidad, potencia y aplicaciones. Además de tener una gran documentación y de que es libre para el público. OpenCV.

Entonces se tienen que unir en un punto investigativo todos los recursos actuales de software y hardware. Pero los estudios no pueden quedar solamente en eso. Las aplicaciones a empresas, industrias o campos tecnológicos de investigación es donde cualquier novedad científica debe terminar. Una vez obtenida la respuesta ante la intriga anterior, la investigación ha procedido a generar un código que realice el "tracking" en tres dimensiones de un objeto, de tal forma que el usuario pueda conocer cuál es la

coordenada tridimensional del mismo. Para poder demostrar la fiabilidad de estos resultados y la potencialidad del trabajo, se pretende simular un brazo robótico que pueda alcanzar al objeto en tiempo real y que simule una realidad aumentada de la escena.

En orientación del campo industrial el siguiente trabajo ha propuesto un brazo robótico de 4 grados de libertad para poder alcanzar el objeto mencionado, se presentará un estudio geométrico del movimiento en el desarrollo de la aplicación.

Este documento pretende llegar a ser una herramienta de estudio sobre visión artificial y robótica para los entes académicos de todos los niveles de educación local y nacional, y que a partir de la misma, se puedan generar muchos trabajos posteriores que revelen la calidad de estudio que tiene la institución educativa a la que pertenece.

CAPÍTULO I.

1 MARCO REFERENCIAL.

1.1 ANTECEDENTES.

El tracking de objetos se ha venido desarrollando desde que la visión artificial existe en forma práctica. Pero que sea un objetivo antiguo o un trabajo ampliamente desarrollado no implica que sea fácil. Todo lo contrario, esta viene a ser una de las más complicadas búsquedas.

Por su necesidad y su gran requerimiento existen varias técnicas y algoritmos que se empeñan en hacer el resultado más real. Entre los inconvenientes que siempre se han encontrado podemos citar: requerimiento de hardware orientado y potente. Software robusto y con gran precisión en los errores.

El hardware de esta ciencia en general se reduce a pocas necesidades como: recoger una imagen y en ocasiones realizar un muestreo sobre la imagen. La primera se resolvió

hace ya muchos años y sin necesidad de que la ciencia lo pidiera sino mas bien el entretenimiento, de hecho, el KINECT, dispositivo utilizado para este trabajo, nace de la misma peculiaridad. Ventajosamente el entretenimiento del hombre vende más, entonces las cámaras han tenido un desarrollo próspero. Este es un punto clave para cuando la visión artificial nace, pues su problema ya solo radica en entender y procesar la imagen a través de software.

Con respecto al software, y como se había mencionado anteriormente, OpenCV acapara la preferencia de los investigadores, profesores y estudiantes para utilizarlo como herramienta principal.

La misma librería tiene muchos algoritmos desarrollados y la ventaja de poder desarrollar otros usando sus funciones. Pero es muy complicado decidir sobre cuál será el mejor algoritmo para el tracking si antes hacer un estudio importante.

Este mismo desarrollo ha provocado que los sensores más precisos ya no sean suficientes a la hora de aportar con datos acerca de un objeto. Hoy por hoy se necesita algo más parecido a la percepción humana. Diversas técnicas se han venido gestando a partir de esta realidad y una de las más versátiles ha sido la visión artificial, ya que por obvias razones, el sentido de la vista es el más necesario para cualquier tipo de labor.

La visión artificial puede ser definida como una técnica o un conjunto de técnicas que con el hardware adecuado puede obtener información de imágenes en forma digital a partir de la cual se puede generar respuestas y órdenes correctas.

Entonces debemos entender que la visión artificial está totalmente entregada al procesamiento de imágenes y que es imperativo recurrir a otra aplicación para poner en uso los datos recolectados por esta ciencia.

Si bien es cierto la visión artificial ha sido siempre un subtema de la Inteligencia Artificial, tiene propósitos propios y exclusivos. Entre ellos mencionaremos: detección de objetos, mapeo de escenarios, generar resultados sobre una consulta en la imagen, realizar chequeos sobre objetos sin necesidad de contacto directo, realizar inspecciones sobre piezas de diversas características, entre otros. Las decisiones que se tomen con estos datos son parte de la inteligencia artificial.

Además tiene aplicaciones industriales como:

- Determinación de la posición de un objeto en el espacio.
- Establecimiento de relaciones entre posicionamiento de objetos
- Determinación de las coordenadas importantes de un objeto
- Mediciones angulares
- Mediciones tridimensionales.

Como hemos podido observar las posibilidades son inmensas. Sin embargo en nuestro país hace falta mucho trabajo investigativo sobre este tema. Hay, entonces, una oportunidad irreplicable de acercarnos al avance intelectual que el mundo está viviendo día con día.

1.2 JUSTIFICACIÓN DEL PROYECTO DE TESIS

Los avances tecnológicos como industriales siempre necesitan de detalles más precisos y fiables de parte de los sensores usados, no representa mayor problema para las grandes compañías. Sin embargo, aceptando la realidad industrial y tecnológica de nuestro país, todo esfuerzo por alcanzar investigaciones tecnológicas de alto nivel es bien recibido. La visión artificial tiene que ser sin duda, el nuevo peldaño en el avance de nuestra sociedad. Su aplicación se extiende por todas las ramas de la ciencia, no puede tener el procesamiento de imágenes tan colosal que tiene el ser humano pero puede ser más preciso con sus propios avances.

En vista de que el tema es muy amplio y el desarrollo en la ESPOCH tiene los fundamentos en investigación necesarios el presente trabajo tiene que no solo dejar un documento de estudio, sino además dar oportunidad a muchas aplicaciones con las herramientas entendidas en este trabajo.

Se pretende obtener importantes avances acerca del tema, problemas resueltos, una aplicación funcional que a la vez sienta un nuevo precedente para futuras investigaciones y que en un tiempo no muy lejano permita que la visión artificial acabe copando la mayoría de los procesos industriales y tecnológicos de nuestro ámbito laboral.

Ahora, si con el pasar de la tecnología hemos tenido buenas cámaras y demás recursos de hardware para el trabajo en visión artificial. ¿por qué nos proponemos realizar este proyecto con el Kinect?

En primer lugar porque es un recurso potente y de grandes prestaciones, no solo se compone de una cámara RGB sino también de una cámara infrarroja con las cuales se puede tener una imagen y a la vez calcular su profundidad con respecto a los objetos que la rodean. También influye el que sea una herramienta novel, que apenas se lanzo a fines del 2010 y que promete un desarrollo impresionante. Por último se notó que al ser un producto de inmensa popularidad tendrá una mejora constante y que cada vez ampliara mas sus horizontes por lo que elegirlo es ir a un trabajo con respaldo.

1.3 OBJETIVOS.

1.3.1 OBJETIVO GENERAL.

- Evaluar tres Algoritmos de Tracking 3D , aplicado a la simulación de un brazo robótico, usando “KINECT”.

1.3.2 OBJETIVOS ESPECÍFICOS

- Realizar el estudio del arte de la investigación actual del Kinect y los algoritmos de tracking de objetos.
- Definir el algoritmo más idóneo para el tracking de objetos usando Kinect.
- Desarrollar el seguidor de un objeto “Tracking 3D” que obtenga las coordenadas geométricas del objeto con el algoritmo elegido.
- Realizar un estudio matemático del Método Geométrico de un brazo robótico de 4 grados de libertad.
- Realizar la simulación de un brazo robótico que alcance el objeto en tiempo real.

1.4 HIPÓTESIS

"La evaluación de algoritmos de “Tracking 3D” permitirá elegir la opción más adecuada con la que, usando Kinect, se pueda simular un movimiento robótico que bajo las coordenadas geométricas obtenidas, pueda alcanzar el objeto".

CAPITULO II

2 MARCO TEÓRICO.

En el presente capítulo se desarrollarán los conceptos teóricos de las distintas herramientas ocupadas para la consecución nuestra investigación. Además, la actualidad del desarrollo de los avances de la Visión Artificial (en adelante V.A.) con Kinect tanto en el ámbito internacional como en el nacional y local.

2.1 ESTUDIO DEL ARTE

Desde el 2011 , cuando el controlador del Kinect fue liberado, las aplicaciones con el mismo se han ido incrementando y han ocupado desarrollos en distintos campos de la ciencia. Es cierto que los trabajos de gran presupuesto llaman la atención y estas le pertenecen a los países que siempre están en la testa de la tecnología. Empero también Latinoamérica se ha presentado en la actualidad con grandes proyectos de investigación.

Aunque, es justo, decir que el volumen de trabajo de aquellos países de primer mundo es superior al nuestro.

Al momento de la creación del SDK Kinect de Microsoft se pretendía que solo se use a esa, como plataforma de investigación para el dispositivo, dando de baja a las versiones libres existentes, sin embargo las bondades del Open Source han hecho que cualquier persona con un Kinect a la mano y conocimientos de programación pueda inventar aplicaciones usando controladores que no tienen nada que ver con Microsoft.

Hay que recalcar que también los controladores libres del Kinect no han dejado de desarrollarse a pesar del SDK de Microsoft. Algunos de ellos, como el Libfreenect aún tienen muchas utilidades por abarcar, pero su constitución actual tiene una gran capacidad. Otro controlador es el desarrollado por el Open NI, que también tiene muchos recursos óptimos, aunque en la actualidad este fue adquirido por una gran empresa y es incierto si se continuará su desarrollo.

Estos trabajos en general se orientan al control remoto de dispositivos con los movimientos humanos, exploración autónoma, robótica y tracking de objetos.

Entre algunos trabajos que han conseguido el renombre público podríamos mencionar:

Turtlebot.- Es un conocido robot que se moviliza con un robot iCreate y la visión del Kinect, conectado a la laptop. El código de este robot es abierto, así que los desarrolladores pueden modificarlo para intereses más específicos, una de las aplicaciones más conocidas del robot es servir como camarero. El robot se puede adquirir en la página oficial del mismo. <http://turtlebot.com/>

Nuevo asistente de quirófano.- Su propósito es evitar que al momento de consultar el expediente del paciente, el cirujano no deba usar las manos para evitar cualquier tipo de contagio. Esta labor la realiza con ayuda de los múltiples sensores del Kinect. Este proyecto resultó ganador en un concurso por impulsar el estudio del Kinect.

En Ecuador, la investigación con el Kinect, también ha tenido su apertura por parte de las tesis de grado y de educación de cuarto nivel. Aunque el número de trabajos no es tan grande como en las potencias mundiales, existen grandes proyectos que aprovechan

al máximo los recursos del Kinect y demuestran que el mismo puede trabajar en varios campos de estudio.

Entre algunos de estos, podríamos mencionar :

Seguridad.- SISTEMA DE SEGURIDAD INTELIGENTE BASADO EN RECONOCIMIENTO DE PATRONES MEDIANTE TECNOLOGÍA KINECT PARA RESTRINGIR EL ACCESO NO AUTORIZADO A CONSOLAS DE ADMINISTRACIÓN Y MONITOREO . Ayala Cajas César Andrés, Guerrero Idrovo Rosa Graciela

Educación.- "SISTEMAS DE EDUCACIÓN PARA NIÑOS DE 3 A 5 AÑOS, MEDIANTE UN ROBOT CONTROLADO POR EL SENSOR KINECT" ESPOCH, Rómulo Byron Ilvay Taday.

Controles Remotos.- “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TELEOPERACION PARA CONTROLAR UN ROBOT HUMANOIDE MEDIANTE UN SENSOR KINECT” ESPOCH, Diego Ñacato Ramiro Estrella.

Desarrollo de Controladores.- "DESARROLLO DE UN PROTOTIPO DE UN API PARA LA INTERACCIÓN DE UN USUARIO CON APLICACIONES CON CONTENIDO 3D UTILIZANDO KINECT", ESPOL, Andrés Estuardo Prieto López

Cabe recalcar que la gran mayoría de estos estudios usan el conocido SDK Kinect para Windows. Esta herramienta contiene potentes controladores y aplicaciones para el Kinect.

Pero contraria a esta esperanza los controladores libres se han seguido desarrollando, y muchos investigadores han puesto sus trabajos a la confianza de estos. Lastimosamente muchos han visto el trabajo con el Kinect como difícil si se lo hace con estas herramientas, así que optan por usar las herramientas ofrecidas por Microsoft que como ya hemos dicho, son muy completa y prácticas.

El estudio del impacto del Kinect en los brazos robóticos de muchos grados de libertad ha sido relegado a un segundo plano, debido principalmente a la gran versatilidad del

sensor para trabajar en cualquier ámbito de la vida y no solo los procesos industriales y científicos.

2.2 VISIÓN ARTIFICIAL

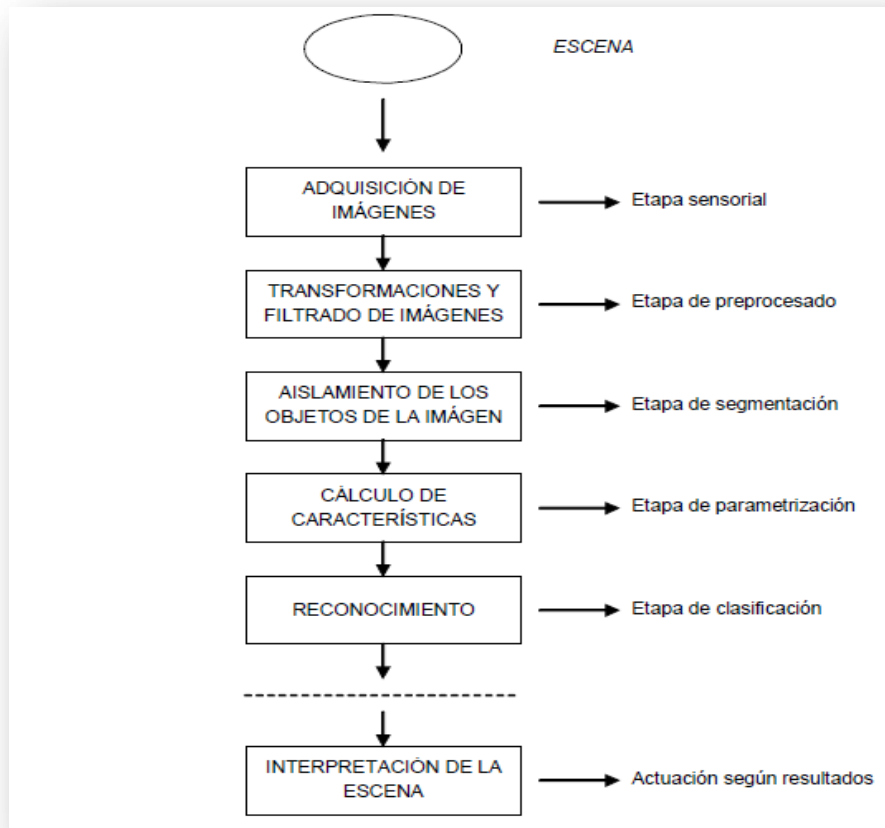
Según Nilsson, la visión por computador o V.A. es una rama de la ciencia que pertenece a la Inteligencia Artificial y que se centra en analizar la posibilidad de los sistemas dotados de un dispositivo capaz de captar la escena de su entorno. La V.A. tiene tal versatilidad que puede apoyar el desarrollo de varios campos y aplicaciones como reconocimiento de caracteres, interpretación de imágenes y escenas, robótica, medicina, etc.

La V.A. asocia hardware, software y también desarrollos teóricos con el fin de poder entender de la manera más humana posible una escena propuesta. Por lo que es un trabajo que demanda de muchos recursos de parte del ente aplicativo. Pero sus grandes prestaciones y su capacidad de acoplamiento en torno a tantas ciencia hace que la investigación sobre este campo sea muy rentable, y además llame la atención tanto de investigadores consagrados como de estudiantes.

Algunas aplicaciones importantes serían: automatización de una línea de ensamblaje, la inspección de circuitos para detectar defectos, teledetección para construir mapas de regiones no fácilmente accesible, comunicación mediante gestos del ordenador. Y estos abarcan sectores como: auxiliar del automóvil, farmacéutica, embotellado, embalaje etiquetado textil, papel, metalúrgica, semiconductores, cerámica, maderas, equipos electrónicos, etc.

En general cualquier trabajo de V.A. consta de dos fases: capturar la imagen y procesarla según nuestro objetivo. Sin embargo, un sistema general de V.A. se puede obtener del siguiente gráfico.

FIGURA 2-I : PROCESO GENERAL DE UN SISTEMA DE VISIÓN ARTIFICIAL.



FUENTE: Técnicas y algoritmos básico de visión artificial.

Este mismo proceso se detallará mas adelante con ejemplos puntuales, pero el gráfico representa la mayoría de los sistemas de V.A.

En cuanto a las herramientas para todos los proyectos de V.A. se necesitan los mismos elemento básicos. El primero es el que obtenga la imagen ya sea esta analógica o digital. La cámara obtiene la imagen y la deja lista para ser procesada. Algunas veces esto lo puede hacer la cámara automáticamente pero otras se necesita de software y de un código de pre procesado. Seguido de esto, se requiere en la parte de software, un lenguaje de programación y una librería orienta a la V.A.

Por último, y según la aplicación, se necesita una herramienta en donde mostrar y comprobar los resultados. Puede ser un robot, un terminal , hardware orientado o simplemente una pantalla en donde se puede notar una simulación.

La captación de la imagen es en realidad una tarea muy compleja que favorablemente fue resuelta hace ya mucho tiempo. Entonces lo que le resta a la V.A. es implementar los mejores procesos de interpretación sobre la imagen obtenida. Y aunque esta interpretación, con respecto al ojo humano, es un trabajo del cerebro, si lo aplicamos en un computador necesitaremos un procesador que analice pesos inmensos de información. Para hacer de esta tarea lo menos costosa se modela la escena con técnicas como: fondos homogéneos, variación en la iluminación con respecto a la necesidad y a los favores de hardware a disposición, diferenciar al objetos o objetos importantes de los innecesarios.

El resto del trabajo lo debe realizar el software por lo cual, gran parte de los trabajos en la V.A. está orientada a la eliminación del ruido en una imagen.

El trabajo de reprocesar una imagen tiene como base a algoritmos matemáticos que calculan diferentes características, desde nuevas intensidades de iluminación hasta diferencia de pixelado; estos procesos aunque son sencillos, conllevan mucho tiempo de cálculo según la resolución de la imagen. Esto es un gran problema para un ordenador que debe dar respuestas en tiempo real, por lo que se debe buscar la mejor forma de darle la menor carga de trabajo posible, de lo contrario resultaría un trabajo derrochador.

En la actualidad, debido a todos estos antecedentes, la V.A. aprovecha los procesadores de nueva generación que realizar varias acciones al mismo tiempo, las cámaras digitales, los sensores de precisión y muchas otras novedades que la tecnología ha dejado en nuestro tiempo.

2.2.1 IMAGEN DIGITAL.

Se define a una imagen como digital cuando su obtención y grabado lo realizó un aparato o una cámara electrónica, y se construye en unidades de pixel siendo estas finitas y medibles en forma matricial.

Las propiedades básicas de una imagen digital son:

Resolución.- se considera el número matricial de filas x columnas de pixeles que contiene a la imagen.

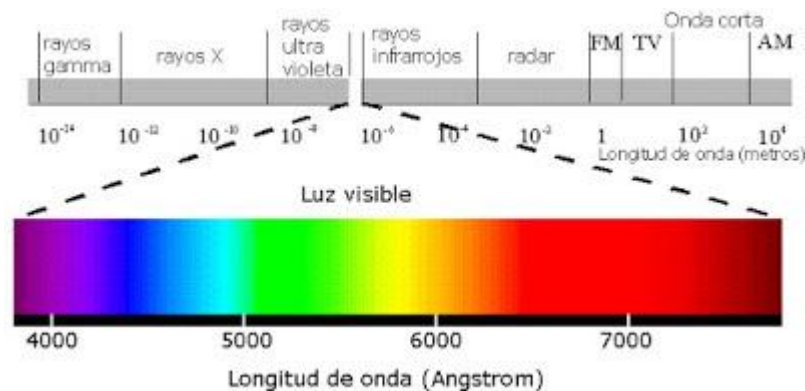
Definición.- Representa cuan nítida esta una imagen. Depende enteramente del numero de bits que se usa para representar un pixel. Aunque se puede representar de forma cuantitativa su cantidad también puede ser cualitativa según un ojo humano o condiciones de software.

2.2.1.1 FUNDAMENTO DE COLOR.

El color no es más que una sentido de percepción que tiene el humano sobre la luz reflejada en las cosas. De allí depende mucho la calidad y cantidad de luz que tiene la escena para la diferencia de los colores. Si un cuarto se llena de luz se puede entender cada uno de los reflejos en los objetos como colores distintos. Mientras que si la luz está totalmente ausente será imposible diferenciarlos, aún percibirlos.

Los colores que el humano puede percibir son tan solo una parte de la longitud de onda total que tiene la luz. Ya que hay frecuencias tan altas, o tan bajas que el ojo humano no lo puede percibir, pero que sin embargo si tienen incidencia sobre el objeto que las recibe. Estas ondas están en constante estudio y se cree posible grandes aplicaciones con ellas.

FIGURA 2-II: LONGITUD DE ONDA DE COLORES VISIBLES



FUENTE: <http://reflexionesfotograficas.blogspot.com/2011/01/el-espectro-visible-los-colores.html>

Existen muchos formatos o modelos de colores, los cuales han sido desarrollados con el mejoramiento del software y el hardware. Podemos mencionar: HSI, CMY y RGB.

El RGB es quizá es más utilizado ya que es sencillo de percibir y de maniobrar. Es el modelo que usan las cámaras y los sensores de color primarios.

Si percepción se basa en tres colores primarios: Rojo, Verde y Azul. De allí su nombre. El estudio indica que a partir de estos colores se puede obtener casi cualquier todo e intensidad de otro color. Incluso el Negro , que reflejaría la ausencia de los colores y el Blanco que significaría la presencia de los tres.

Para representar la cantidad de cada color de forma cuántica se usa un rango igual para todos los colores de 0 a 255.

Ejemplo: Rojo= 255 , Verde= 65, Azul= 225.

El resultado es el color **amarillo**.

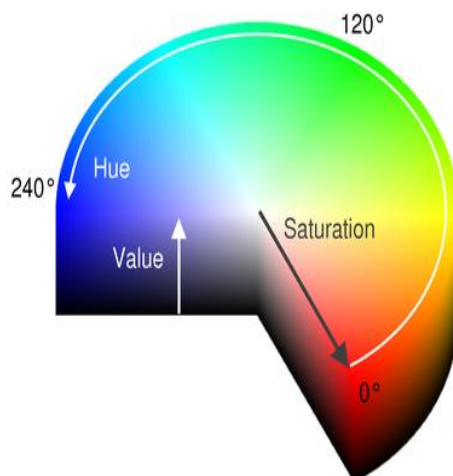
El formato HSV en cambio, es un formato de color que es una transformación no lineal del modelo base RGB. Su nombre deriva de las siglas: Hue, Saturation, Value, que significaría respectivamente:

MATIZ .- se presenta como un grado de ángulos cuyo valores están en un rango de 0-360 y con los grados se podría representar el color.

SATURACIÓN.- Detona como la distancia al eje básico de brillo blanco-negro. Su rango de medida está en 0-100 %, entonces cuanto más bajo es este valor la imagen tiende a ser más gris y con menos colorido.

VALOR .- Es la altura entre el blanco-negro. De la misma manera se puede medir entre 0-100, siendo el cero el negro.

FIGURA 2-III: FORMATO DE IMAGEN HSV



FUENTE: <http://doc.qt.digia.com/qq/qq26-adaptivecoloring.html>

2.2.2 ADQUISICIÓN DE IMÁGENES.

Es el proceso de captar una escena del mundo real y guardarlo ya de forma digital o analógica.

Las primeras imágenes adquiridas eran analógicas y su trabajo estaba reservado a verdaderos profesionales del arte, incluso llevaba mucho tiempo llegar a captarlas de forma correcta. Sin embargo, la V.A. hubiera tenido un espacio relativamente nulo debido a la dificultad de este proceso.

En la era digital, como en muchos otros ámbitos, los procesos se hicieron más sencillos y accesibles a todo el público. Hoy, con una cámara web de muy bajo costo se puede incursionar fácilmente en el mundo de la V.A.

Entre las cámaras más populares para la adquisición de imágenes se puede mencionar:

- Cámaras WEB
- Cámaras digitales.
- Kinect

La imagen digital obtenida se mide en píxeles . Es la unidad de medida del tamaño de la imagen, cada pixel se compone de una combinación de los tres colores principales: rojo, verde, azul, Cada pixel tiene una ubicación matricial en el espacio de la imagen.

2.2.3 PROCESAMIENTO DE IMÁGENES.

Para que cualquier imagen digital obtenida pueda ser estudiada se requiere que antes sea procesada. Este procesamiento se puede realizar en la cámara, si es que esta tiene la facultad de hacerlo, o de preferencia con un software vinculado a este trabajo.

El procesamiento básico consiste en acondicionar la imagen a formatos requeridos y aplicar filtros para eliminar ruidos y para tener un mejor enfoque de la escena, con ello harán falta menos recursos para poder analizarla a cabalidad.

En ocasiones, antes de un procesamiento de imágenes se debe pre procesar la imagen, para aumentar la fiabilidad de los resultados, podemos nombrar algunas operaciones del pre procesado como :

Inversión ,Adición y substracción, Producto o división por una constante, Logaritmo y Exponencial, Complementación, And, Or y Xor, Silicing, Clipping,Umbralización, Binarización.

2.2.3.1 TRANSFORMACIONES MORFOLÓGICAS.

Las transformaciones morfológicas cambian la forma y la estructura de los elementos de una imagen. Permiten modificar estas formas para separar unos objetos de otros, obtener contornos de objetos, volver simples a formas complejas, eliminar ruidos sobre objetos necesarios, etc.

Generalmente estas transformaciones se usan en imágenes previamente binarizadas (en blanco y negro) o también imágenes en niveles de gris.

2.2.3.1.1 DILATACIÓN BINARIA

La dilatación, también llamada “crecimiento”, “llenado”, “expansión” produce un efecto de engrosamiento en los bordes en el objeto. Haciendo que pequeños objetos que estén junto al mayor sean absorbidos y ya no se entiendan como ruido. Existen métodos para la dilatación, por ejemplo sumando los pixeles que rodea a uno central y si supera un umbral se pone a una.

Este efecto es muy valorado para aumentar el contorno de los objetos y unir líneas discontinuas.

FIGURA 2-IV: DILATACIÓN BINARIA.



FUENTE: COMPUTER VISION AND ROBOTIC GROUP.

2.2.3.1.2 EROSIÓN BINARIA.

Es la función dual de la dilatación, no significa que sea su función inversa. Dicho de otra forma, no significa que si a una imagen primero la dilatamos y luego la erosionamos, obtendremos la imagen original, estos procesos convierten de forma importante y definitiva la imagen. Si la dilatación expandía los bordes y contornos de los objetos, la erosión reduce los contornos de los objetos. Se usa para separar pequeños objetos unidos a los objetos más grandes. Igualmente que en la dilatación, existen muchos métodos para erosionar la imagen, uno de ellos se puede parecer mucho

a la Umbralización . En la siguiente imagen se puede notar que todos los pequeños objetos que no son reconocidos, son eliminados incluso los bordes los objetos restantes parecen más pálidos y formados.

FIGURA 2-V: EROSIÓN BINARIA



FUENTE : COMPUTER VISION AND ROBOTIC GROUP

2.2.3.1.3 APERTURA.

Consiste en usar una erosión y luego una dilatación. Como se dijo anteriormente, la imagen resultante no va a ser igual a la original. Si primero eliminamos los ruidos muy pequeños y luego afirmamos los objetos restantes nos quedarán objetos ensanchados pero sin mucho en ruido. Esta transformación es muy usada en aplicaciones como :

- Segmentación de objetos
- Descomposición de objetos en elementos más simples
- Extracción de formas determinadas.
- Elimina salientes estrechos.
- Separación objetos que no están demasiado pegados.
- Aumentar los agujeros que está dentro.

FIGURA 2-VI: APERTURA.



FUENTE : COMPUTER VISION AND ROBOTIC GROUP

2.2.3.1.4 CIERRE.

Se realiza primero una dilatación y luego una erosión. Y sus aplicaciones son parecidas a la transformación anterior. Incluso se llega a usar una después de otra para lograr una imagen totalmente libre de ruido y dado que el costo computacional de estos procesos son bajos, son una gran idea para que la segmentación sea de calidad.

FIGURA 2-VII: IMAGEN TRAS PROCESO DE CIERRE.



FUENTE : COMPUTER VISION AND ROBOTIC GROUP

2.2.4 HISTOGRAMA DE UNA IMAGEN.

Es una gráfica en un tipo de barras que representa la distribución de los datos de una imagen, muestra: acumulación o tendencia, la variabilidad o dispersión, y como están distribuidas las características de la imagen.

Le aporta al sistema algunas particularidades como :

Síntesis: Permite realizar un resumen digerible de los datos.

Análisis: analizar los datos pues se puede reconocer.

Capacidad de comunicación: comunica la información de forma clara y concisa y se permite procesarla en otro software o hardware .

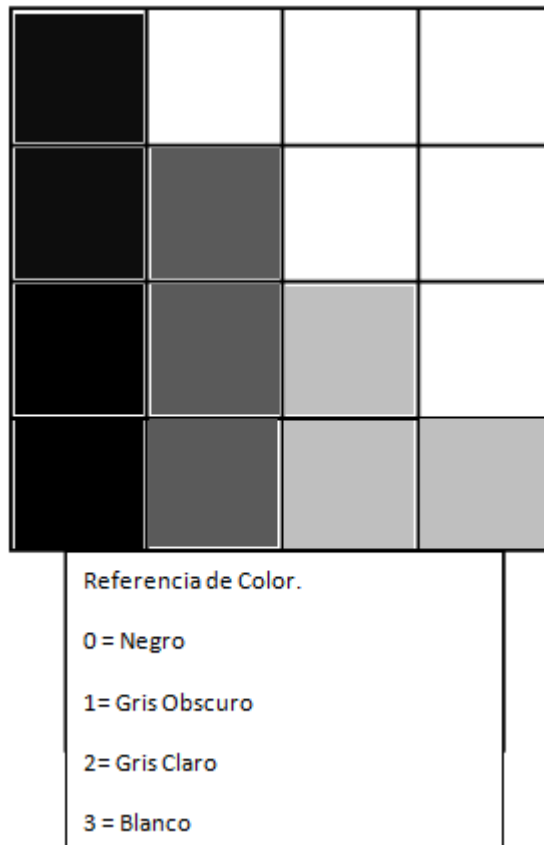
Los histogramas de imágenes en escala de grises son muy usados debido a su facilidad de comprensión. En general se componen de la siguiente manera: el eje horizontal representa los distintos tonos de gris desde el negro puro al blanco y el eje vertical representa el número de píxeles que contienen la imagen para cada tono.

Una imagen con el histograma bien distribuido es considerada como bien iluminada. Sin embargo, también se pueden estudiar otras situaciones comunes como:

- Tonos apagados
- Sombras o zonas observadas
- Sobreexposición y zonas quemadas
- Sombras pálidas
- Contraluz
- Distribución homogénea.

Para entender para entender de forma más clara a lo que se refiere el histograma de una imagen se presenta el siguiente ejemplo.

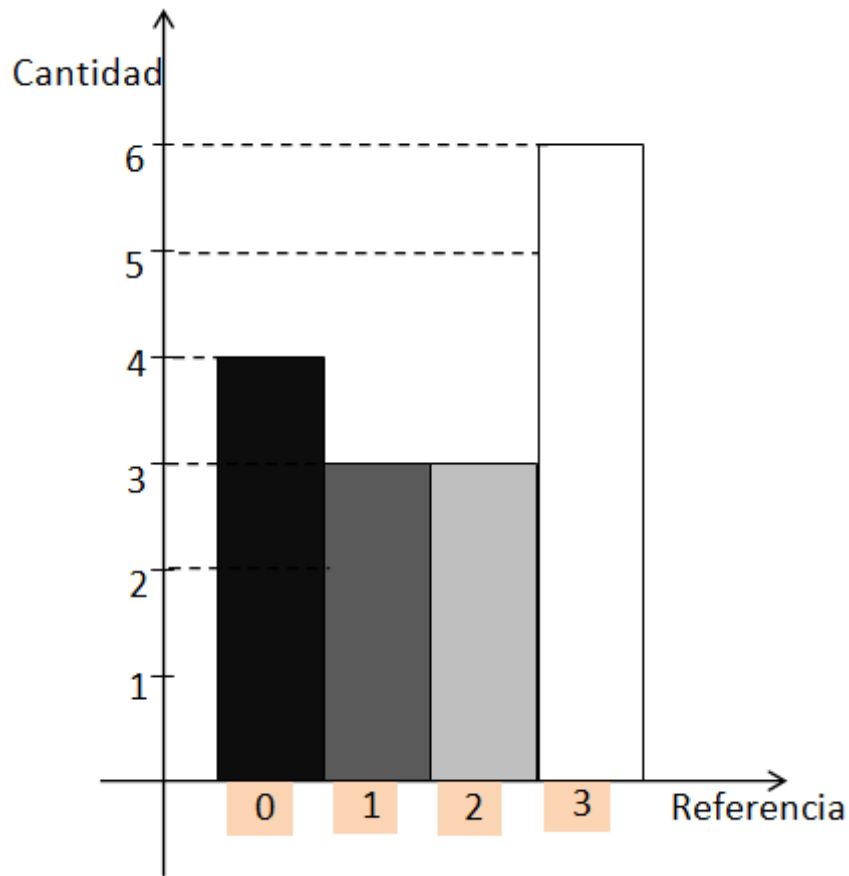
FIGURA 2-VIII : IMAGEN EN ESCALA DE GRISES.



FUENTE: Ilbay Llangarí Luis Guido.

La anterior imagen esta en escala de grises y no tiene demasiados tonos para poder estudiar en ella un histograma sencillo de diseñar y que pueda facilitar el entendimiento de su uso.

FIGURA 2-IX: HISTOGRAMA DE LA IMAGEN.



FUENTE : Ilbay Llangarí Luis Guido.

Podemos notar que el color negro tiene 4 unidades de color dentro de la imagen, los dos grises tienen tres, mientras que el color blanco tiene 6 unidades, dejando a la imagen ligeramente balanceada. En el análisis de una imagen de muchos tonos de grises, el histograma será una gráfica más extensa pero nos podrá ayudar a comprender cuál es la distribución de colores o de tonos de la imagen.

Muchas técnicas de segmentación se basan en el histograma para separar un objeto de su entorno.

2.2.5 SEGMENTACIÓN.

Segmentar una imagen consiste en generar zonas privadas de cada uno de los elementos que forman parte de la escena, diferenciando los objetos con respecto al fondo y a su entorno. En este proceso se pueden encontrar fondos complejos o sencillos siendo los últimos los usados para la investigación y el desarrollo ya que impulsa al resultado primario del estudio.

Una segmentación adecuada debe mostrarnos los objetos diferenciados y listo para el siguiente proceso, esto significa que cada grupo de pixeles debe pertenecer únicamente al objeto y separarse de su entorno.

La etapa de segmentación es crucial para el reconocimiento de formas siendo tan complicado según la complejidad de la escena. No existe relación entre la aplicación la complejidad de la escena ya que siempre se puede adecuar la escena para que no sea tan difícil de segmentar.

Se puede decir que la segmentación es la etapa verdadera del reconocimiento pues el resultado de esta etapa es tener los objetos perfectamente ubicado en la imagen resultante, dependiendo de la técnica utilizada.

2.2.5.1 UMBRALIZACIÓN.

En teoría es el método más simple ya que se basa en clasificar cada pixel de la imagen como parte o no del objeto.

Generalmente existen dos formas de concretar el umbralizado: *thresholding*, *histogramming*. En cualquiera de las dos se necesita trabajar sobre valores homogéneos de gris, para medir la intensidad de cada pixel.

En el primer caso (*thresholding*) si los objetos son más claros que el fondo son considerados parte del elemento, de lo contrario serían parte de lo que le resta de la imagen.

En el segundo caso(*histogramming*) se necesita realizar un estudio sobre el histograma de la imagen y se toma un rango a partir del cual el objeto es hallado y el resto de la gráfica no tiene importancia para el estudio.

2.2.5.2 SEGMENTACIÓN ORIENTADA A REGIONES.

La segmentación orientada a regiones trabaja con la conectividad de la imagen, que significa que para que un objeto sea reconocido como tal debe tener las condiciones necesarias: compartir características como color, textura, etc.; o , que sean los pixeles conectados dentro de una curva o de líneas de bordes.

2.2.5.3 BASADA EN BORDES.

Es una técnica de segmentación que acude a los bordes como diferenciadores de un objeto con el fondo o con otro objeto. Requiere un aprendizaje previo y demanda un procesador de imágenes potente para porque segmentarlo adecuadamente.

Los resultados pueden llegar a ser un tanto debido a la precisión de cálculos con respecto a bordes de objetos que se mueven en tiempo real.

2.2.5.4 POR COLORES.

Otro método ampliamente utilizado por su gran fiabilidad y su versatilidad al combinarse con otros métodos. Tiene muchos caminos al momento de su implementación pero su fin es el mismo: segmentar el objeto que tenga el color elegido. El método más usado y entendible es presentar un color a seguir y evaluar cada pixel para notar su proximidad o igualdad con respecto al color.

Su principal problema es que la escena no debería tener muchos objetos del mismo color pues la segmentación no sería única.

2.2.6 ILUMINACIÓN DE LA ESCENA.

Una sección importante en la construcción de la escena es la iluminación, ya que a partir de las condiciones de luz se pueden tener colores más vivos u opacos, estructuras más definidas, y fondos mas determinados.

Un entorno debidamente iluminado es imprescindible para obtener unas condiciones de partida óptimas. En cambio, si la luz no es adecuada, los efectos sobre los resultados pueden ser incluso nocivos. Aunque existen muchas cámaras y sensores preparadas para diversas condiciones de luz, no se le puede confiar la totalidad de este punto a la máquina.

La iluminación de la escena consta de tres sectores importantes: el tipo iluminación, el fondo, la posición de la cámara. De estas tres la más importante es el tipo de iluminación.

2.2.6.1 ILUMINACIÓN FRONTAL.

En esta tipo, la luz incide directamente sobre el objeto ya sea de forma oblicua o difusa, vertical u horizontalmente. También influye mucho el color de la lámpara de iluminación, si bien una iluminación amarilla es más potente y tiene un mayor campo de efecto, la luz blanca es ideal para reconocer colores.

2.2.6.2 ILUMINACIÓN POR RETROALIMENTACIÓN.

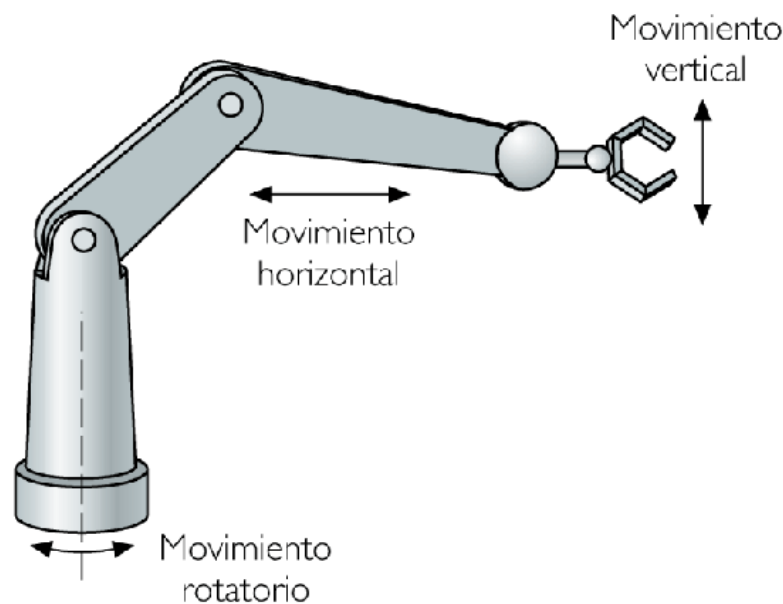
Donde se ilumina una pantalla de forma cuadrada o circular de tal modo que lo que se busca es el contorno del objeto como una sombra particular. Este objeto puede estar detrás o delante del fondo. Este tipo de iluminación puede variar en otros dos tipos: Con la iluminación delante del fondo y la iluminación detrás del fondo.

2.3 BRAZO ROBÓTICO

Un robot es una máquina programable que tiene la capacidad de imitar el comportamiento de seres humanos o animales y que a su vez puede realizar actividades repetitivas y peligrosas, además puede integrar una gran variedad de sensores que le permite adaptarse a nuevas situaciones y entornos de trabajo. (Ospina, 2012)

La robótica es sin duda la rama electrónica e informática con mayores promesas en la actualidad. Su desarrollo es complejo de lograr debido a las grandes condiciones que se le presentan y los inconvenientes de hardware y software que inevitablemente se presentarán. Todo robot está compuesto de su parte mecánica y de una programación. Por lo que resulta imposible que un robot actúe por voluntad propia.

FIGURA 2-X: BRAZO ROBÓTICO TIPO SCARA



FUENTE : <http://www.arqhys.com/articulos/brazo-robotico.html>

El brazo robótico es uno de los apartados en el estudio de la robótica, ya que su funcionalidad va desde maniobra industrial hasta aplicaciones en la exploración industria. Su programación en general suele ser orientada a una sola finalidad y su trabajo puede ser incluso simulado antes de pasar a una implementación en real.

Un robot puede tener brazos robóticos con propósito de cumplir su misión y también un brazo robótico puede actuar aparte como el componente total del robot.

2.3.1 ESPACIO DE TRABAJO.

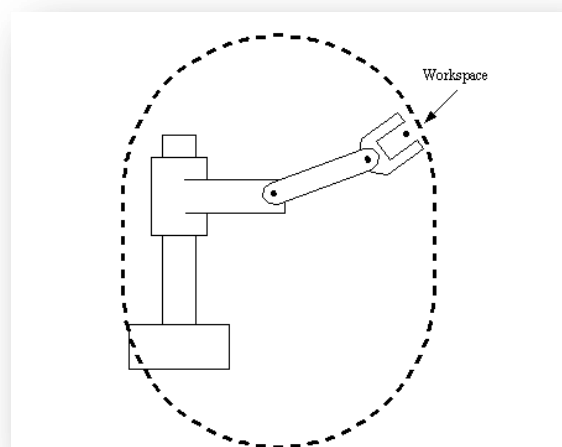
Es la zona de trabajo del brazo robótico. Su dimensión está definida por el número de grados de libertad y por el elemento manipulador.

Por lo que en un espacio físico pueden existir lugares inalcanzables para el brazo robótico, sin embargo, en muchas ocasiones esto no representa mucho problema pues las trayectorias del robot están establecidas.

Muchas industrias que desarrollan brazos robóticos tienen que definir su espacio de trabajo. Incluso este puede ser un parámetro de construcción, a parte de los grados de libertad y eslabones.

Dimensionalmente el espacio de trabajo de un robot es el punto más lejano al que el brazo puede llegar y estudiando de allí su tipo de movimiento y sus grados de libertad.

FIGURA 2-XI:ESPACIO DE TRABAJO DE UN BRAZO ROBÓTICO



FUENTE : http://www-pagines.fib.upc.es/~rob/protegit/treballs/Q2_03-04/general/carmorf.htm

2.3.1.1 GRADOS DE LIBERTAD.

Los Grados de Libertad (GDL) son el número de articulaciones que tiene un brazo robótico, se entiende pues, que entre más grados de libertad, mas variedad de movimientos tendrá el brazo. Pero también se entiende que entre mas GDL se tenga, será más costoso, tanto material como computacionalmente, el mover el brazo hasta la posición necesaria.

2.3.2 CINEMÁTICA DEL ROBOT

La cinemática del robot estudia analítica y matemáticamente el movimiento del mismo. Según sea el tipo de estudio que se siga se busca encontrar: las coordenadas del elemento terminal, los ángulos de los eslabones, su orientación y las coordenadas finales de las articulaciones. Existen dos formas o problemas principales para realizar este análisis: cinemática directa y cinemática inversa.

FIGURA 2-XII: DIFERENCIA CINEMÁTICA INVERSA Y DIRECTA

Diagrama entre cinemática directa e inversa.		
	Cinemática directa -->>	
Valor de las coordenadas Articulares (q_0, q_1, \dots, q_n)		posición y orientación del extremo del robot ($x, y, z, \alpha, \beta, \gamma$)
	<<-- Cinemática inversa	

FUENTE : <http://proton.ucting.udg.mx/materias/robotica/r166/r91/r91.htm>

2.3.2.1 CINEMÁTICA DIRECTA.

La cinemática directa busca encontrar la matriz de transformación homogénea la misma que debe relacionar la posición el elemento final del brazo con respecto al sistema de coordenadas originales.

2.3.2.1.1 MATRIZ DE TRANSFORMACIÓN HOMOGÉNEA.

La matriz de transformación homogénea es una de 4 x 4 que contiene la información la de posición de coordenadas homogéneas. Una matriz de transformación homogénea consta de sub matrices. Su composición es la siguiente:

$$T = \begin{bmatrix} R_{3 \times 3} & P_{3 \times 1} \\ f_{1 \times 3} & 1 \times 1 \end{bmatrix} = \begin{bmatrix} \text{matriz de rotación} & \text{vector posición} \\ \text{transformada perspectiva} & \text{escalado} \end{bmatrix}$$

La cinemática directa tiene muchos métodos para seguir entre los más importantes mencionaremos brevemente.

2.3.2.1.2 ALGORITMO DENAVIT - HARTERBERG.

Este algoritmo define un sistema (X_i, Y_i, Z_i) de coordenadas en cada elemento o eslabón, donde i representa los 6 grados de libertad que tiene el brazo robótico. Una vez se procede a seguir el algoritmo a cabalidad para no confundir los datos.

2.3.2.2 CINEMÁTICA INVERSA.

La cinemática inversa será más explicado pues es la que usaremos para el propósito de nuestro estudio robótico y posterior simulación. La premisa de la cinemática inversa dice que se necesita saber el punto final hacia donde se debe dirigir el brazo robótico y con este dato, entontar las coordenadas de cada una de las articulaciones.

Existen muchos métodos para determinar el movimiento necesario de las articulaciones del robot para que su movimiento sea adecuado según su necesidad. Los más usados son:

- Método Geométrico
- Método de la matriz de transformación homogénea.
- Matriz Jacobiana

No se puede apuntar con seguridad al más adecuado o al más óptimo; ya que los tres tienen sus ventajas y desventajas. Pero no existe gran controversia al momento de asegurar que el Método Geométrico es menos exigente a la hora de implementar. Por tal razón, este apartado se centrará únicamente en este método.

2.3.2.2.1 MÉTODO GEOMÉTRICO.

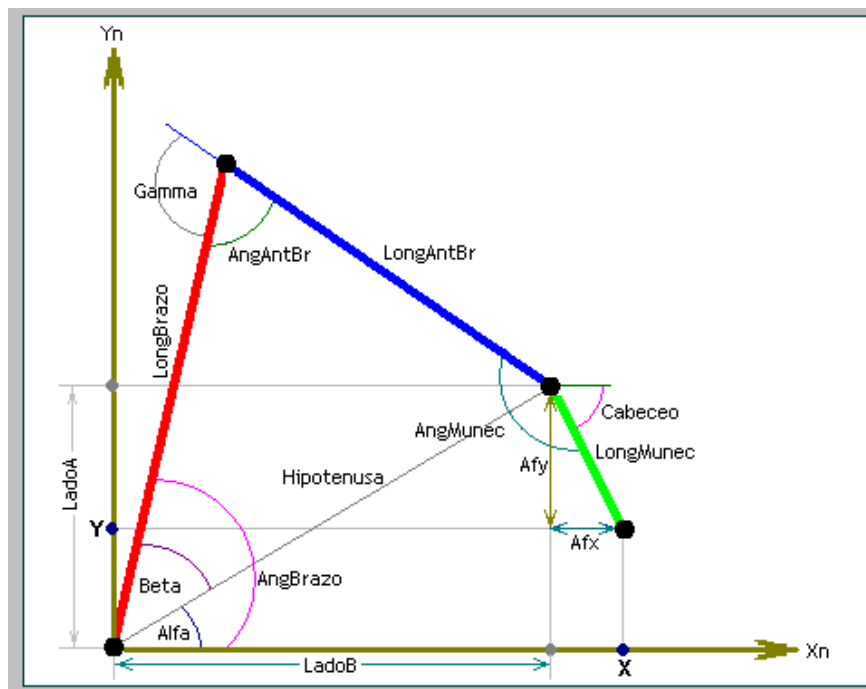
Se procura encontrar los datos necesarios para poder orientar el brazo robótico, su proceso se sirve de identidades geométricas, y trigonometría básica.

Este método tiene un fundamento puramente geométrico y no necesita más que los siguientes datos iniciales:

- Extensión de las articulaciones.
- Coordenadas finales del terminal del brazo.

Entonces aplicamos trigonometría básica y leyes geométricas muy conocidas.

FIGURA 2-XIII: BRAZO ROBÓTICO 3 GRADOS DE LIBERTAD.



FUENTE : <https://sites.google.com/site/proyectosroboticos/cinematica-inversa-ii>.

Dados los grados de libertad necesarios se extenderá recíprocamente la cantidad de cálculo necesario. Por ejemplo para el brazo expuesto en la figura tendríamos los siguientes datos explicativos. Teniendo conocimiento de: Longitud de Brazo, Longitud de Antebrazo, Longitud de la muñeca, Ángulo de Cabeceo, Posición final en x, Posición final en y. Buscamos hallar:

- Ángulo del brazo.
- Ángulo del antebrazo.
- Ángulo de la muñeca.

FIGURA 2-XIV: RESOLUCIÓN BRAZO 3 GDL

$$Afx = \cos(\text{Cabeceo}) * \text{LongMunec}$$

$$\text{LadoB} = X - Afx$$

$$Afy = \sin(\text{Cabeceo}) * \text{LongMunec}$$

$$\text{LadoA} = Y - Afy$$

$$\text{Hipotenusa} = \sqrt{(\text{LadoA})^2 + (\text{LadoB})^2}$$

$$\text{Alfa} = \tan^{-1}\left(\frac{\text{LadoA}}{\text{LadoB}}\right)$$

$$\text{Beta} = \cos^{-1} \frac{(\text{LongBrazo}^2 - \text{LongAntBr}^2 + \text{Hipotenusa}^2)}{2 * \text{LongBrazo} * \text{Hipotenusa}}$$

$$\text{AngBrazo} = \text{Alfa} + \text{Beta}$$

$$\text{Gamma} = \cos^{-1} \frac{(\text{LongBrazo}^2 + \text{LongAntBr}^2 - \text{Hipotenusa}^2)}{2 * \text{LongBrazo} * \text{LongAntBr}}$$

$$\text{AngAntBr} = -(180 - \text{Gamma})$$

$$\text{AngMunec} = \text{Cabeceo} - \text{AngBrazo} - \text{AngAntBr}$$

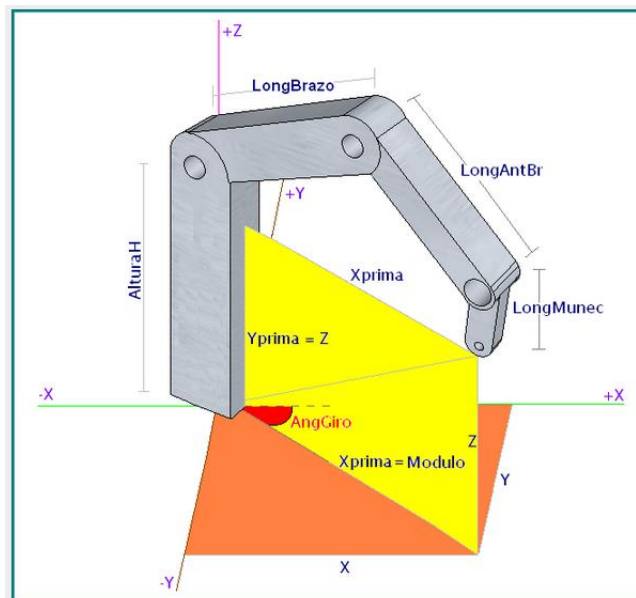
FUENTE : Ilbay Llangarí Luis Guido.

Si al mismo brazo robótico anteriormente estudiado le aumentamos un grado de libertad, podríamos acceder a un movimiento en tres dimensiones.

El problema radica en que los cálculos se extiendan demasiado, dados los nuevos parámetros. Sin embargo, podemos aplicar un proceso que nos haga utilizar los mismo cálculos anteriormente resueltos. Para ello, primero nos fijaremos en el movimiento que produce la 4 articulación.

Si nos damos cuenta, esta mueve a todo el brazo en un ángulo definido y al final el brazo nos queda de forma que solo necesitamos realizar los mismos cálculos hechos. Hay que recalcar, que el punto final también ha sido cambiado por el movimiento en el que intervino el último eslabón o el elemento terminal. Entonces son tres las incógnitas : el ángulo de giro, el nuevo punto en X y el nuevo punto en Y (Esto puede variar según el tipo de sistema de coordenadas que se esté usando).

FIGURA 2-XV: BRAZO ROBÓTICO 4 GRADOS DE LIBERTAD.



FUENTE : <https://sites.google.com/site/proyectosroboticos/cinematica-inversa-iii>

A partir de los 4 GDL, los brazos robóticos pueden acceder a cualquier punto en el espacio tridimensional de trabajo. La base y primera articulación suelen ser de materiales resistentes y relativamente pesados para poder soportar el peso del movimiento total.

2.3.2.3 SIMULACIÓN DE UN BRAZO ROBÓTICO.

Los costos de implementación en robótica son en general muy altos, tomando en cuenta el estudio previo que se tiene que realizar ya que este debe ser preciso para que su construcción no falle en ningún detalle, ya que esto significaría el desperdicio de dinero.

Las simulaciones son, por tal motivo, prueban ser más que una simple herramienta didáctica, sino también un paso importante en el desarrollo de los robots y en este caso, brazo robótico.

Entre los simuladores más usados se encuentran:

- Robot Studio
- SSim
- Robomind

Escoger uno u otro depende mucho de los requerimientos.

Sin embargo, si se necesita un estudio matemático para un movimiento robótico se pueden emplear otras herramientas orientadas como el Matlab.

Entre estos podemos destacar el movimiento de las articulaciones del brazo robótico. Ya que todo robot que se precie de serlo necesita tener articulaciones, unos más que otros, pero se considera que un robot funcional debe tener al menos tres grados de libertad.

2.4 SOFTWARE.

Existe decenas de entornos y programas para aplicaciones de V.A., pero para los fines propios y por las ventajas de experiencia y la documentación se han usado los siguientes sistemas operativos, lenguajes y librerías.

2.4.1 LINUX UBUNTU 14.04 LTS.

Es un sistema operativo basado en Unix, perteneciente a Linux. El sistema fue desarrollado gracias y completado por varios investigadores desde los años 60. Con el tiempo su premisa de Open Source ha sido preferida por muchos investigadores y aficionados al software. También ha servido de plataforma de varias aplicaciones, librerías y lenguajes que hacen de Linux es rincón de la libertad del software.

Actualmente acaba de ser lanzada la versión del año 2015.

2.4.2 LENGUAJE C++.

Es un lenguaje de programación ampliamente usado desde su creación en 1983. De principio fue pensado como una extensión del C, se llamaba *C with class*, pero en 1983 lo renombraron magistralmente C++. Debido a su gran difusión y éxito se llegó a estandarizar.

Ocupa el primer lugar de uso entre los distintos lenguajes debido a su éxito entre los programadores, sus potentes herramientas y que muchos recursos están escritos en el mismo lenguaje. Tiene las ventajas del C original pero con expresiones, flexibilidad y mejor concisión y eficiencia.

Tiene varias evoluciones como la programación orientada a objetos.

Forma de instalación en Ubuntu 14.04.

FIGURA 2-XVI: INSTALACIÓN C++ EN UBUNTU.

```
tesis@tesis:~$ sudo apt-get install build-essential
[sudo] password for tesis:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
build-essential ya está en su versión más reciente.
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
 linux-headers-3.13.0-32 linux-headers-3.13.0-32-generic
 linux-image-3.13.0-32-generic linux-image-extra-3.13.0-32-generic
```

FUENTE : Ilbay Llangarí Luis Guido.

2.4.3 OPENCV.

OpenCV Es una potente librería de visión artificial y machine learning, tiene licencia BSD, lo que permite utilizar y modificar el código.

Está escrito en C/C++ lo que la hace más propensa a seguir en desarrollo, prueba de ello es que tiene una gran comunidad, casi 50000, y más de 7 millones de descargas.

Es una librería muy comercial y tiene más de 2500 algoritmos.

Sus funciones principales son identificar objetos, caras, clasificar acciones humanas, tracking de objetos, modelos 3d.

Una mayoría de aplicaciones de Open Source de V.A. han elegido a OpenCV para su desarrollo puesto su gran capacidad y su documentación amplia. Entre sus aplicaciones más populares tenemos:

- Detector de movimiento
- Detección de parámetros en dos y tres dimensiones.
- Segmentación y reconocimiento.
- Detección de Objetos.
- Calibración de cámaras.

Sus funciones básicas son:

Core: Se definen las estructuras de datos básicas que usan el resto de los módulos.

Imgproc: módulo procesamiento de imágenes como : filtrado lineal / no lineal, transformaciones afines, conversión del espacio de color, histogramas.

Video: módulo de análisis de video que incluye algoritmo para la estimación del movimiento. Extracción de fondo.

Calib3d: Algoritmos básicos de visión múltiple como calibración de cámaras stereo y otras cámaras.

Features 2d: detectores de características.

Objdetect: detección de objetos instancias o clases predefinidas como: caras, ojos, gente, coches.

Highgui: Todo lo relacionado a la interfaz gráfica de OpenCV. Permiten importar imágenes y video.

GPU: algoritmos acelerados x hardware para distintos módulos OpenCV.

2.4.4 CMAKE.

Es un sistema de compilación de código abierto. Es desarrollado por Kitware. Se compone de una familia de herramientas para construir y probar el paquete de software. Se usa para controlar el proceso de compilación.

2.4.5 MATLAB.

Laboratorio de Matrices. Es un sistema integrado para la resolución de cálculos matemáticos complejos a través de una matriz a dimensional que no se necesita declarar en tamaño. Además es usado para el tratamiento de datos, y una fundamental herramienta en la ingeniería por sus prestaciones en la obtención de resultados de alto nivel y su comunicación con otros tipos de IDE y de hardware de variadas utilidades, impresión de gráficas sobre análisis de señales y creación de interfaz de control.

Tiene paquetes de soluciones prediseñadas llamados TOOLBOX, las cuales están orientadas en ramas particulares como:

- Identificación de sistemas
- Diseño de sistemas de control
- Procesamiento de señales
- Redes Neuronales.

2.4.5.1 FUNCIONES USADAS.

Matlab cuenta con muchas funciones específicas que pueden ayudar a las aplicaciones de simulación.

2.4.5.1.1 FOR.

Función usada para crear un bucle repetitivo, ejecutando una misma acción con valores variables un número de veces definido en la misma constitución de la función.

Su forma es la siguiente:

FOR i=1 : 1: 20

end

2.4.5.1.2 PLOT3.

Función utilizada para dibujar un gráfico en un sistema tridimensional. Se requiere la definición previa de los valores de cada uno de los ejes (X,Y,Z)

Su escritura es la siguiente:

plot3(funx, funy, funz);

2.4.5.1.3 HOLD.

Solo puede tener dos estado, encendido o apagado. Sirve para mantener los gráficos anteriores impresos junto los gráficos actuales, de tal forma que pueda mostrar la progresión de la función o las funciones dentro del tiempo ejecutado.

Su escritura es la siguiente: **hold on;**

2.4.5.1.4 FUNCIONES TRIGONOMÉTRICAS.

Se refiere a todas la funciones trigonométricas existentes. Matlab cuenta con una gran gama de funcionalidad para la trigonometría.

Las funciones posibles para utilizar en matlab son: seno , coseno, tangente, arcoseno, arcocoseno, arcotangente y todas su funciones inversas. Es importante mencionar que Matlab toma en cuenta a los ángulos en unidades de radianes y no de grados, por lo que en ciertos casos son necesarias las transformaciones entre unidades.

2.4.5.1.5 FUNCTION.

Es el equivalente a las funciones particulares de otros lenguajes de programación. Se puede escribir en un archivo .m distinto y dadas las entradas definidas devolverás las respuestas necesarias. Se la puede llamar las veces que sean necesarias y evita la aglomeración de código fuente innecesario y repetitivo en uno solo estándar. Además es ideal al momento de la corrección de errores ya que minimiza la búsqueda de los código mal escritos o mal estructurados. Su escritura es la siguiente:

```
function [xxx yyy zzz]= Dof3(i, w, fw)
```

```
xxx = [0, BrazoPX, AntBrPX, MunecPX];
```

```
yyy = [0, BrazoPY, AntBrPY, MunecPY];
```

```
zzz = [0, Puntox1 ,Puntox2 ,Puntox3 ];
```

2.5 KINECT.

FIGURA 2-XVII : KINECT XBOX 360.



FUENTE : http://www.elrastro.cl/aviso_6716551-kinect-xbox-360-mas-1-juego.html

Kinect es un sensor compuesto desarrollado por Microsoft con la firme intención de aumentar el uso de la videoconsola XBOX. Se lanzó en el 2010 y a partir de la liberación de su código en el 2011 ha sido la plataforma de muchas investigaciones respecto a la V.A. Su éxito se debe a su bajo costo y a sus prestaciones completas. Se conoce al Kinect como un sensor, pero en realidad Kinect es un sistema de sensores, independientes que trabajan en conjunto según la necesidad.

Al notar que muchos investigadores usaban el Kinect en la PC con los controladores libres que algunos grupos venían desarrollando, la Microsoft pensó en demandarlos y cortar desde la cabeza todos estos desarrollos "piratas" sobre su producto.

No lo hizo. En cambio lanzó al público su propio controlador de Kinect que funciona en Windows 7 y 8. No tiene aún SDK para Linux. Tiene muchas ventajas ya que se puede usar como una clase o una función en el lenguaje orientado a objetos C#, facilitando así, la construcción de aplicaciones.

Sin embargo, los controladores libres aún se siguen desarrollando y son muchos los programadores que se vuelcan a usarlo. En parte, porque el sistema operativo Linux se

ha convertido en sinónimo de desarrollo Open Source, y también por el espíritu mismo de esta tendencia.

Entre las librerías más usadas están las desarrolladas por Open NI y OpenKinect.

2.5.1 MENÚ DE COMPONENTES

Generales

- Infrarrojo (IR) CMOS camera.
- Protector Infrarrojo -830, 60mW diodo laser.

Audio

- Micrófonos

Controladores

- Motor
- Acelerómetro (3- axes)

Procesador y Memoria

- Prime Sense Chip PSI080 -A2
- 64 MB DDR2 SDRAM

Kinect

- Color Camera 640 x 480; 640x480 @fps output.
- IR Camera: 1280 x 1024 , 640x480 @30 fps output
- Rango de operación (Sensor Depth) . Rango= 0,8 - 3,5.
- FOV= 58° H, 45 V, 70D
- Resolución espacial (@ 2m distancia) = 3mm
- Resolución Depth (@ 2m distancia) = 1cm

Además Kinect soluciona muchos problemas de comunicación con el computador gracias al tipo de adaptador USB para la PC.

También tiene la facultad de ajustar la cámara según una inclinación deseada pues tiene un motor integrado en su base. Posee unos 27° de trabajo verticales.

2.5.2 VISION DEL KINECT.

Los elementos más usados del Kinect son su cámara RGB y su sensor de profundidad llamado Depth.

2.5.2.1 VISION RGB.

La cámara RGB es una cámara clásica que podemos hallar en cualquier dispositivo del obtención de imágenes. Esta posee una resolución de ocho bits de tipo VGA. La diferencia con las demás se halla tan solo en la resolución y los frames por segundo (fps) de la cámara. Ya que esta tiene unos 30 fps. Los FPS son la cantidad de imágenes que se recoge por segundo, entre mas FPS se pueda obtener más definido se notará el video.

A simple vista se puede notar como una cámara más potente que las convencionales, pero cabe recalcar los siguientes elementos propios de la misma:

Las cámaras CMOS tienen un menor costo y son menos afectadas por los cambios de luz, pues tiene una estructura simple lo cual desemboca en trabajos del mismo calibre.

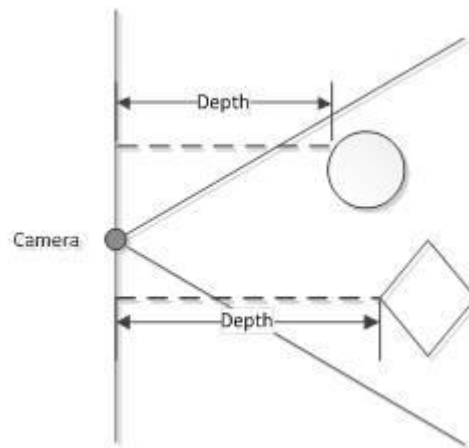
La resolución de la cámara también se puede modificar a través de código **FUENTE**, y de preferencia esta debe ser igual al del sensor infrarrojo.

2.5.2.2 IMAGEN DEPTH.

La imagen de profundidad (Depth) está completamente unificada al llamado sensor 3D, esta se compone de dos partes: un proyector de rayos IR y un sensor CMOS monocromático. El sensor que percibe los rayos infrarrojos puede capturar datos de video en 3D bajo casi cualquier estado de la luz. Tiene una resolución de (640 x 480) con 16-bit de profundidad a 30 fps. Y unos 2048 niveles de sensibilidad.

La técnica que Kinect usa para obtener la información de la profundidad fue desarrollada por la dueña de su chip Prime Sense. La técnica se llama luz estructurada. La misma técnica a venido siendo utilizada en aplicaciones donde se necesita mucha precisión, y por lo tanto los sensores usados tienden a tener costos muy elevados.

FIGURA 2-XVIII: PROYECCIÓN DEL INFRARROJO DEL KINECT.

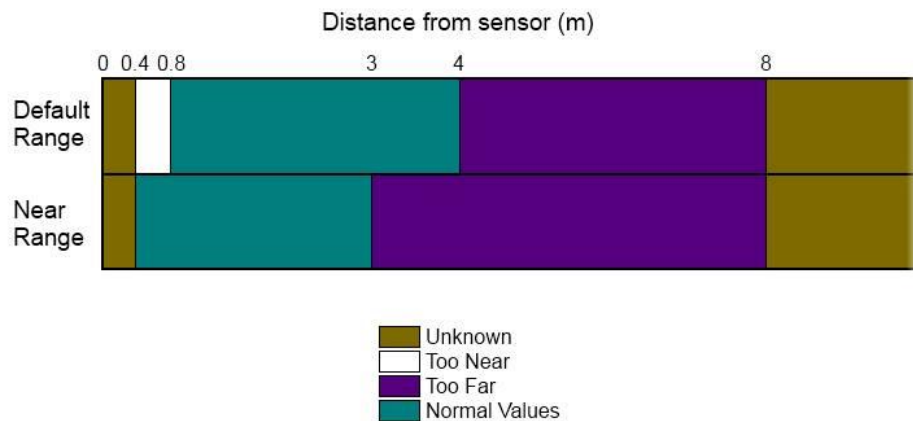


FUENTE : <https://msdn.microsoft.com/en-us/library/hh973078#CommunityContent>

Aunque el paquete Kinect, contiene especificaciones básicas para su uso, solo el investigador tienen la última palabra en cuanto a la distancia que se necesita estudiar. En la siguiente imagen podemos verificar los rangos de acción. Empero, hay que recalcar que la forma de devolver información de parte del sensor no es la misma en cada uno de los rangos. Las medidas varían bastante mientras el objeto es cercano y

varían muy poco cuando la distancia es demasiado extensa. Por ellos, algunas fórmulas de conversión del dato devuelto por el sensor a distancia real, no son confiables.

FIGURA 2-XIX: RANGO NOMINAL DE VISIÓN DE PROFUNDIDAD DEL KINECT.



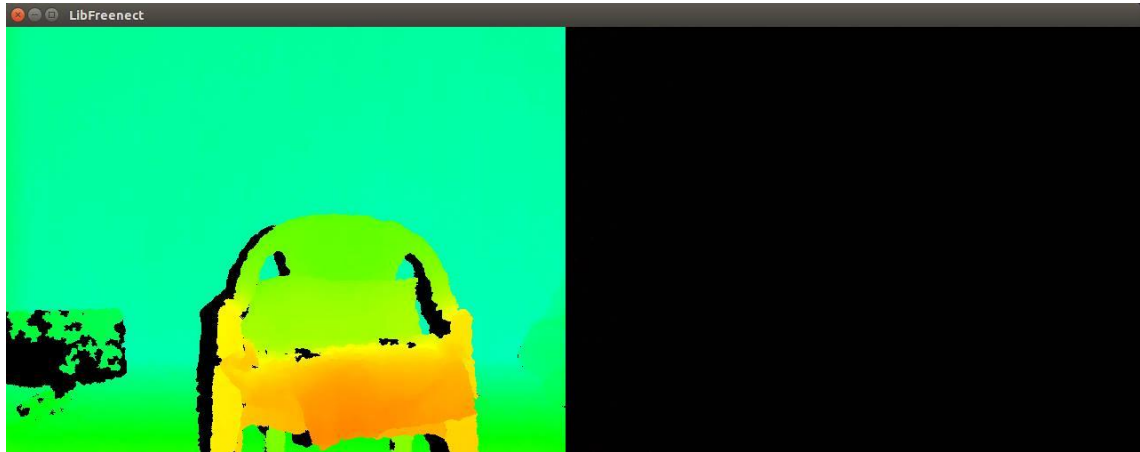
FUENTE : <https://msdn.microsoft.com/en-us/library/hh973078#CommunityContent>

2.5.2.2.1 LUZ ESTRUCTURADA.

La luz estructurada consiste en que a partir de una **FUENTE** de luz, o de un haz de luz se la proyecte sobre una escena, los objetos situados en la misma serán impactados por el haz y lo deformarán según su posición y según su forma. Esta deformación puede ser observada y recogida por otra cámara estratégicamente situada. Una vez recogida la información se puede saber la distancia a la que se encuentra el objeto.

En la siguiente imagen podemos ver la imagen Depth de una sencilla escena. La parte derecha está completamente negra pues la escena se tomo con ausencia total de luz. Comprobándose así la explicación de luz estructurada.

FIGURA 2-XX: IMAGEN RGB Y DEPTH EN AUSENCIA DE LUZ.



FUENTE : Ilbay Llangarí Luis Guido.

2.5.3 LIBFREENECT.

Es el controlador para el Kinect, desarrollado como Open Source por la comunidad Open Kinect.

FIGURA 2-XXI : LOGO OPEN KINECT



FUENTE : http://openkinect.org/wiki/File:Logo_openkinect_shaded.svg

Open Kinect , es una comunidad con más de 2000 miembros, que ha desarrollado varios controladores, demos y ejemplos para Kinect, que se pueden instalar en Linux, Apple y Mac. Sus funciones más populares son las de la imagen RGB y Depth, además de ciertos ejemplos de compilación con OpenCV, y el uso del motor y los leds del Kinect.

Todo código esta bajo la disposición Apache 2.0 o licencias GPL2 opcionales. Tiene ejemplos desarrollados para Python, C++, C#, Java JNI.

En una investigación realizada en una universidad española se usa una fórmula matemática para convertir la información de profundidad que arroja este controlador.

$$d_{mm} = \frac{1000}{-0.00307 * draw + 3.33}$$

Esta ecuación será evaluada y comparada con el tratamiento exclusivo que se le ha dado a la información de profundidad obtenida en el presente proyecto.

2.6 TRACKING DE OBJETOS.

El tracking de objetos es una de las clásicas aplicaciones de la V.A. Su importancia radica que cualquier trabajo o problema que se pueda resolver a través de la V.A. implica reconocer un objeto y poder seguirlo en el escenario y a la vez que este seguimiento sea en tiempo real.

Empero, que el número de aplicaciones existentes sea extenso, no implica que se haya convertido en un trabajo fácil para los investigadores. De hecho se cuenta en uno de los más complicados aún hoy.

El motivo de este es que existe muchos métodos para la obtención de la posición y segmentación del objeto, pero ninguna se puede alzar como estándar y presumir que goza de toda la potencialidad y diversidad necesaria. Que un escenario se vea afectado por algunos factores externos influyen mucho en este problema.

La iluminación, la complejidad de la escena , el tamaño del objeto , su posición y hasta su forma, son algunos de los problemas que siempre aparecerán en el camino.

Se puede decir que un objeto está siendo seguido cuando sobre el o a su alrededor se dibuja una marca. Y además conocemos su posición en tiempo real ya sea bidimensional o tridimensional.

Debido a que los algoritmos en tres dimensiones dependen demasiado del tipo de cámara que se va a utilizar no se puede definir con precisión que un algoritmo va a funcionar de forma global con cualquier hardware. No así con los algoritmos en dos

dimensiones que no dependen con tanta premura de un tipo específico de hardware. Por tal motivo para poder puntualizar tres nombres de algoritmos y llevar a cabo su estudio hay que partir de la estandarización mediática que nos ofrece el seguimiento en dos dimensiones y a partir de allí extender estas propiedades hacia las prestaciones del Kinect y lograr el tracking 3D.

2.6.1 ALGORITMOS DE TRACKING DE OBJETOS.

Al existir muchos tipos de cámaras de video y de captura se ha disparado su uso y su investigación. Existe gran cantidad de algoritmos para segmentación y tracking de objetos, pero se realizará el estudio sobre tres por sus condiciones correctas y respuestas por evaluar.

2.6.1.1 MÉTODO CAMSHIFT.

Este es uno de los algoritmos más usados para el seguimiento de objeto por su facilidad de comprensión y por su maleabilidad.

Es una modificación del algoritmo MeanShift que trataba predecir la nueva posición de un objeto, mediante el grado de pertenencia de un conjunto de puntos seleccionado manualmente y conocido como región de interés (ROI). El CAMShift, usa la misma base, y complementa la posición del objeto analizando y comparando el histograma de dos posiciones anteriores.

2.6.1.1.1 ALGORITMO CAMSHIFT.

Tiene tres puntos básicos e importantes:

- 1.- Detección de objetos en movimientos
- 2.- El seguimiento de los mismos en cada instante

3.- El respectivo análisis.

Su funcionamiento se basa en estudiar el histograma de la imagen en tiempo real y continuamente. Podríamos definirlo de la siguiente manera.

Para la evaluación se realiza una posición manual y se la contrasta con la posición estimada del algoritmo.

En los ensayos se detecta y se sigue constantemente incluso en oclusiones cortas. El seguimiento de la trayectoria de este método se basa en tres etapas:

INICIALIZACIÓN.

- Adquiere y convierte la imagen RGB a HSV. Para calcular el histograma matiz color.
- Pre procesamiento de la imagen desde la cámara
- El usuario selecciona de forma manual la ROI.

El nuevo formato HSV permite independizar el todo de brillo en un color determinado siendo muy útil al momento de realizar la detección de un objeto a partir de su histograma.

CALCULO DEL HISTOGRAMA

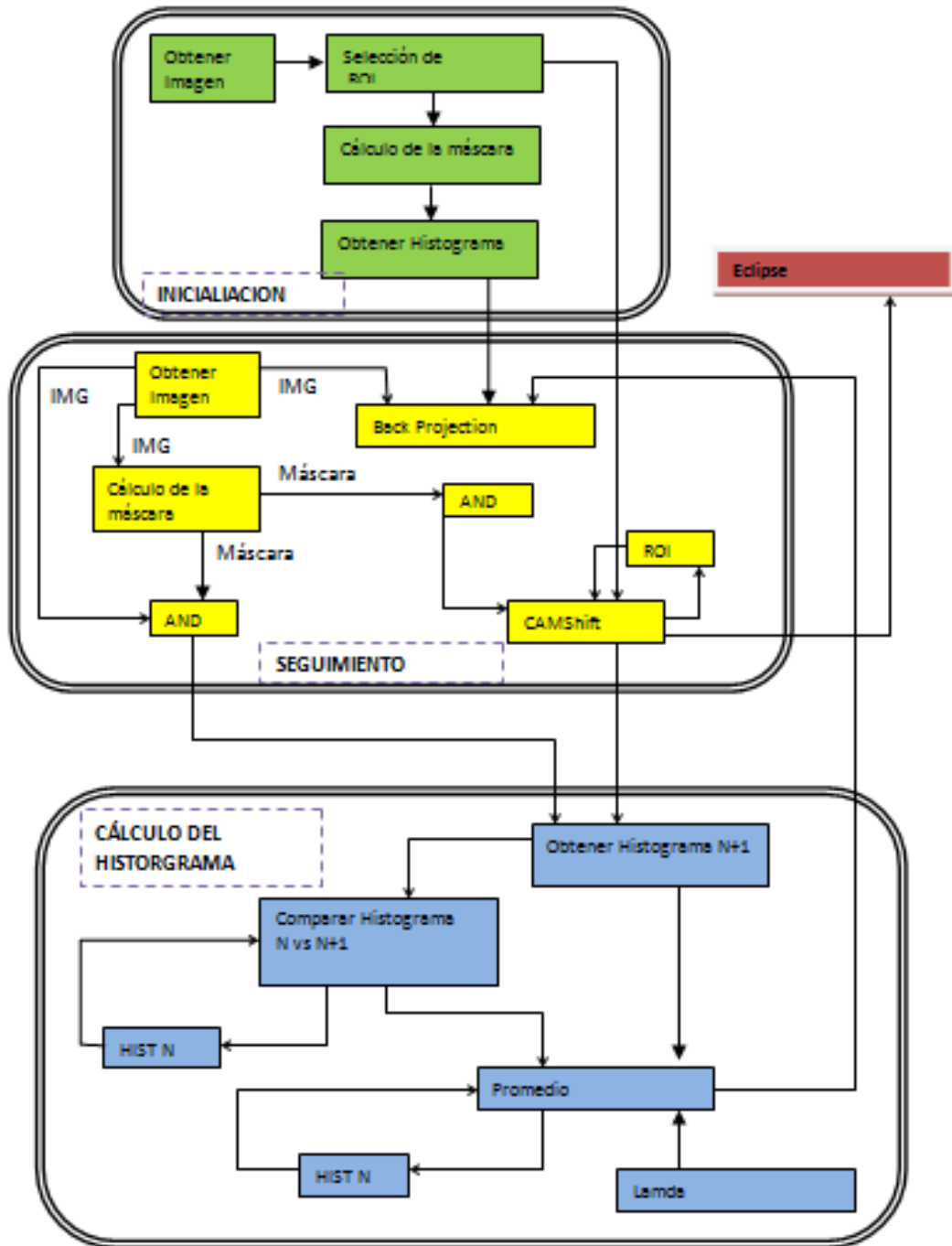
Se realiza el cálculo del histograma a partir del ROI, se compara el histograma inicial contra el actual. De principio, como es de esperarse no se puede tener un histograma, entonces la respuesta saldrá solo del actual. En cambio en las siguientes iteraciones se puede comparar de forma más detenida. Por tal motivo la primera detección es muy rápida mientras que las siguientes tardan notoriamente un poco más.

SEGUIMIENTO

- Se ubica o se halla la ubicación del objeto en coordenadas.
- Se realiza el cálculo del backproject a partir del Histograma del color se crea una imagen que incida la zona posible de seguimiento.
- Se aplica el filtro Canny. El filtro Canny es un operador que usa algoritmos de múltiples etapas para detectar bordes. Esto se focaliza en el objeto de interés.

- Finalmente se usa una AND entre las dos imágenes, anterior y actual y se coloca de backproject.
- La AND se aplica PIXEL x PIXEL.

FIGURA 2-XXII: DIAGRAMA DE ALGORITMO CAMSHIFT.



FUENTE :

GIAR Algoritmo de Seguimiento de objetos basado en visión asistida por computador en tiempo real utilizando CAMShift e histogramas ponderados.

VENTAJAS

- No hay diferencia entre detección y seguimiento pues se ocupa el mismo algoritmo
- Tomará como característica el blob anterior propuesto.
- Funciona bien para objetos que sobresalgan de del fondo.
- Una tasa de 20 fps.

DESVENTAJAS

- En exteriores se pierde con mucha facilidad.
- Requiere un previo entrenamiento.
- Susceptible a cambios de luz.

2.6.1.2 MÉTODO DE UMBRALIZACIÓN.

Es uno de los métodos más antiguos y más sencillos de implementar. Su base es también muy práctica. Si bien es cierto tiene resultados bastante buenos, es muy delicado ante las variaciones de los efectos externos.

Es una de las técnicas de segmentación que más se usa en las industrias. Su funcionamiento básico trata de separar el objeto o los objetos buscados de el fondo de la escena o de lo que sobra de la escena.

La práctica más usada es hacer la segmentación estudiando el histograma. Cuando este presenta dos picos en el valle se debe fijar en uno de ellos. Esto se realiza una vez binarizada la imagen.

Usualmente se ocupa el algoritmo OTSU, desarrollado por el investigador del mismo nombre para ubicar el umbral.

Existe otro tipo de Umbralización que se compara con un umbral impuesto por el color designado en formato HSV.

En este caso el umbral no completa la interpretación de la imagen entonces se necesita emplear técnicas de procesamiento morfológicas como: apertura y cierre para eliminar completamente el ruido que aún queda tras el umbral.

De este modo se logra obtener una imagen sencilla y libre, en lo posible, del ruido. Y se logra tener un algoritmo bastante robusto y de grandes prestaciones.

2.6.1.2.1 ALGORITMO DE UMBRALIZACIÓN.

Los pasos del siguiente algoritmo son los siguientes, cabe recalcar que este algoritmo ha sido extendido con operaciones morfológicas que segmentan de manera más concreta la imagen.

INICIALIZACIÓN

- Obtención de la imagen partir de la cámara Kinect

- Se realiza un pre proceso de la imagen convirtiendo

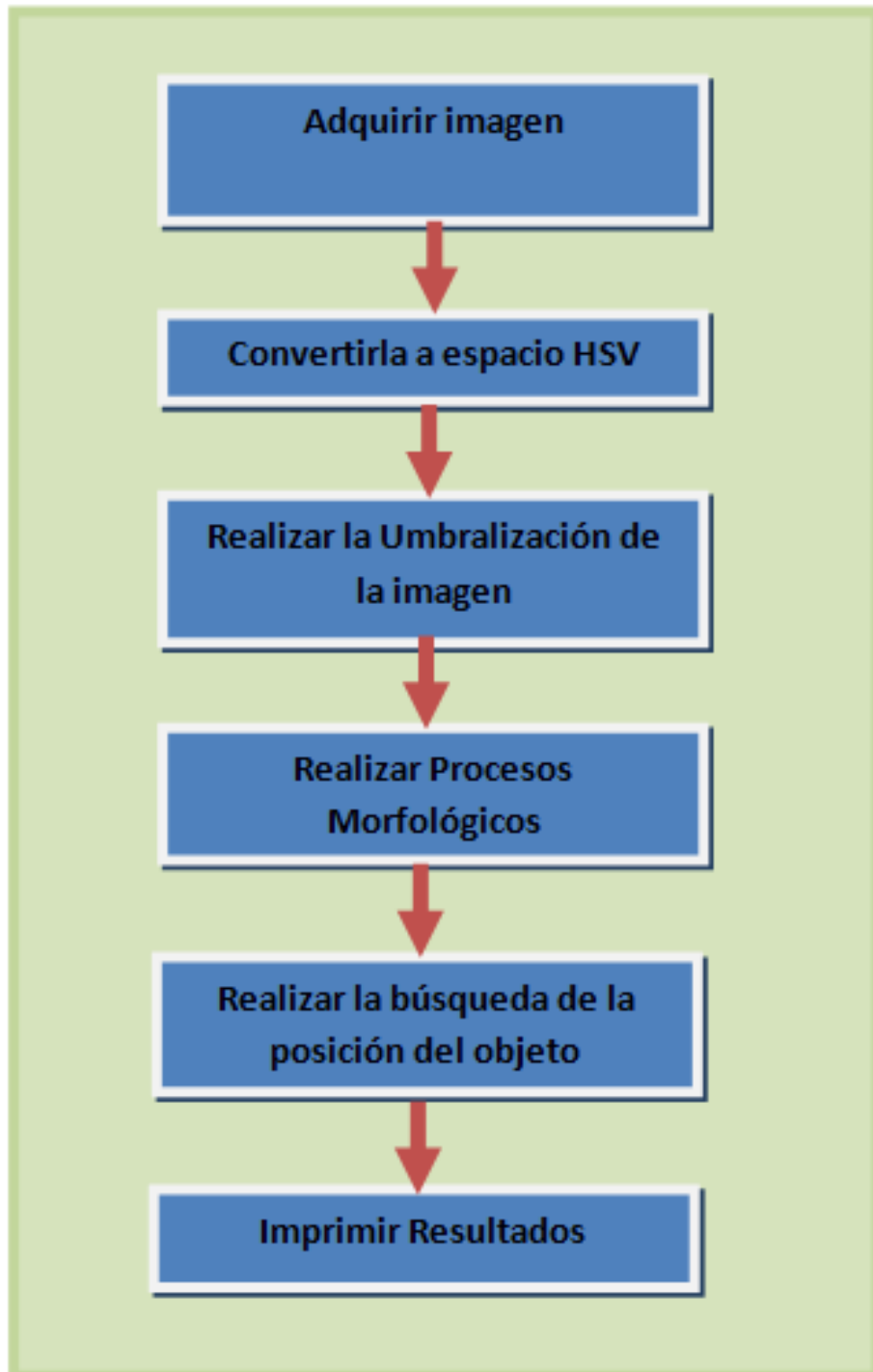
PROCESAMIENTO DE IMAGEN

- Realizar la Umbralización de la imagen para obtener un campo sencillo a seguir siendo tratado.
- Realizar una serie de acciones morfológicas para la eliminación de ruido y segmentación de la imagen

CÁLCULOS

- Obtención de el centro del área del objeto encontrado tras cálculos matemáticos.
- Dibujo de un punto sobre el punto y el área encontrada.
- En la siguiente imagen se presenta una imagen de la misma.

FIGURA 2-XXIII: DIAGRAMA DE ALGORITMO DE UMBRALIZACIÓN.



FUENTE : Ibay Llangarí Luis Guido.

VENTAJAS

- Proceso sencillo, fácil de entender y de mejorar
- Resultados fiables y concretos.
- Respuesta rápida.

DESVENTAJAS

- Necesidad de muchas transformaciones para obtener una respuesta aceptable.
- No tan robusto en espacios exteriores.

2.6.1.3 MÉTODO POR COINCIDENCIA DE COLORES.

El del color, es un método eficiente y ligero que se basa en comparar cada pixel de la imagen con el color elegido para su demostración. Para ello primero se debe proponer un color con cual comparar y como resultado esta operación se debe tener una imagen segmentada y el objeto separado.

Este método tiene una particularidad y es que resulta imposible saber cuál es la configuración RGB genuina de un objeto. Por tal razón se necesita proponer también un rango de aceptación para el color con el cual los colores un tanto alejados a la configuración dicha puedan ser aceptados como el objeto.

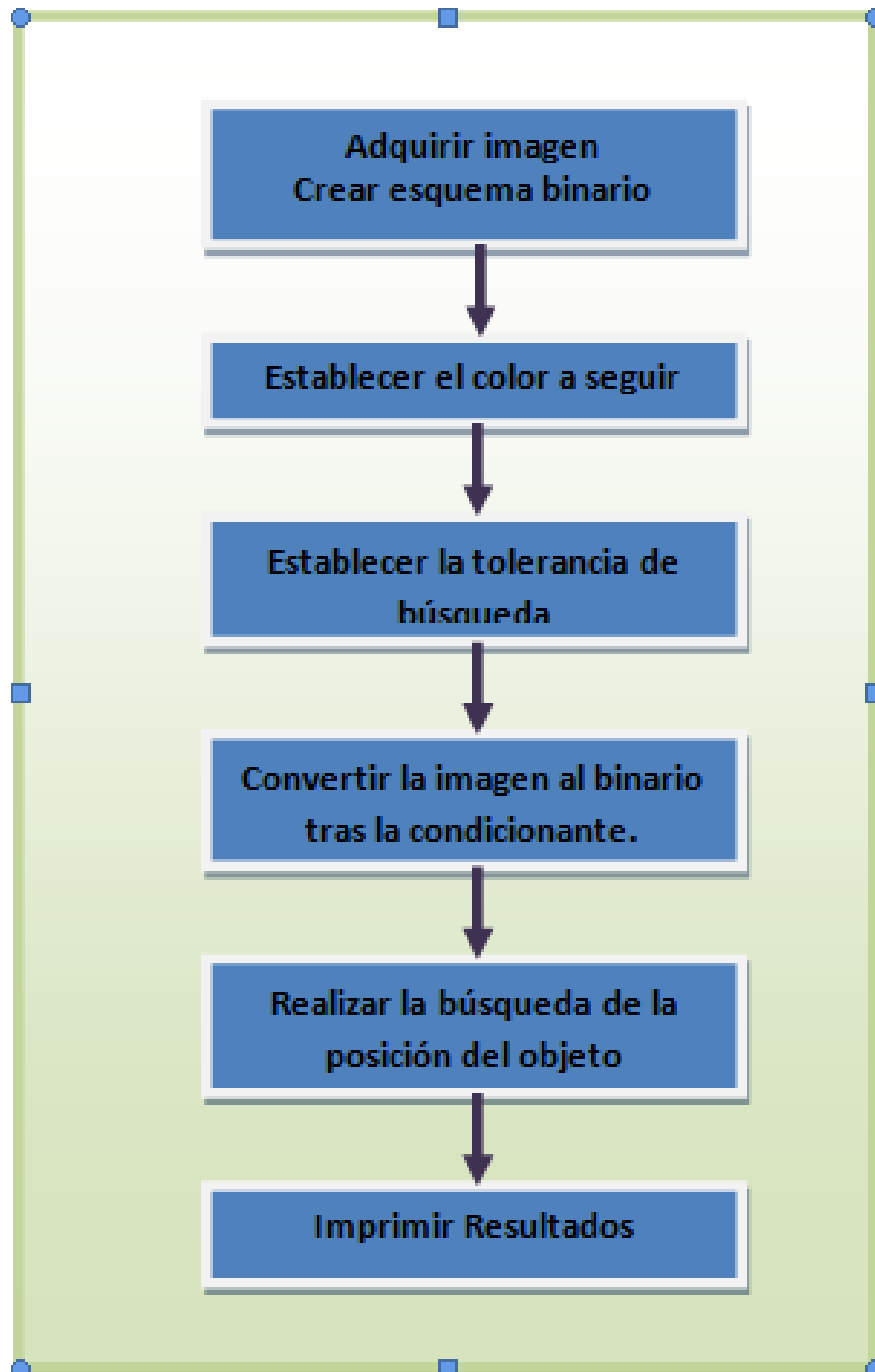
2.6.1.3.1 ALGORITMO POR COINCIDENCIA DE COLORES.

El algoritmo consta de los siguiente pasos principales.

- Proponemos el color a seguir, en formato RGB.
- Establecemos el umbral del color, esto significa proponer un rango que color de aceptación como el color propuesto a seguir.
- Se toma cada píxel y se realiza un cálculo que después será impreso en una imagen binaria previamente creado. Realizamos el mapeo de cada pixel . Si cada canal esta dentro del intervalo: [*valor - tolerancia, valor + tolerancia*] ponemos el pixel en blanco. Caso contrario lo ponemos en Negro.
- Se recorre la imagen binarizada esperando encontrar el objeto resultante para encontrar su posición.
- Encontradas las posiciones , se enmarca el objeto resultante.
- Se muestra las imágenes resultantes.

En la siguiente imagen se muestra el diagrama del algoritmo.

FIGURA 2-XXIV: DIAGRAMA ALGORITMO POR COINCIDENCIA DE COLORES.



FUENTE: Ibay Llangarí Luis Guido.

VENTAJAS

- Resultados fiables.
- Algoritmo sencillo de implementar

DESVENTAJAS

- Necesidad de imponer un rango para que los colores puedan ser aceptados dentro de uno de los principales.
- Imposibilidad de tener un objeto completamente de acuerdo a la configuración del color.
- Susceptible a cambios de luz.
- Necesidad de resoluciones medianas, pues dada una imagen en Alta Definición, se tendría un gran tiempo de comparación. Se alentaría el proceso.

2.7 RESUMEN COMPARATIVO.

TABLA 2-1 :TABLA COMPARATIVA DE LOS ALGORITMOS

Algoritmo de Coincidencia de Colores		Algoritmo Umbralización	Algoritmo CAMShift
Trabaja en imágenes de tipo RGB		Trabaja con imágenes tipo HSV	Trabaja con imágenes RGB y con el histograma del mismo.
Costo	Computacional mediano.	Costo computacional mediano.	Costo computacional medio-alto, dependiendo las necesidad.
Resultados confiables en entornos controlados de luz.		Resultados muy confiables en entornos controlados de luz.	Respuestas confiables en entornos controlados de luz.
Se puede perder en la búsqueda al aplicarse cambios bruscos de luz.		Capacidad relativamente media para adaptarse a cambios de luz.	Se pierde con facilidad en cambios externos de luz
Resultado	bastante fiables, su tiempo de consecución depende en gran parte a la definición de la imagen.	Tiempo de consecución bajo dependiendo de la configuración de los parámetros internos.	Capacidad de predicción de la posición del objeto según la posición anterior y confiando en el histograma de la imagen. Tiene una función predefinida en OpenCV.

FUENTE : Ilbay Llangarí Luis Guido.

CAPÍTULO 3

3 SELECCIÓN DEL ALGORITMO MAS ADECUADO.

En este capítulo se detallarán la implementación de los algoritmos a evaluar. Sus resultados y los parámetros que se tomaron en cuenta para escoger al algoritmo más adecuado. Además se darán a conocer las ventajas y desventajas de los algoritmos en la práctica.

3.1 INTRODUCCIÓN.

Los algoritmos han sido seleccionados debido a su objetividad y potencialidad. Además se han tomado anteriores trabajos de investigación para como seguridad de que los algoritmos tienen potencialidad para funcionar con el Kinect. Entre muchos trabajos de investigación podemos citar los siguientes :

Percepción Activa para Seguimiento de Objetos en Entornos Urbanos, Adrián Gonzales Jimenez, Escuela Técnica Superior de Ingenieros (ETSI).

Algoritmo de seguimiento de objetos basado en visión asistida por computador en tiempo real utilizando CAMShift e histogramas ponderados, A. Hernández, R. Verrastro, L. Di Matteo, M. Prieto, M. Marufo, J. Gomez, C. Verrastro y (GIAR), Univ. Tecnológica Nacional Fac. Regional Bs. As., Buenos Aires, Argentina.

Se presentan a consideración para la investigación tres algoritmos:

- Algoritmo CAMShift.
- Algoritmo Coincidencia de Colores.
- Algoritmo por Umbralización.

3.2 MÉTODOS USADOS.

El análisis de las pruebas se hizo en base al método T STUDENT, ya que los resultados cuantitativos entre algoritmos son sencillos de comparar, pero complicados de incluir en algún método definitivo.

Para poder realizar las pruebas y el análisis se debe entender que la aplicación SIEMPRE puede detectar el objeto, lo cual supondría una eficiencia del 100%, sin embargo no es ese detalle el que se busca ponderar sino la precisión de los datos con respecto a la realidad.

Para realizar los cálculos estadísticos, se aprovecharon las herramientas del Microsoft Excel.

También se tomó en cuenta la cantidad de recursos computacionales que consume cada algoritmo en plena ejecución, el cual no se sujetara a comparaciones estadísticas sino cuánticas y servirán como guía para el investigador a orientarse por el algoritmo más adecuado para esta específica aplicación.

3.2.1 MEDICIÓN.

La medición es un proceso indispensable en la ciencia de donde se recogen datos sobre un experimento o sobre una magnitud física que puede ser comparada o evaluada según los propósitos de la aplicación, dependiendo de la naturaleza del sistema se pueden tomar gran cantidad de mediciones para tener una conclusión más fiel a la realidad.

La medición más usada se hace a través de un proceso directo. En el cual interviene una herramienta de medición. Sin embargo para nuestro caso se necesitará un medición indirecta. Ya que nuestro medidor, Kinect, no mide las unidades físicas que nosotros escogimos, sino cantidades de datos informáticos. Datos que nosotros tenemos que transformar con procesos matemáticos en cm.

3.2.1.1 ERRORES.

Son datos de estudio que nos ayudan a ver la exactitud y la precisión que tiene un proceso con respecto a una respuesta ideal. Se puede entender que un sistema debería tener los mínimos errores posibles para que su resultado o respuesta tenga la fiabilidad necesaria.

3.2.1.2 ERROR ABSOLUTO.

Es simple de medir y su fórmula se basa en una sustracción entre el valor real de una magnitud y el valor medido por el sistema.

$$EA = |P^* - P|$$

En donde:

P^* = Valor Real

P = Valor Obtenido

3.2.1.3 ERROR RELATIVO.

Es el resultado de la división en el error absoluto y el valor real. Si los multiplicamos por 100, podemos entenderlo como el porcentaje de error que tiene la medición y darnos una idea clave sobre su precisión.

$$ER = \frac{EA}{P}$$

$$ER = \frac{EA}{P} * 100$$

3.3 DESARROLLO DE LAS APLICACIONES.

Para realizar la evaluación se implemento cada uno de los algoritmos con los mismas herramientas y buscando todas el mismo fin. Obtener las tres coordenadas dimensionales de un objeto específico. Como se sugirió anteriormente se usaron los siguiente software:

- OpenCV
- C++
- CMake
- Vim
- Libfreenect

El trabajo se basa en el siguiente proceso para cada uno de los algoritmos propuestos:

Adquisición de la imagen.- Se realiza a partir del Kinect, se invoca a la imagen en RGB y la imagen Depth. Se necesitó de las dos imágenes para completar el proceso de tracking.

Algoritmo.- Se procesa la imagen y se realizan los cálculos correspondientes.

Resultado.- Como resultado se deberían tener dos cosas fundamentales: la imagen en tiempo real de la escena donde se muestre con algún signo o símbolo el seguimiento del

objeto y las coordenadas en tres dimensiones del objeto. Estos deben estar en una sola escala manejable.

FIGURA 3-I: DESARROLLO APLICATIVO DE LAS EVALUACIONES

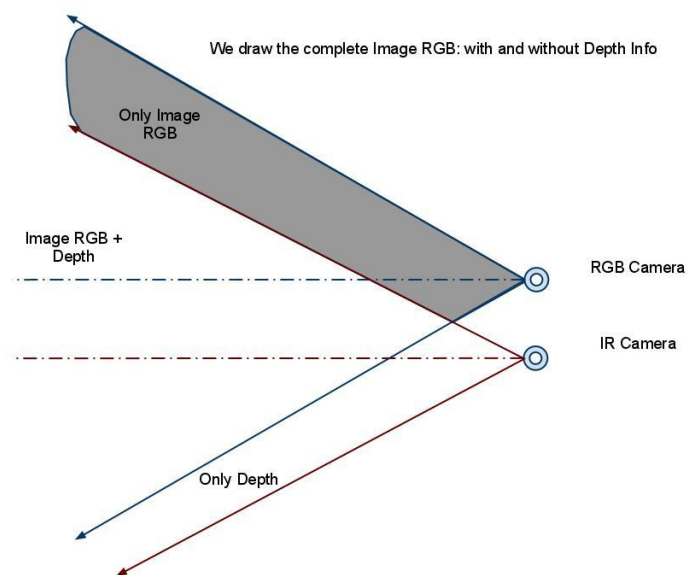


FUENTE: Ilbay Llangarí Luis Guido.

3.4 TRATAMIENTO DE LA VISIÓN DEL KINECT.

La "doble" visión del Kinect es el baluarte más apreciado de su construcción. Sin embargo, dentro de todas sus virtudes, es quizá el único problema que tiene, después de lograr su completo funcionamiento por código, ya que por razones de construcción la visión RGB y la Depth no son exactamente las mismas en cuanto a dimensión y posición. Ya que por defecto en las dos cámaras tienes distintas resoluciones y eso puede hacer caer en el error de los cálculos finales.

FIGURA 3-II: DOBLE VISIÓN DEL KINECT.



FUENTE: <https://landerpfc.files.wordpress.com/2011/03/verticalblackwideline.jpg>

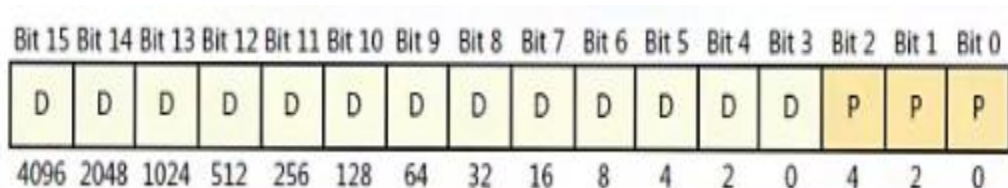
El tema de la resolución se puede resolver mediante el código del programa sin que esta llegue a ser un gran problema. Sin embargo, el darnos cuenta que las dos cámaras miran en posiciones distintas es algo que no se puede arreglar con tanta facilidad.

La calibración del Kinect se debe realizar de una forma similar a como lo hacen las cámaras convencionales, o más bien orientar el espacio de trabajo con un detalle capaz de no equivocarse los cálculos generados. O más bien los resultados se deben trabajar teniendo en cuenta esta peculiaridad.

3.5 OBTENCIÓN DEL DATO DE PROFUNDIDAD.

Ha sido el punto más importante de las tres aplicaciones. Pero el proceso que se ha llevado a cabo es el mismo en todas las ocasiones.

FIGURA 3-III : BITS DE INFORMACIÓN DEPTH



Fuente : USANDO SDK KINECT

Como entendemos de la FIGURA 3-III, la información que devuelve Kinect de su sensor infrarrojo tiene 16 bits de tamaño short. De los cuales los tres primeros son usados por el SDK Kinect para aplicaciones de reconocimiento esquelético. Por lo que han quedado otros 12 bits. Sin embargo, llegados a la práctica solo se necesitó los próximos 5 bits puesto que su información es de la suficiente profundidad necesaria para la investigación.

Ha sido necesario, entonces, analizar el comportamiento del Kinect en cada uno de aquellos rangos. La siguiente figura muestra resultados aproximados sobre pruebas de distancia.

ILUSTRACIÓN 3-IV : RANGO DE BITS - PROFUNDIDAD DEL KINECT

5	4	3	2	1	0
---	---	---	---	---	---

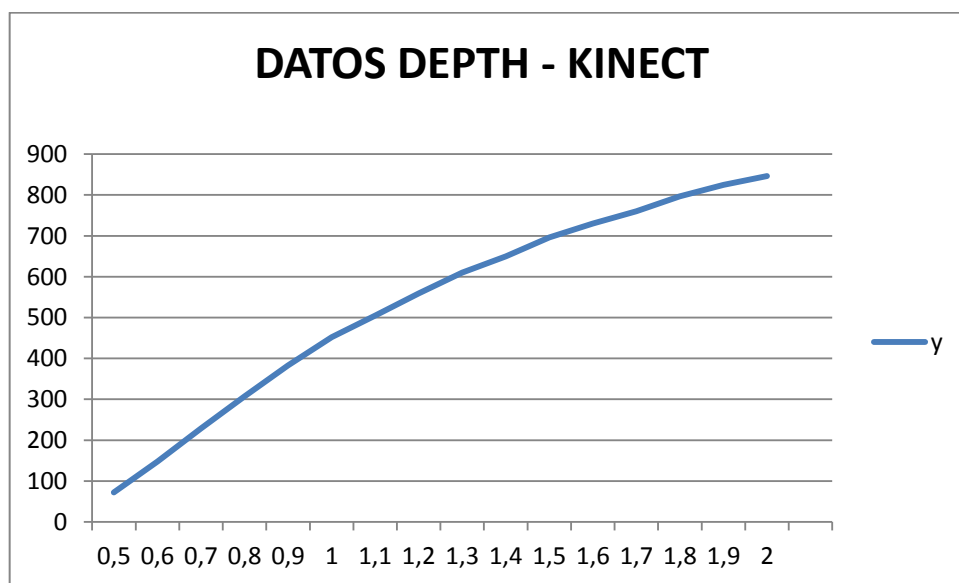
RANGO DE BITS	DISTANCIAS APROXIMADAS(cm)
0 [0 - 256]	[0 - 74]cm
1 [256 - 512]	[74 - 110] cm
2 [512 - 768]	[110 - 172]cm
3 [768 - 1024]	[172 - 328]cm
4 [1024 - ?]	-

FUENTE : Ilbay Llangarí Luis Guido.

3.5.1 FÓRMULA MATEMÁTICA DE TRANSFORMACIÓN.

Se debió tener en cuenta que la información de profundidad es tiene una variación más notoria en distancias cortas mientras que en distancias largas esta diferencia se torna lenta. Por lo que describe una curva no lineal a la que a simple análisis no se puede englobar en una sola fórmula.

ILUSTRACIÓN 3-V : DATOS BINARIOS PUROS DEPTH DEL KINECT.



FUENTE : Ilbay Llangarí Luis Guido.

La solución menos costosa y sencilla, ha sido generar una fórmula matemática para cada uno de los grupos de bits separados en la tabla anterior. Se usó las fórmulas de ecuación de línea dado que cada grupo de bits, muestra cierta linealidad. Los métodos numéricos no fueron ocupados pues su consecución ha presentado más errores que aciertos. El dato de profundidad será conocido como *pval*.

3.5.1.1 CASO 0.

Es cuando el objeto está más cercano al origen de visión. Por supuesto esto también tiene un rango inicial puesto que se ha preferido comenzar desde cierta distancia a contar el muestreo. Su fórmula es la siguiente:

$$cero = (0,130434 * pval) + 40,6086.$$

3.5.1.2 CASO 1.

Es el rango continuo. Su fórmula es la siguiente:

$$cero1 = (0,140625 * pval) + 38.$$

3.5.1.3 CASO 2.

Las distancias en este rango son más cortas entre el principio y el fin. Su fórmula es la siguiente:

$$cero2 = (0,242187 * pval) - 14.$$

3.5.1.4 CASO 3.

Es el último rango del que se ha realizado fórmula ya que su visión excede los tres metros de profundidad, superando nuestro espacio de trabajo escogido. Su fórmula es la siguiente:

$$cero3 = (0,531120332 * pval) - 235,9$$

3.5.1.5 CASO 4.

Es un caso *default* si el rango es irreconocible, pinta el pixel de color negro.

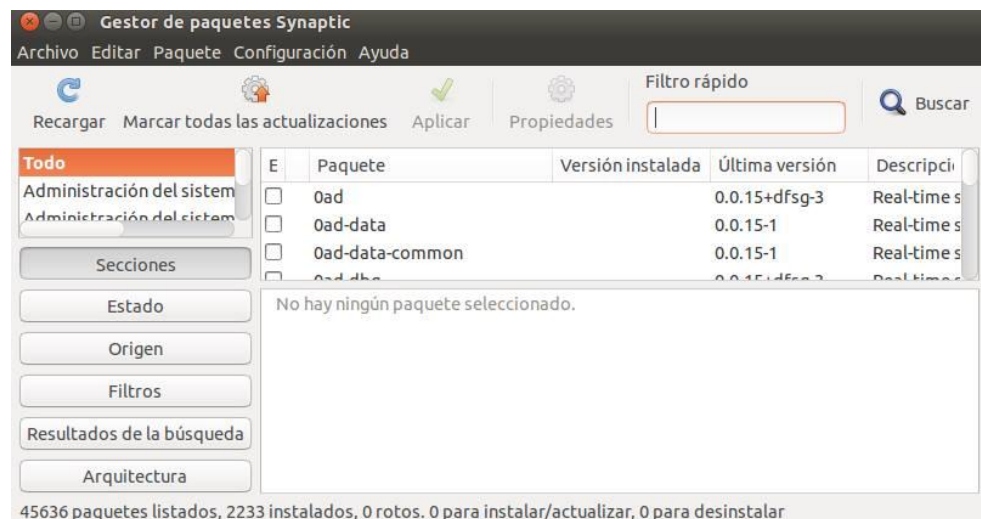
3.6 INSTALACIÓN DEL SOFTWARE REQUERIDO.

Debido a la singularidad del trabajo investigativo en Open Source fue necesario incluir un apartado en donde se muestre la instalación de las herramientas más importantes que intervinieron en el presente trabajo.

3.6.1 PROCESO INSTALACIÓN DE LIBFREENECT.

- Abrir el Gestor de paquetes Synaptic:

FIGURA 3-VI:SYNAPTIC DE UBUNTU 14.04.



FUENTE : Ilibay Llangarí Luis Guido.

Buscar y descargar :

- git core
- CMake
- freeglut3-dev
- pkg - config
- build - essential
- libxmu -dev
- libxi -dev
- libusb -1.0-0-dev

Abrir el terminal y ejecutar :

- `git clone https://github.com/OpenKinect/libfreenect.git`

Aparece el directorio Libfreenect. Se crea una carpeta build y en ella se compilan los ejecutables que vienen como demos en el Libfreenect, los cuales se encontrarán en la carpeta *bin*.

3.6.2 INSTALACIÓN DE OPENCV

El proceso resumido es el siguiente:

- Instalar los paquetes necesarios para el requerimiento del funcionamiento del OpenCV.

FIGURA 3-VII : INSTALACIÓN DE PAQUETES NECESARIOS PARA EL OPENCV

```
tesis@tesis:~$ sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev libjasper-dev libopenexr-dev cmake python-dev python-numpy python-tk libtbb-dev libeigen3-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev libqt4-opengl-dev sphinx-common texlive-latex-extra libv4l-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev default-jdk ant libvtk5-qt4-dev
```

FUENTE : Ilbay Llangarí Luis Guido.

- Una vez tenido esto, descargamos el directorio que contiene al OpenCV

FIGURA 3-VIII:COMPILACIÓN DEL PAQUETE OPENCV

```
tesis@tesis:~$ cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON -D WITH_VTK=ON ..
```

FUENTE : Ilbay Llangarí Luis Guido.

- Entonces el momento de la configuración indispensable para que el OpenCV trabaje correctamente.

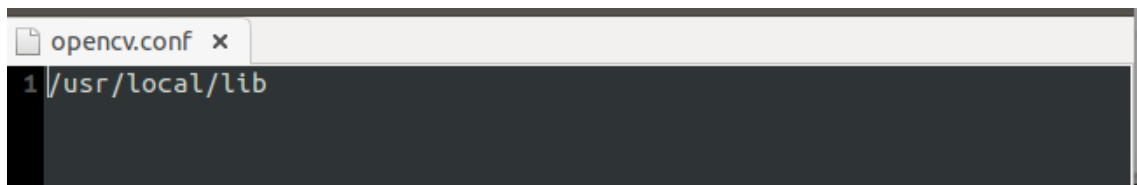
FIGURA 3-IX: CONFIGURACIÓN DEL OPENCV.

```
tesis@tesis:~$ sudo gedit /etc/ld.so.conf.d/opencv.conf
```

FUENTE : Ilbay Llangarí Luis Guido.

- El código anterior abriría un editor de texto, en el cual se debe guardar lo siguiente.

FIGURA 3-X: CONFIGURACIÓN DE LA RUTA DE LAS LIBRERÍAS INSTALADAS.



```
opencv.conf x
1 /usr/local/lib
```

FUENTE : Ilbay Llangarí Luis Guido.

- Escribimos el siguiente código para abrir otro editor de textos.

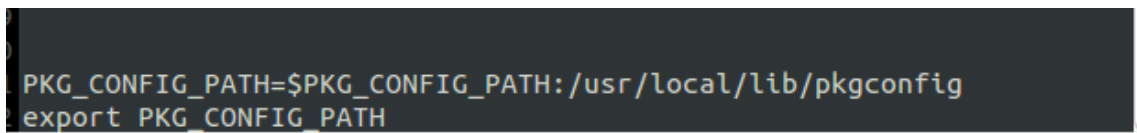
FIGURA 3-XI: CÓDIGO DE GENERACIÓN DE BASH.

```
tesis@tesis:~$ sudo gedit /etc/bash.bashrc
```

FUENTE : Ilbay Llangarí Luis Guido.

Y al final de lo ya escrito añadimos lo siguiente:

FIGURA 3-XII: CONFIGURACIÓN FINAL DEL PKG_CONFIG_PATH.



```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

FUENTE : Ilbay Llangarí Luis Guido.

Ahora tenemos el OpenCV totalmente configurado y se puede comprobar probando alguno de los ejemplos demo.

3.7 PRUEBA DEL ALGORITMO CAMSHIFT.

El algoritmo CAMShift posee la virtud de tratar de predecir la posición del objeto basado en la posición anterior. Su proceso de implementación es el siguiente.

- **Apertura de la cámara.-** El Kinect se apertura a partir del código del programa base. Se puede hacer el llamado de la imagen RGB pero por los recursos de software se necesita decidir la mejor decisión. Se definen los parámetros.

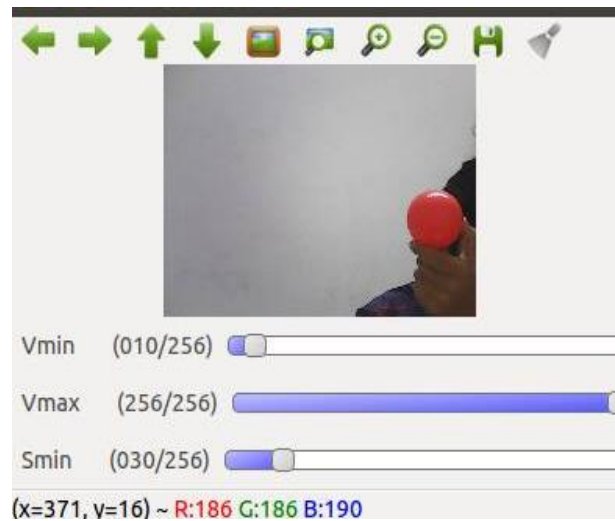
FIGURA 3-XIII: INIZIALIZACION Y DEFINICIÓN DE

```
Mat image;
bool backprojMode = false;
bool selectObject = false;
int trackObject = 0;
bool showHist = true;
Point origin;
Rect selection;
int vmin = 10, vmax = 256, smin = 30;
```

FUENTE : Ilbay Llangarí Luis Guido.

- **Obtención de la imagen.-** Este proceso es el primero que requiere el trabajo entre el OpenCV y el Libfreenect, ya que la imagen RGB que obtiene el Kinect tiene amplitudes exactas y definidas.

FIGURA 3-XIV:OBTENCIÓN DE LA IMAGEN.



FUENTE : Ilbay Llangarí Luis Guido.

- **Seleccionar la región de interés.-** se utiliza una función que permite que con el cursor se seleccione una región de interés en la imagen RGB. El Histograma selecciona solo extrae a partir del Hue.

FIGURA 3-XV: CÓDIGO DE LA OBTENCIÓN DEL ROI.

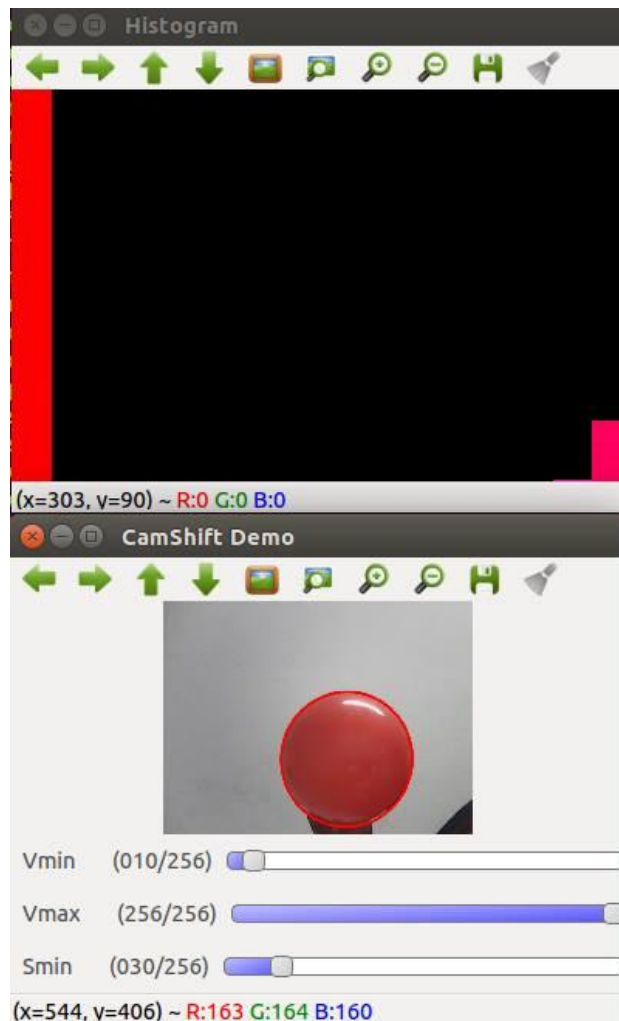
```
if( selectObject )
{
    selection.x = MIN(x, origin.x);
    selection.y = MIN(y, origin.y);
    selection.width = std::abs(x - origin.x);
    selection.height = std::abs(y - origin.y);

    selection &= Rect(0, 0, image.cols, image.rows);
}
```

FUENTE : Ilbay Llangarí Luis Guido.

- **Cálculo del histograma.-** Se realiza el cálculo del histograma a partir del ROI.

FIGURA 3-XVI:HISTOGRAMA DE LA IMAGEN Y EL OBJETO.



FUENTE : Ilbay Llangarí Luis Guido.

- **Seguimiento.-** Con el cálculo del backproject a partir del Histograma del color se crea un posible seguimiento del objeto.

FIGURA 3-XVII:HISTOGRAMA DEL OBJETO SEGUIDO.

```
Mat roi(hue, selection), maskroi(mask, selection);  
calcHist(&roi, 1, 0, maskroi, hist, 1, &hsize, &phranges);  
normalize(hist, hist, 0, 255, CV_MINMAX);
```

FUENTE : Ilbay Llangarí Luis Guido.

- **Filtro.-** Se aplica un filtro (Canny) para detectar los bordes para determinar con mejor precisión al objeto.

- **Respuesta.-** Se aplica una operación AND pixel por pixel entre las dos imágenes.

FIGURA 3-XVIII: OBJETO ENCONTRADO.



FUENTE : Ilbay Llangarí Luis Guido.

- **Llamado al Depth.-** El algoritmo presenta un problema en el llamado a la información del Depth. Como la posición no es constante debido a que el algoritmo también pretende predecir un posición futura, la clase que envía la respuesta Depth no responde de manera correcta al llamado. Cabe indicar que para los tres algoritmos se hace exactamente el mismo tipo de llamado, ya que es el desarrollado por la investigación, no es un proceso establecido sino uno creado.

3.7.1 RESUMEN DE RESULTADOS

Debido a la imposibilidad del llamado al cálculo Depth explicado anteriormente, no hemos podido encontrar las tres coordenadas requeridas.

Sin embargo, se ha podido realizar el seguimiento bidimensional y sobre esto se tienen algunas conclusiones:

- El algoritmo CAMShift está todo el tiempo tratando de ubicar un objeto con respecto al histograma anterior lo que suele causar muchas pérdidas de seguimiento, o el seguimiento de áreas que no corresponden solo al objeto.
- La selección del ROI, significa una gran ayuda y también de gran atractivo investigativo, quizá se pueda mejorar el algoritmo para que sea más robusto.

3.8 PRUEBA DE ALGORITMO DE COINCIDENCIA DE COLORES.

- **Apertura de la cámara.-** se debe aplicar el mismo llamado de función que se usó anteriormente pues el Kinect envía la misma imagen contenida en la misma clase del programa.

FIGURA 3-XIX: INICIALIZAR LA CÁMARA Y EL SENSOR.

```
device.startVideo();
device.startDepth();
```

FUENTE: Ilbay Llangarí Luis Guido.

- **Establecer el color y el umbral.-** En esta sección del código se menciona el color que se debe seguir y el umbral aceptable del seguimiento.

FIGURA 3-XX: ESTABLECIMIENTO DEL COLOR.

```
int red = 225;
int blue = 65;
int green = 225;
int tolerance = 50;
```

FUENTE : Ilbay Llangarí Luis Guido.

- **Transformación.-** en base a un cálculo pixel por pixel se convierte la imagen RGB a una binarizada en donde la respuesta debería mostrar una forma blanca del objeto a seguir.

FIGURA 3-XXI: COMPARACIÓN DEL PIXELADO POR COLOR.

```

for( i = 0; i < frame->height; i++)
  for( j = 0; j < frame->width; j++) /*Si els 3 canals compleixen la condició: blanc*/
    if( ((imgData[i*step+j*canals]) <= blue + tolerance && (imgData[i*step+j*canals]) >= blue - tolerance )
        && ((imgData[i*step+j*canals+1]) <= green + tolerance && (imgData[i*step+j*canals+1]) >= green - tolerance )
        && ((imgData[i*step+j*canals+2]) <= red + tolerance && (imgData[i*step+j*canals+2]) >= red - tolerance ))
      imgDataResult[i*result->widthStep+j*result->nChannels]=255;
    else imgDataResult[i*result->widthStep+j*result->nChannels]=0;

```

FUENTE : Ilbay Llangarí Luis Guido.

- **Cálculos.-** se escriben los cálculos matemáticos y lógicos para encontrar el objeto , realizar el seguimiento y dibujar una muestra de que el programa sigue al mismo.

FIGURA 3-XXII: CALCULAR LA POSICIÓN DEL OBJETO.

```

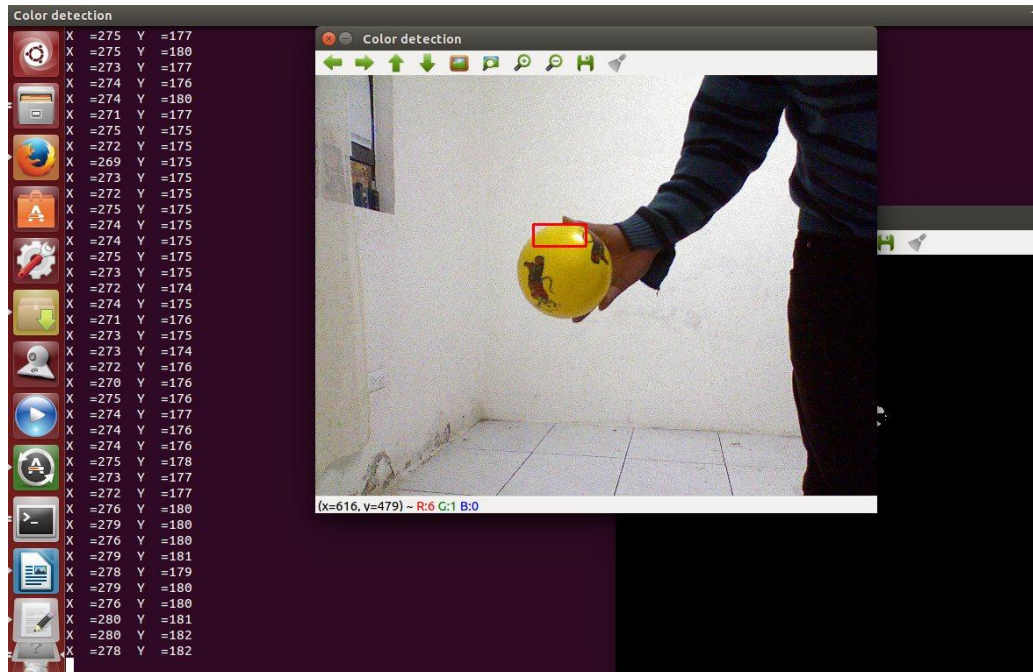
if(j < minX) minX = j;
if(j > maxX) maxX = j;
if(i < minY) minY = i;
if(i > maxY) maxY = i;

```

FUENTE: Ilbay Llangarí Luis Guido.

- **Seguimiento.-** Se puso a prueba un seguimiento de color amarillo y como se puede observar en la imagen, el algoritmo obtiene las coordenadas bidimensionales del objeto.

FIGURA 3-XXIII:SEGUIMIENTO DEL OBJETO.



FUENTE: Ilbay Llangari Luis Guido.

- **Llamada al DATO en del Depth.-** Se llama a la información que la función DEPTH está generando todo el tiempo y con ella se obtienen las tres coordenadas. Es necesario repetir que el llamado al dato DEPTH es el mismo para todos los algoritmos, su respuesta depende en su totalidad a las coordenadas resultantes en (X,Y)
- **Resultados.-** Evaluación de los resultados, estadísticamente.

3.8.1 TABLA DE RESULTADOS

TABLA 3-1 : MEDIDAS TOMADAS POR COINCIDENCIA DE COLORES.

DISTANCIA REAL(cm)	DISTANCIA OBTENIDA(cm)	ERROR ABSOLUTO	ERROR RELATIVO(%)
50	50,23	0,23	0,46
50	50,16	0,16	0,32
100	100,67	0,67	0,67
100	100,98	0,98	0,98
150	152,3	2,3	1,533333333
150	153,09	3,09	2,06
200	203,45	3,45	1,725
200	203,54	3,54	1,77
250	254,09	4,09	1,636
250	254,83	4,83	1,932

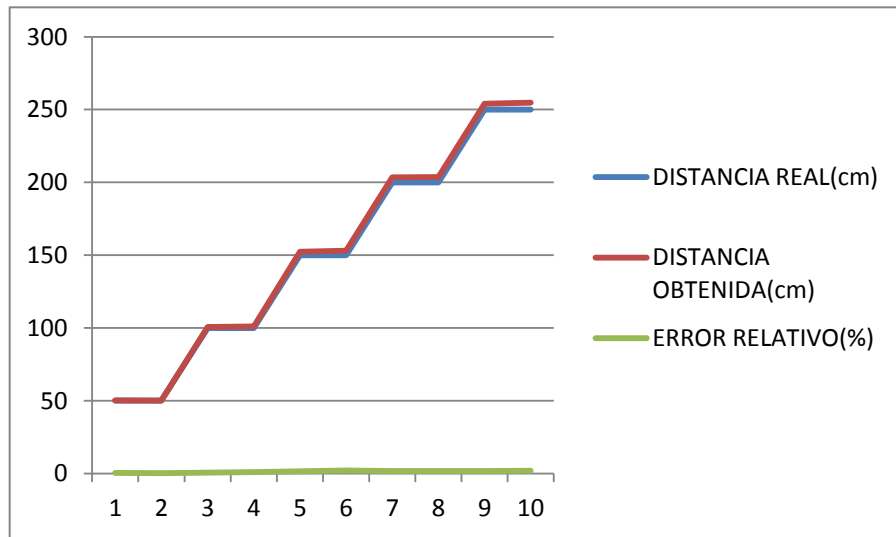
Promedio de Error

1,31

FUENTE : Ilbay Llangarí Luis Guido.

En la siguiente figura podemos observar un gráfico de las distancias. Se nota cómo la medición a compara se va separando entre mayor fue la distancia a medir. Esto se debe a que la información de profundidad que envía el Kinect no es lineal. Sin embargo se ha realizado la precisión que tiene el algoritmo es bastante aceptable.

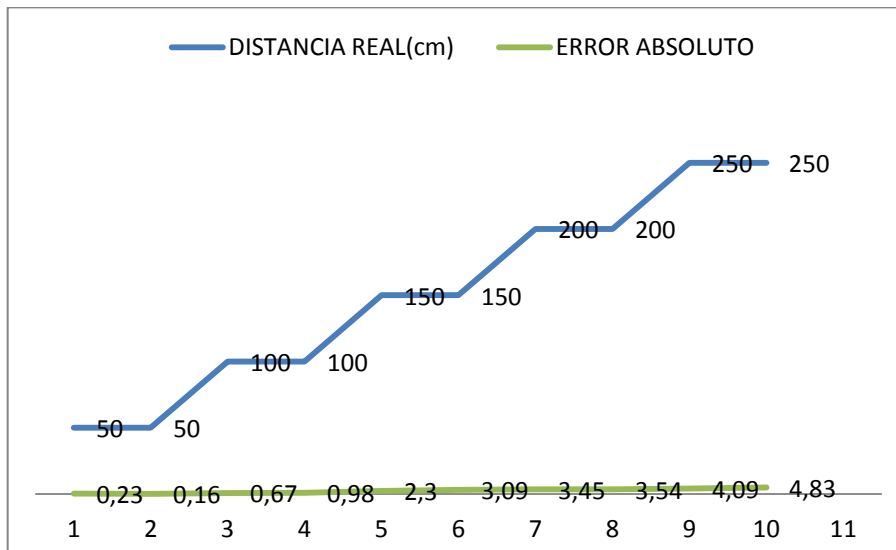
FIGURA 3-XXIV: DISTANCIA REAL, OBTENIDA Y ERROR DE ÉL ALGORITMO.



FUENTE: Ilbay Llangarí Luis Guido.

En el siguiente gráfico podemos observar el error en las distancias real y obtenida va creciendo entre mayor es la magnitud de la misma. Pero basado en el gráfico se puede afirmar que el error puede estar dentro de un rango admisible.

FIGURA 3-XXV: DISTANCIA REAL VS ERROR ABSOLUTO.



FUENTE : Ilbay Llangarí Luis Guido.

3.8.2 CONCLUSIONES

- El algoritmo funciona de buena manera con el color definido, aunque existe esa ambigüedad en la exactitud del color.
- La impresión de las coordenadas no es tan rápida, quizá debido al proceso que debe cumplir, aunque si retrase es imperceptible de notar.
- El algoritmo se pierde en cuanto a la iluminación por lo que se debe tener una fuente de luz continua y evitar los cambios bruscos.
- Entre más lejano es el objeto, la medición tiende a tener un error relativo más alto.

3.9 PRUEBA DE ALGORITMO DE UMBRALIZACIÓN.

En esta sección se detallará los puntos importantes en la implementación del presente algoritmo.

3.9.1 PROCESO DE IMPLEMENTACIÓN.

- **Apertura de la cámara.-** El mismo proceso de llamado en todos los algoritmos. Se realiza haciendo el llamado a las clases que apertura el video y la imagen Depth. El siguiente código es el que sirve para abrir las dos cámaras y hacerlas funcionar, pero para tener el resultado de lo que esas cámaras proyectan se necesita llamar esas respuestas en imprimir en ventanas.

FIGURA 3-XXVI: INICIALIZACIÓN DE LA CÁMARA Y EL SENSOR.

```
device.startVideo();  
device.startDepth();
```

FUENTE: Ilbay Llangarí Luis Guido.

- **Invocación de la imagen RGB y Depth.-** Se llama a las dos Fuentes de información y se alista una segunda imagen para los resultados.

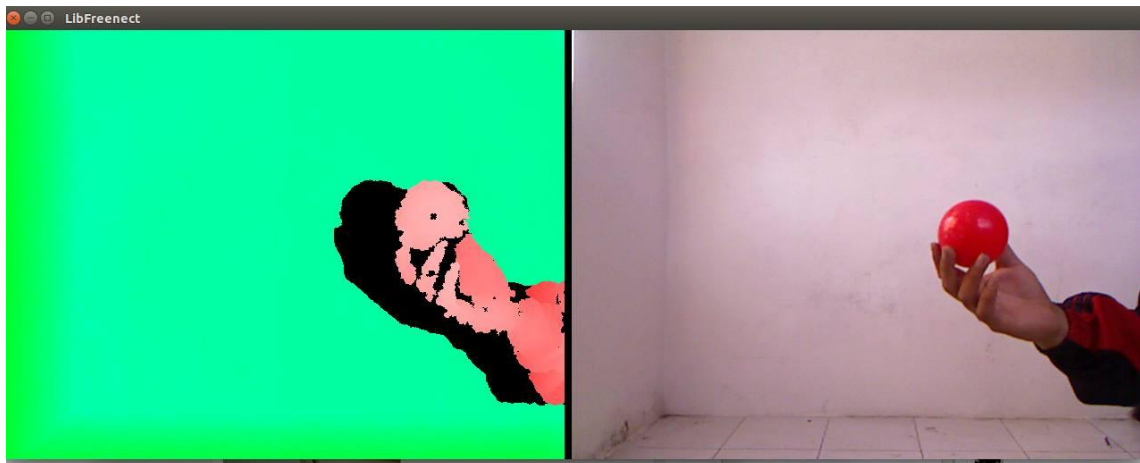
FIGURA 3-XXVII : LLAMADA A LOS DATOS DE LOS DOS VIDEOS.

```
void VideoCallback(void* _rgb, uint32_t timestamp) {
m_rgb_mutex.lock();
//Mutex::ScopedLock lock(m_rgb_mutex);
uint8_t* rgb = static_cast<uint8_t*>(_rgb);

void DepthCallback(void* _depth, uint32_t timestamp) {
m_depth_mutex.lock();
```

FUENTE: Ilbay Llangarí Luis Guido.

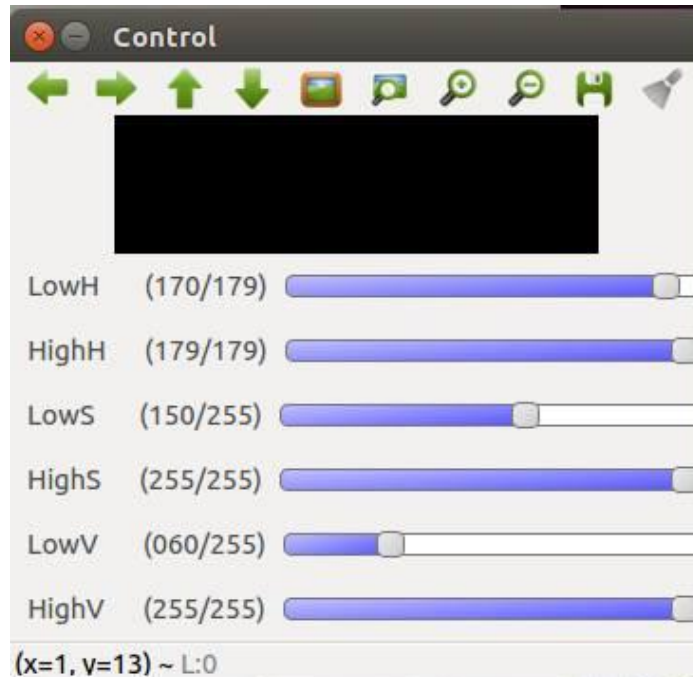
FIGURA 3-XXVIII: IMAGEN DEPTH VS RGB



FUENTE : Ilbay Llangarí Luis Guido.

- **Parámetros.-** Se crean y establecen parámetros movibles mediante el uso de trackbars. La ventana puede variar los número dentro de cada rango en el HSV, aunque en la primera expresión el rango ya está definido para seguir un solo color, el cual solo puede ser variado así mismo .

ILUSTRACIÓN 3-XXX: TRACKBARS DE LOS PARÁMETROS.



FUENTE: Ilbay Llangarí Luis Guido.

- **Conversión.-** Parte importante en donde se convierte la imagen RGB a HSV y dados los parámetros anteriores se procede a obtener una nueva imagen.

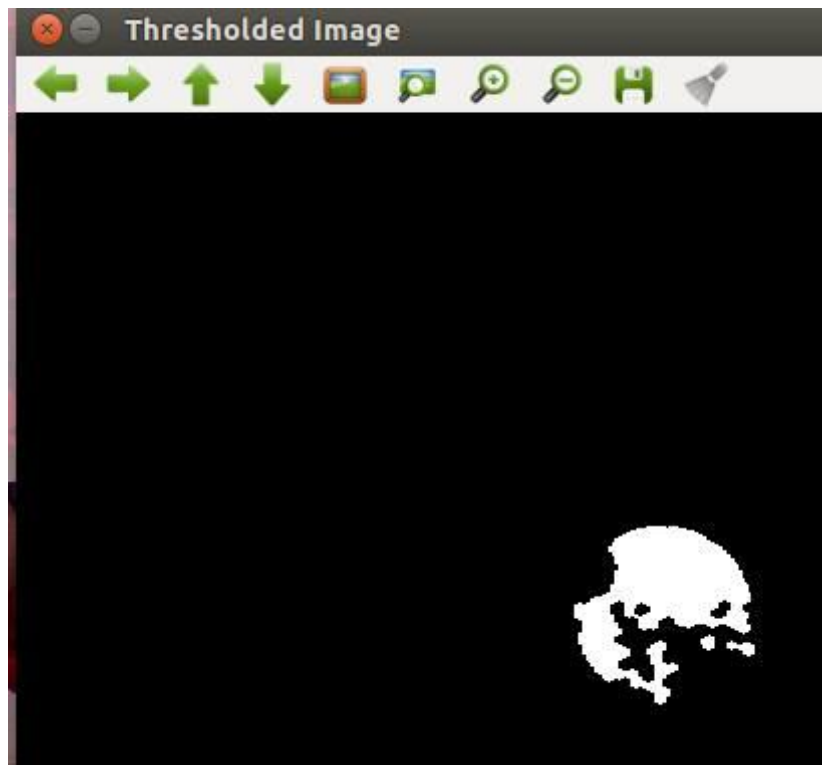
FIGURA 3-XXXI : CONVERSIÓN DE FORMATO DE IMAGEN.

```
cvtColor(rgbMat, imgHSV, COLOR_BGR2HSV);
```

FUENTE : Ilbay Llangarí Luis Guido.

- **Umbralización y transformaciones Morfológicas.-** Mediante código se procede a realizar la Umbralización bajo los parámetros de los trackbars propuestos. Las transformaciones morfológicas se realizan para eliminar todos los ruidos que podría aún tener la imagen anterior.

FIGURA 3-XXXII : IMAGEN UMBRALIZADA.



FUENTE: Ilbay Llangarí Luis Guido.

- **Cálculos de Posición X y Y.**

FIGURA 3-XXXIII :CÁLCULO DE LA POSICIÓN DEL OBJETO

```
Moments oMoments = moments (imgThresholded);  
double dM01 = oMoments.m01;  
double dM10 = oMoments.m10;  
double dArea = oMoments.m00;  
  
posX = dM10/dArea;  
posY = dM01/dArea;
```

FUENTE: Ilbay Llangarí Luis Guido.

- **Cálculos de posición z.** El siguiente código expresa la condición básica para obtener la información del Depth necesaria. Si el programa envía los datos del Depth de cada píxel se tomaría un tiempo exagerado y el gasto computacional subiría en demasía. Por lo que se genera la condición mostrada en el código, para solo pedir el dato de la coordenada (X,Y). Entonces solo responde con pocos datos y el programa puede controlarse sin problemas.

FIGURA 3-XXXIV : CÓDIGO CONDICIÓN DE MUESTREO EN EL DEPTH

```
//printf("r1 x-> %d ", pval);  
//printf("r2 y-> %d\n ", lb);  
  
if((j>posY && j<posY+12)&&  
(k>posX && k<posX+12))
```

FUENTE : Ilbay Llangarí Luis Guido.

3.9.2 MEDICIÓN RESULTADOS.

Para obtener resultados a partir de la aplicación primero hubo que obtener mediciones físicas de forma directa de tal forma que se puede tener base de comparación. En vista de la gran cantidad de información posible requerida se han tomado solo 10 muestras las cuales puedan ser comparadas con los resultados digitales del tracking.

TABLA 3-2: RESULTADOS PRUEBA ALGORITMO DE UMBRALIZACIÓN.

DISTANCIA REAL(cm)	DISTANCIA OBTENIDA(cm)	ERROR ABSOLUTO	ERROR RELATIVO(%)
50	50,15	0,15	0,3
50	51,09	1,09	2,18
100	98,78	1,22	1,22
100	100,54	0,54	0,54
150	152,51	2,51	1,673333333
150	152,09	2,09	1,393333333
200	202,13	2,13	1,065
200	201,5	1,5	0,75
250	254,3	4,3	1,72
250	252,2	2,2	0,88

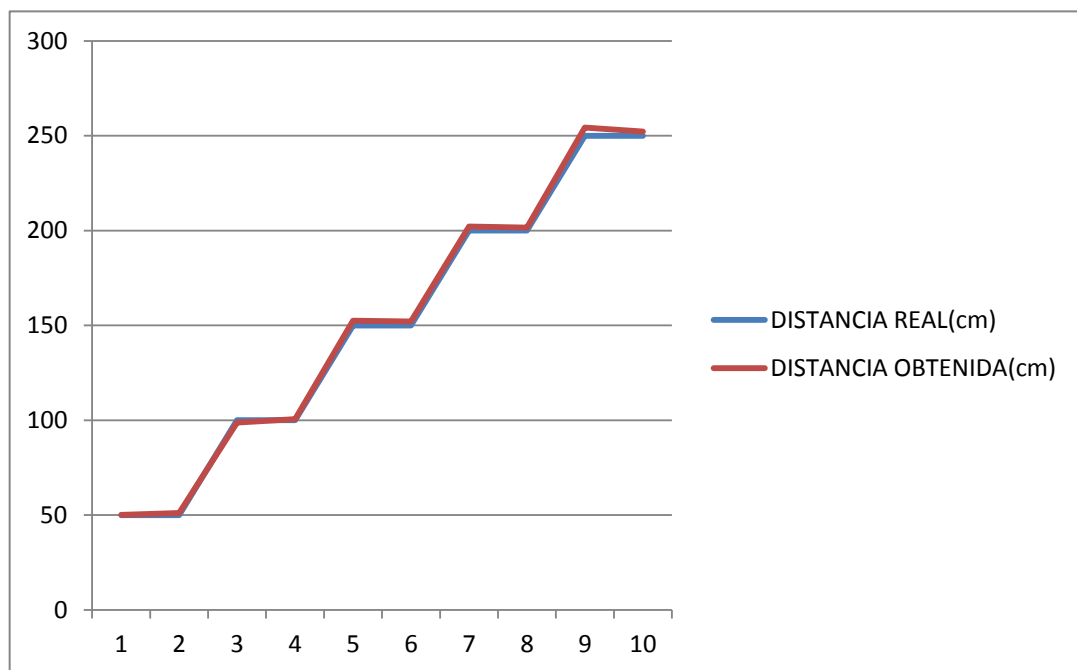
Promedio de Error

1,17

FUENTE : Ilbay Llangarí Luis Guido.

En el siguiente gráfico de resultados podemos observar el mismo fenómeno anterior, aunque con el detalle de que el primer dato obtenido es quien más se aleja del real, mientras que el segundo se apega mucho al valor ideal. Esto se da, porque el primer dato impreso indica el límite del rango del visión impuesta por el código y el segundo indica el dato obtenido en el rango medio, debido a la gran cantidad de impresiones en tiempo real, se ha reducido en lo posible el rango de muestreo.

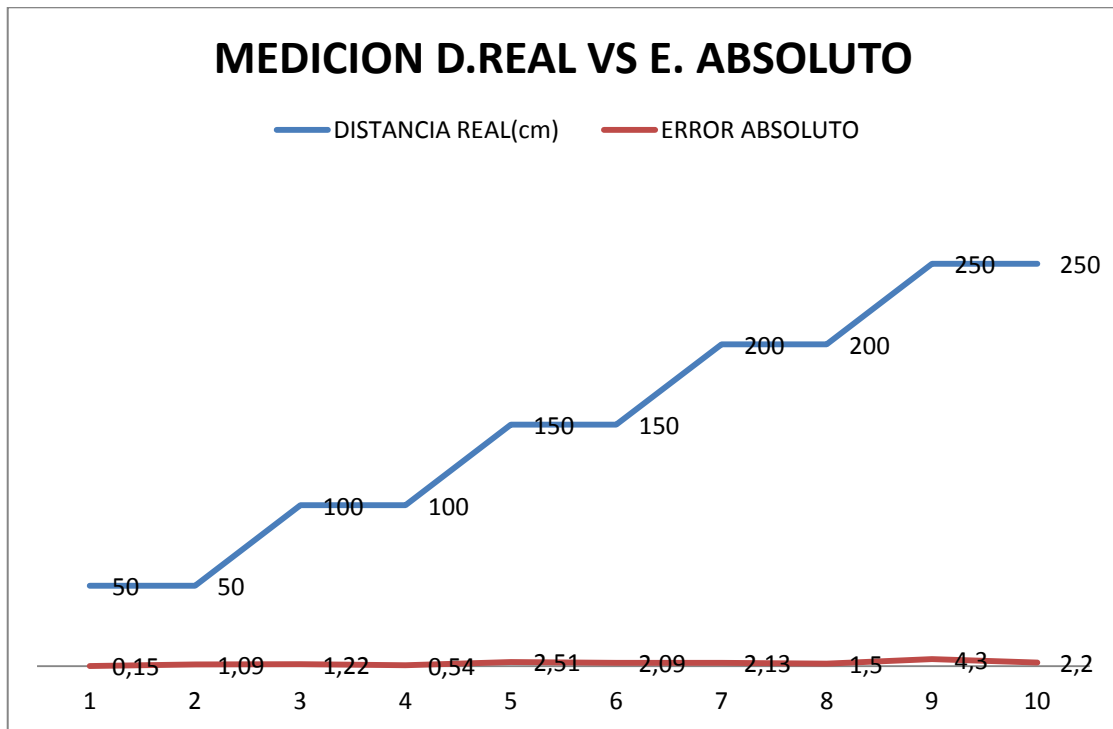
FIGURA 3-XXXV : DISTANCIA REAL VS DISTANCIA OBTENIDA.



FUENTE: Ilbay Llangarí Luis Guido.

En la siguiente gráfica podemos ver el nivel de error en tanto va aumentando la magnitud de la distancia. Podemos verificar que el algoritmo tiene un error ligeramente menor al anterior.

FIGURA 3-XXXVI: MEDICIÓN REAL VS ERROR ABSOLUTO.



FUENTE : Ilbay Llangarí Luis Guido.

3.9.3 CONCLUSIONES DE RESULTADOS.

- La impresión de las coordenadas se produce de forma inmediata.
- Suele perderse en algunos puntos ciegos de la imagen.
- Los resultados son bastantes fiables.

3.10 COMPROBACIÓN DE LA HIPÓTESIS.

Luego de realizar el mismo trabajo con los tres algoritmos se han obtenido muchas respuestas reveladoras. Según la hipótesis una evaluación de los algoritmos nos haría indicar con acierto al más adecuado para utilizar en el resto del proyecto.

3.10.1 ESTUDIO COMPARATIVO DE RESULTADOS

Para el análisis comparativo de los resultados y la comprobación de la hipótesis se ha usado el método T-STUDENT, con un nivel de confianza de 95%, el cual es el más usado para comprobación de hipótesis. Los datos se han procesado en el software R. Se han realizado 30 muestras para cada algoritmo.

3.10.1.1 PRUEBA ALGORITMO CAMSHIFT

Para las pruebas del algoritmo CAMShift se han ocupado diferentes escenas, en las cuales las condiciones de luz también han variado. Se han propuesto rangos ideales de medición para su comparación posterior , dando los siguientes resultados.

TABLA 3-3: PRUEBAS ALGORITMO CAMSHIFT

x	y	z	Xo	Yo	Zo
1	1	100	1	1	0
1	1	100	2	2	0
1	1	100	2	3	0
1	1	100	1	4	0
1	1	100	2	2	0
1	1	100	4	5	0
1	1	100	2	3	0
1	1	100	3	1	0
1	1	100	2	4	0
1	1	100	1	5	0
1	1	100	3	2	0
1	1	100	4	2	0
1	1	100	1	4	0
1	1	100	4	1	0
1	1	100	1	0	0
1	1	100	1	2	0
1	1	100	3	5	0
1	1	100	2	1	0
1	1	100	2	3	0
1	1	100	0	2	0

1	1	100	2	1	0
1	1	100	4	2	0
1	1	100	2	3	0
1	1	100	3	4	0
1	1	100	1	3	0
1	1	100	2	0	0
1	1	100	1	0	0
1	1	100	2	2	0
1	1	100	2	2	0
1	1	100	1	2	0

continua

FUENTE: Ilbay Llangarí Luis Guido .

Para analizar estos datos mediante el método T-STUDENT, es necesario primero demostrar la distribución normal de los mismos. Para ello se ha usado el test de *shapiro-wilk* Obteniendo los siguientes resultados.

Para el dato X:

W=0,8818.

A un nivel de confianza de 95% y con un número de muestras n=30, el valor crítico para este ítem es 0,927. Entonces, debido a que el valor estadístico W es menor al valor crítico se puede concluir que X **no** lleva una distribución normal.

Para el dato Y:

W=0,9771.

A un nivel de confianza de 95% y con un número de muestras n=30, el valor crítico para este ítem es 0,927. Por lo tanto, como el valor estadístico W es mayor al valor crítico se puede concluir que Y lleva una distribución normal.

Para el dato Z:

Debido a la imposibilidad del algoritmo para devolver este valor no se puede realizar los cálculos anteriores.

Se notó que el rango X al no llevar una distribución normal y el rango Z al no tener datos disponibles para su estudio, ha sido imposible continuar con el método T-STUDENT. De tal manera que el presente algoritmo ha quedado exento de posteriores análisis.

3.10.1.2 PRUEBA ALGORITMO UMBRALIZACIÓN.

Para la consecución de esta prueba se han tenido en cuenta los mismos parámetros anteriormente planteados. Dando como resultado las siguientes tablas.

TABLA 3-4. PRUEBAS ALGORITMO UMBRALIZACIÓN

x	y	z	Xo	Yo	Zo
1	1	100	0	5	98,88
1	1	100	3	1	99,88
1	1	100	2	3	98,02
1	1	100	4	5	100,02
1	1	100	2	2	98,02
1	1	100	3	4	99,31
1	1	100	2	2	101,01
1	1	100	3	2	99,59
1	1	100	3	4	101,28
1	1	100	1	5	103,56
1	1	100	2	3	102,22
1	1	100	4	2	102,22
1	1	100	-1	1	101,84
1	1	100	4	1	99,59
1	1	100	1	0	100,84
1	1	100	1	2	99,88
1	1	100	3	4	99,88
1	1	100	2	1	99,88
1	1	100	2	3	101,56
1	1	100	0	2	98,59
1	1	100	1	1	99,59
1	1	100	4	2	100,59

1	1	100	2	2	99,59
1	1	100	3	4	101,88
1	1	100	1	3	99,88
1	1	100	2	2	100,02
1	1	100	1	0	100,02
1	1	100	3	2	99,88
1	1	100	2	0	101,02
1	1	100	3	3	100,02

PRUEBAS ALGORITMO UMBRALIZACIÓN

continua

FUENTE: Ilbay Llangarí Luis Guido

Del mismo modo, fue necesario comprobar la distribución que tiene cada uno de los apartados.

Para el dato X:

W=0,9292.

A un nivel de confianza de 95% y con un número de muestras $n=30$, el valor crítico para este ítem es 0,927. Entonces, debido a que el valor estadístico W es menor al valor crítico se puede concluir que X **si** lleva una distribución normal.

Para el dato Y:

W=0,92716.

A un nivel de confianza de 95% y con un número de muestras $n=30$, el valor crítico para este ítem es 0,927. Entonces, debido a que el valor estadístico W es menor al valor crítico se puede concluir que X **si** lleva una distribución normal.

Para el dato Z:

W=0,9491.

A un nivel de confianza de 95% y con un número de muestras $n=30$, el valor crítico para este ítem es 0,927. Entonces, debido a que el valor estadístico W es menor al valor crítico se puede concluir que X **si** lleva una distribución normal.

Una vez determinado las inclinaciones de los ejes tridimensionales, se procede a aplicar la prueba T-STUDENT, de esta forma se puede comprobar o desechar la hipótesis la cual es:

H₀: El Algoritmo por Umbralización es el adecuado para realizar el tracking 3D de objetos, aplicado a la simulación de un brazo robótico, mediante Kinect.

En caso de no comprobarse la hipótesis nula, se plantea tiene la hipótesis alternativa:

H₁: El Algoritmo por Umbralización no es el adecuado para realizar el tracking 3D de objetos, aplicado a la simulación de un brazo robótico, mediante Kinect.

Para comprobar si la diferencia de datos llega a ser importante se comparó la media de los datos ideales contra la media de los datos obtenidos. Se han obtenido los siguientes resultados.

Para el Eje X se obtiene:

$$t = -4,74782.$$

A un nivel de confianza de 95% y con GDL=29, el valor crítico para este ítem es **2,045**. Entonces, debido a que el valor estadístico t es menor que el valor crítico se acepta la hipótesis nula para este eje.

Para el Eje Y se obtiene:

$$t = -5,16269.$$

A un nivel de confianza de 95% y con GDL=29, el valor crítico para este ítem es 2,045. Entonces, debido a que el valor estadístico t es menor que el valor crítico se acepta la hipótesis nula para el eje Y.

Para el Eje Z se obtiene:

$$t = -1,23402.$$

A un nivel de confianza de 95% y con GDL=29, el valor crítico para este ítem es 2,045. Entonces, debido a que el valor estadístico t es menor que el valor crítico se acepta la hipótesis nula para el eje Z.

INTERPRETACIÓN

Dado a que la hipótesis nula se ha comprobado para los tres ejes se puede afirmar que el algoritmo por Umbralización es el adecuado para el tracking 3D, aplicado a la simulación de un brazo robótico, mediante Kinect.

3.10.1.3 PRUEBA DE ALGORITMO POR COINCIDENCIA DE COLORES.

Se obtuvieron los siguientes resultados.

TABLA 3-5: PRUEBA ALGORITMO DE COLORES

x	y	z	Xo	Yo	Zo(cm)
1	1	100	1	1	102,56
1	1	100	1	1	98,82
1	1	100	1	2	98,02
1	1	100	1	1	100,02
1	1	100	1	2	99,02
1	1	100	2	4	99,31
1	1	100	2	3	103,56
1	1	100	2	5	99,59
1	1	100	2	3	101,28
1	1	100	2	5	99,56
1	1	100	2	3	102,22
1	1	100	2	2	102,22
1	1	100	2	5	101,84
1	1	100	2	1	99,59
1	1	100	2	0	100,84
1	1	100	2	2	99,88
1	1	100	2	4	99,88
1	1	100	2	1	99,88
1	1	100	2	4	101,56
1	1	100	2	2	98,59
1	1	100	2	1	99,59
1	1	100	2	2	100,59

1	1	100	2	2	99,59
1	1	100	3	4	101,88
1	1	100	1	3	99,88
1	1	100	2	2	100,02
1	1	100	1	0	100,02
1	1	100	3	2	99,88
1	1	100	2	0	101,02
1	1	100	3	3	100,02

FUENTE: Ilbay Llangarí Luis Guido.

Nuevamente ha sido necesario comprobar la distribución de los ejes:

Para el dato X:

W=0,7380.

A un nivel de confianza de 95% y con un número de muestras $n=30$, el valor crítico para este ítem es 0,927. Entonces, debido a que el valor estadístico W es menor al valor crítico se puede concluir que X **no** lleva una distribución normal.

Para el dato Y:

W=0,9270.

A un nivel de confianza de 95% y con un número de muestras $n=30$, el valor crítico para este ítem es 0,927. Entonces, debido a que el valor estadístico W es menor al valor crítico se puede concluir que X **si** lleva una distribución normal.

Para el dato Z:

W=0,9400.

A un nivel de confianza de 95% y con un número de muestras $n=30$, el valor crítico para este ítem es 0,927. Entonces, debido a que el valor estadístico W es menor al valor crítico se puede concluir que X **si** lleva una distribución normal.

Ahora procedemos a implementar la prueba T-STUDENT para la comprobación de la hipótesis la que en este caso sería:

H₀: El Algoritmo por Colores es el adecuado para realizar el tracking 3D de objetos, aplicado a la simulación de un brazo robótico, mediante Kinect.

En caso de no comprobarse la hipótesis nula, se plantea tiene la hipótesis alternativa:

H₁: El Algoritmo por Colores no es el adecuado para realizar el tracking 3D de objetos, aplicado a la simulación de un brazo robótico, mediante Kinect.

Para el eje X, no es posible realizar la prueba por que sus datos estaban ligeramente fuera de la distribución normal.

Para el Eje Y se tiene:

t= -4,96798.

A un nivel de confianza de 95% y con GDL=29, el valor crítico para este ítem es 2,045. Entonces, debido a que el valor estadístico t es menor que el valor crítico se acepta la hipótesis nula para el eje Y.

Para el Eje Z se tiene:

t= -1,52256.

A un nivel de confianza de 95% y con GDL=29, el valor crítico para este ítem es 2,045. Entonces, debido a que el valor estadístico t es menor que el valor crítico se acepta la hipótesis nula para el eje Z.

INTERPRETACIÓN

Aunque el algoritmo ha logrado comprobar la hipótesis planteada en dos ejes, el eje X y su falta de distribución normal han obligado a desechar la hipótesis nula y a adoptar la hipótesis alternativa en la cual el algoritmo por Colores no es el adecuado.

En base a todos los datos analizados y procesados anteriormente se ha podido comprobar que una evaluación de algoritmos Tracking 3D , ha permitido escoger al **Algoritmo de Umbralización** como el más adecuado para el Tracking aplicado a la simulación de un brazo robótico, mediante Kinect.

CAPITULO 4

4 SIMULACIÓN DEL BRAZO ROBÓTICO.

En el presente capítulo se va a realizar el estudio de movimiento robótico que alcance al objeto con base a las coordenadas encontradas en el tracking. Para el movimiento robótico se ha realizado un previo estudio de simulación con la herramienta Matlab debido a sus prestaciones antes mencionadas.

4.1 ESPACIO REAL DE TRABAJO PROYECTADO.

Una de las más atractivas características del Kinect es su gran espacio de trabajo, el mismo que se extiende hacia varios metros en la profundidad. Sin embargo se tiene que limitar un tamaño de trabajo finito y no tan extenso por las siguiente razones:

4.1.1 CALIDAD DE INFORMACIÓN

La fiabilidad de la información que devuelven los sensores del Kinect, en especial el infrarrojo, se torna menos confiable y más difícil de análisis en cuanto más lejano es el

espacio de censado. Esto sucede porque la variación de bits para escribir la información se torna más estático cuanto más lejos se ve.

4.1.2 PROPÓSITOS DE LA INVESTIGACIÓN

La presente investigación es un estudio confirmativo y simulado de lo que vendría después. Entonces, si se toma un espacio de trabajo demasiado grande, la implementación de un brazo robótico físico se tornara demasiado costoso , así como el desarrollo del software.

4.2 ESTUDIO MATEMÁTICO - GEOMÉTRICO DEL BRAZO ROBÓTICO

Por su potencia al momento de resolver problemas matemático se usó el software matlab.

Como ya se ha dicho se optó por un brazo robótico de 4 GDL y el método es cinemática inversa hablando del método geométrico antes mencionado.

Una vez halladas las variables se procede a escribirlas en un programa de matlab, para ver la simulación.

El primer paso a realizar es definir las dimensiones de los eslabones. Como este es un brazo de 4 grados de libertad, necesitamos tres eslabones y cuatro articulaciones en total.

Tomaremos un punto inicial en el espacio tridimensional y haremos que el brazo se mueva a un punto final.

Proceso detallado en la siguiente imagen.

FIGURA 4-I : ESTABLECIMIENTO DE PARÁMETROS DEL MOVIMIENTO.

```
pi    = atan(1)*4;  
rad   = pi/180;  
grad  = 180/pi;  
t1=2;  
t2=6;  
int=0.1;  
extrem1=14;  
extrem2=6;  
int2=-0.1;  
yini=-2;  
w1   = -2 : 0.1 :6;  
fw1  = 14 : -0.1 : 6;  
gw1  = 0 : (2/80): 2;  
top= (abs(t1)+abs(t2))/0.1)+1;
```

FUENTE : Ilbay Llangarí Luis Guido.

El primer paso es definir las variables y constantes necesarias. Matlab realiza los cálculos trigonométricos en radianes. Por lo que algunos resultados tienen que ser transformados para que las funciones realicen los cálculos correctos.

Posterior a esto, presentamos los puntos iniciales del brazo robótico. Y el recorrido que debe hacer para llegar al punto final. Si notamos, los ejes (X, Y, Z) tienen definida su trayectoria a partir de vectores unidimensionales cuyas distancias relativas están definidas en el primer punto.

FIGURA 4-II : PUNTO INICIAL Y ESPACIO DE MOVIMIENTO.

```
BaseX = 0;  
BaseY = 0;  
Cabeceo = 90;  
gw = 0 : (2/80): 2;  
x = w(1, i);  
y = 14 - fw(1, i);  
z = gw(1, i);  
xx = atan(abs(z)/abs(w)); |
```

FUENTE : Ilbay Llangarí Luis Guido.

Ahora se realizan los cálculos trigonométricos. Para que se pueda notar el movimiento de forma detallada fue preciso anotar el movimiento de forma lenta e imprimir resultados en intervalos cortos, los cuales están definidos en los vectores del primer punto.

FIGURA 4-III : OBTENCIÓN DEL ÁNGULO DE GIRO.

```
Puntox1=(tan(xx)*BrazoPX);  
Puntox2=(tan(xx)*AntBrPX);  
Puntox3=(tan(xx)*MunecPX);
```

FUENTE: Ilbay Llangarí Luis Guido.

FIGURA 4-IV : ÁNGULOS DE TODOS LOS ESLABONES.

```
Afx = cos(rad*Cabeceo)*LongMunec;
LadoB = x-Afx;

Afy = sin(rad*Cabeceo)*LongMunec;
LadoA = y-Afy-BaseY;

Hipotenusa = ( (LadoA^2)+(LadoB^2) )^(1/2);

Alfa = atan2(LadoA, LadoB);
Beta = acos( ((LongBrazo^2)-(LongAntBr^2)+(Hipotenusa^2))/(2*LongBrazo*Hipotenusa) );

AngBrazo = Alfa + Beta;
AngBrazo*grad

Gamma = acos( ((LongBrazo^2)+(LongAntBr^2)-(Hipotenusa^2))/(2*LongBrazo*LongAntBr) );
AngAntBr = -((180*rad)-Gamma);
(AngAntBr*grad) 180

AngMunec = ((rad*Cabeceo)-AngBrazo-AngAntBr);
(AngMunec*grad) 180
```

FUENTE : Ilbay Llangarí Luis Guido.

Una vez conseguidos los ángulos necesarios para la representación de cada punto ahora es necesario calcular las coordenadas de todos los eslabones en cada instancia del movimiento.

FIGURA 4-V : PUNTOS EN EL ESPACIO DE LAS ARTICULACIONES

```
PXa=LongBrazo* cos (AngBrazo);  
PYa=LongBrazo*-sin (AngBrazo);  
  
PXb=LongAntBr* cos (AngAntBr+AngBrazo);  
PYb=LongAntBr*-sin (AngAntBr+AngBrazo);  
  
PXc=LongMunec* cos (AngMunec+AngAntBr+AngBrazo);  
PYc=LongMunec*-sin (AngMunec+AngAntBr+AngBrazo);
```

FUENTE : Ilbay Llangarí Luis Guido.

Entonces se encuentran los puntos con los cuales ya se puede proceder a imprimir los resultados. Cabe recalcar que todos los procesos matemáticos se hallan en una función aparte.

FIGURA 4-VI : GRÁFICA DEL BRAZO EN MOVIMIENTO.

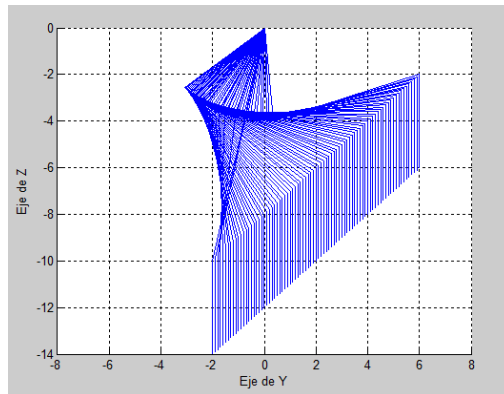
```
for i=1 : 1: top  
  
    [leg1_w leg1_fw leg1_gw] = degres(i, w1, fw1,gw1);  
    plot3(leg1_gw, leg1_w, leg1_fw);  
    xlabel('Eje de X') %eje horizontal  
    ylabel('Eje de Y')%eje vertical  
    zlabel('Eje de Z')%eje de transversal  
    view(55, 35)  
    %axis([-4 8 -8 6]);  
    axis([-15 15 -8 8]);  
    hold on;  
    grid on;  
    pause (0.3);
```

FUENTE : Ilbay Llangarí Luis Guido.

4.2.1 GRÁFICAS DE RESULTADOS.

- Vista del Brazo Robótico en los ejes (Y,Z).

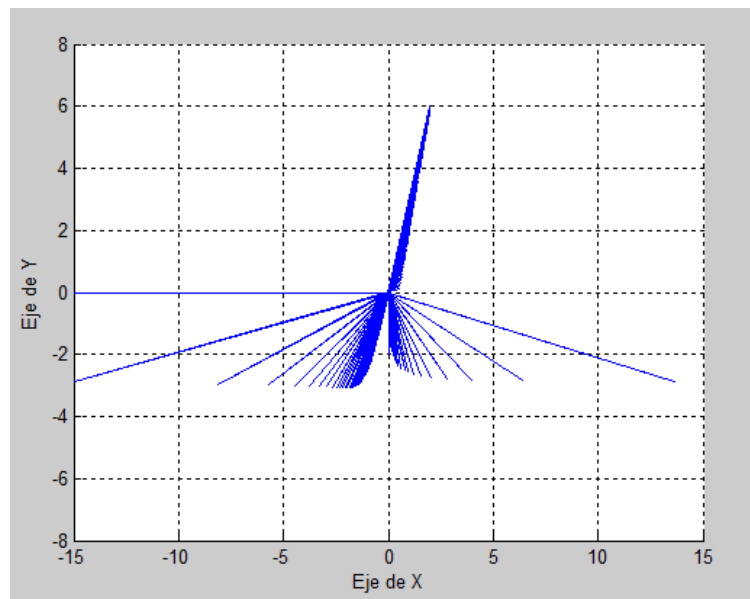
FIGURA 4-VII: MOVIMIENTO EN EL PLANO Y,Z.



FUENTE : Iabay Llangarí Luis Guido.

- Vista del Brazo robótico en los ejes (X,Y).

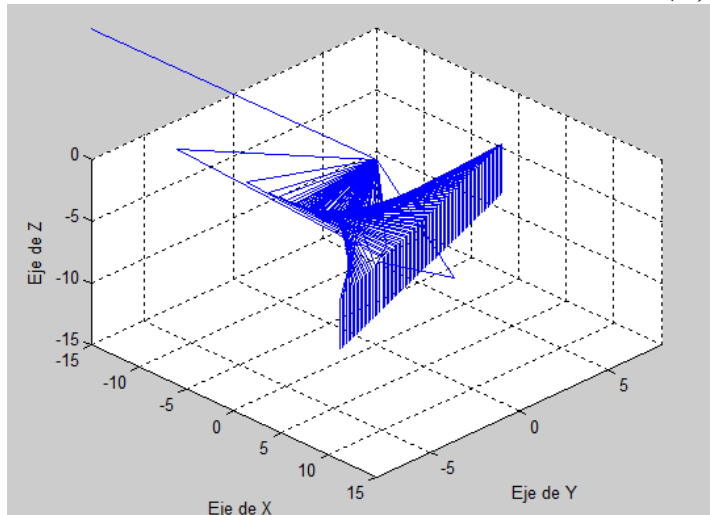
FIGURA 4-VIII : MOVIMIENTO EN EL PLANO Y,X.



FUENTE : Iabay Llangarí Luis Guido.

- Vista Isométrica del Brazo Robótico:

FIGURA 4-IX: VISTA DEL BRAZO EN LOS PLANOS (X,Y,Z)



FUENTE : Ibay Llangarí Luis Guido.

4.3 SIMULACIÓN DEL BRAZO ROBÓTICO EN TIEMPO REAL.

Como paso final en esta investigación se ha realizado una sencilla simulación de un brazo robótico sobre la imagen RGB en tiempo recogida desde el Kinect. Pero la misma tiene que tener parámetros reales y razonables.

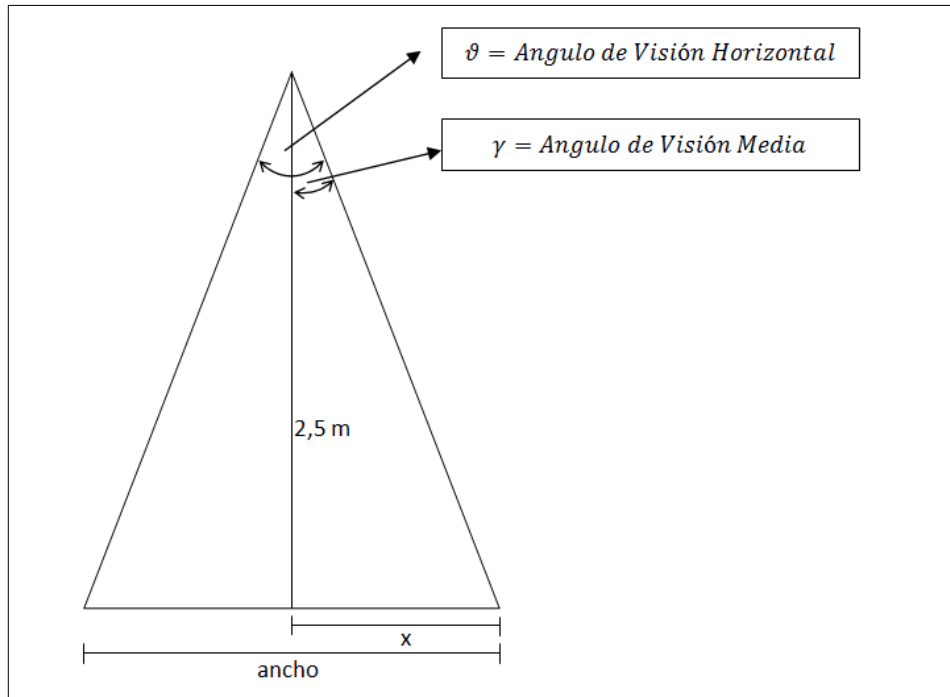
4.3.1 ESPACIO REAL DE VISIÓN DEL KINECT

Debido a que la simulación está representada en la imagen RGB del Kinect, es necesario conocer el espacio real que mira la cámara. En primer lugar se ha tomado un espacio de profundidad de : 2,5 metros, desde la mira de la cámara. Entonces es imprescindible saber los ángulos de visión que tiene el dispositivo para calcular las distancias reales, Los ángulos de visión del Kinect son :

4.3.1.1 HORIZONTALMENTE (58°)

Aplicando trigonometría obtenemos.

FIGURA 4-X : VISIÓN HORIZONTAL DE LA CÁMARA KINECT.



FUENTE : Ilbay Llangarí Luis Guido.

$$\gamma = \frac{\vartheta}{2} = \frac{58}{2} = 29^\circ$$

$$x = \tan(\gamma) * 2,5$$

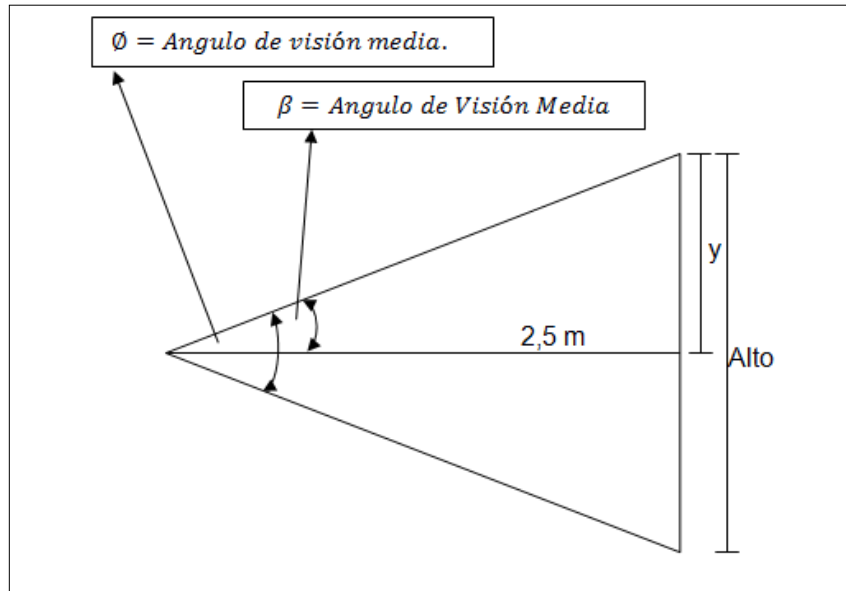
$$x = 1,3858 ;$$

$$ancho = 2 * x = 2,77 m$$

4.3.1.2 VERTICALMENTE (43.5°)

Aplicando trigonometría tenemos:

FIGURA 4-XI : VISIÓN VERTICAL DEL KINECT.



FUENTE : Ilbay Llangari Luis Guido.

$$\beta = \frac{\phi}{2} = \frac{43,5}{2} = 21,75^\circ$$

$$y = \tan(\beta) * 2,5$$

$$y = 0,998$$

$$alto = 2 * y = 1.994 \text{ m} \cong 2 \text{ m}$$

4.3.2 ESLABONES DEL BRAZO ROBÓTICO

Tomando en cuenta que la resolución es de 640 x 480, entendemos como unidad a cada pixel. Por lo tanto aproximamos los eslabones.

Brazo = 500.

Antebrazo = 350.

Muñeca = 100.

Angulo de Cabeceo = 90°

4.3.3 PROGRAMACIÓN.

La simulación de este brazo tiene muchos inconvenientes ya que el espacio físico que podría ocupar está limitado a la resolución de la imagen. Sin embargo, los resultados son bastante apreciables. El punto más importante dentro de la programación es realizar los cálculos trigonométricos dentro del código para lo cual se ha tenido que invocar a librerías de C++ especializadas.

FIGURA 4-XII : LIBRERÍAS NECESARIAS PARA TRIGONOMETRÍA.

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
```

FUENTE : Ilbay Llangarí Luis Guido.

Cabe indicar que los cálculos son hechos en radianes.

Para la simulación del brazo robótico hemos usado el comando de dibujo de líneas. Para la simulación del brazo robótico hemos usado el comando de dibujo de líneas.

FIGURA 4-1 : CÓDIGO DE HALLAZGO DE LOS PUNTOS DE LAS ARTICULACIONES.

```
double AngMunec =(1.5708-AngBrazo-AngAntBr);
int a=AngBrazo*57.2956;
int b= AngAntBr*57.2956;
int c= AngMunec*57.2956;
double PXa = LongBrazo * cos(AngBrazo);
double PYa = LongBrazo * -sin(AngBrazo);
double PXb =LongAntBr * cos(AngAntBr+AngBrazo);
double PYb = LongAntBr * -sin(AngAntBr+AngBrazo);
double PXc =LongMunec * cos (AngMunec+AngAntBr+AngBrazo);
double PYc =- LongMunec * -sin(AngMunec+AngAntBr+AngBrazo);
```

FUENTE : Ibay Llangarí Luis Guido.

4.3.4 RESULTADO FINAL.

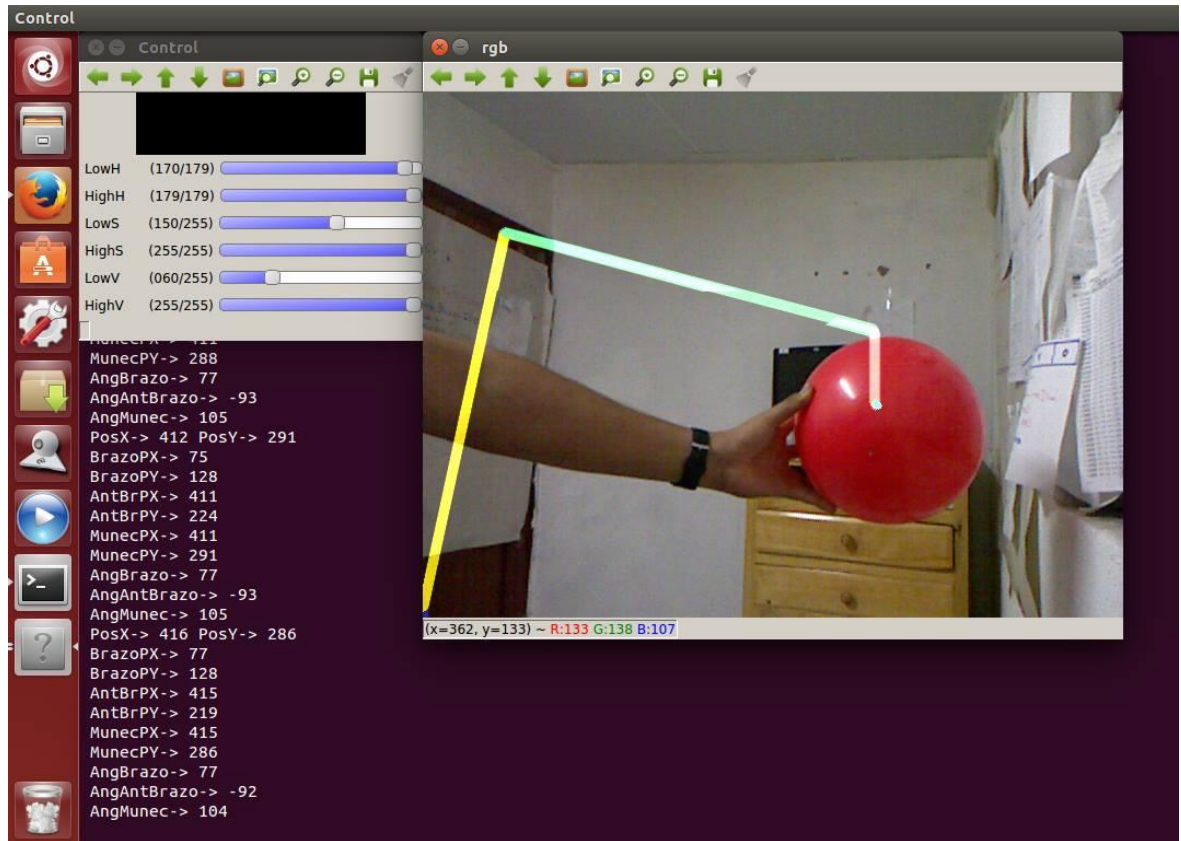
El resultado final es la prueba de funcionalidad de la aplicación. Desde la evaluación de los algoritmos hasta la simulación del brazo robótico en tiempo real. Esta simulación se ha realizado en base al estudio matemático del brazo robótico. La investigación genera un punto de partida en el trabajo con Kinect en la localidad y el país, dado que esta implementado en Open Source.

En la siguiente figura podemos verificar la simulación del brazo robótico. Sigue al objeto rojo y se imprimen los parámetros matemáticos y geométricos tanto del objeto como del brazo. Para la aplicación se han visto necesarias el trabajo conjunto de las herramientas consideradas en el marco teórico. Cumpliendo de manera exitosa el fin común del proyecto.

Se puede observar el resultado de la aplicación en los siguientes URL.

- <https://www.youtube.com/watch?v=e3eEtCv6IZ4>
- <https://www.youtube.com/watch?v=0uYr90hItvA>

FIGURA 4-XIII : IMAGEN FINAL BRAZO ROBÓTICO.



FUENTE : Ibay Llangari Luis Guido.

CONCLUSIONES

- Una evaluación de algoritmos de tracking nos permite escoger al más adecuado para realizar una simulación de brazo robótica usando Kinect.

Bajo la premisa de escoger las herramientas que mejor trabajen con nuestro Hardware se ha podido revalidar el método de prueba y error como uno bastante fiable y el cual nos ha permitido mencionar resultados que a la postre han brindado facilidades y buenos resultados sobre el proyecto.

- Los desarrollos investigativos usando Kinect son escasos en nuestra ciudad, pero lo existentes tienen mucho potencial.

Tras haber realizado un estudio del arte hemos confirmado algo que implícitamente esperábamos encontrar. Y es que la educación superior de nuestra tiene escasos proyectos con el Kinect. Esto a pesar de el costo accesible del dispositivo y de su gran capacidad adaptativa a muchos campos.

- La investigación basada en Open Source tiene muchas libertades, pero es más compleja de llevar a cabo debido a la falta de documentación.

Y esto es aún más difícil cuando la mayor cantidad de trabajo desarrollado está en un idioma extranjero. Sin embargo, viendo la realidad gran cantidad de investigadores usan Open Source, el presente proyecto debía acomodarse a esta corriente de tecnología.

- El algoritmo de Umbralización es el más adecuado para usar con el Kinect en el tracking de objetos de a su sencillez de proceso y a su fiabilidad.

El respaldo a esta conclusión es la implementación y las pruebas presentadas con anterioridad. Sus datos son fiables, entendibles y comparables con las demás

pruebas. Además, no usa gran cantidad de recursos por el compendio de funciones sencillas que se usan en su consecución.

- La visión de las dos cámaras principales del Kinect no son exactamente iguales por lo que se necesita de calibración.

Conclusión que no se podría saber sino después de trabajar arduamente con las visiones que tiene Kinect. Kinect XBOX 360, tiene una visión ligeramente dispareja debido a su constitución física. Las nuevas versiones de Kinect han resuelto este problema.

- El método geométrico es el más adecuado para simular un brazo robótico conociendo los datos finales de posición y la información sobre las dimensiones de los eslabones.

En vista de que existen muchos otros métodos podemos decir que es el más adecuado por su sencillez y por qué tener firmes conocimientos trigonométricos nos adelantan en gran manera la ejecución del mismo.

- Kinect es una herramienta de resultados confiable en la tarea de seguimiento de objetos.

Y es otra de las conclusiones que no tienen respaldo científico en el entorno de educación local, por lo que sus verdaderas prestaciones estaban solamente valoradas basadas en la teoría mas no en la práctica. Ahora se puede decir con seguridad que Kinect puede competir e incluso superar a gran cantidad de cámaras desarrolladas para visión artificial.

RECOMENDACIONES.

- Para futuras investigaciones realizadas con el Kinect se recomienda usar los controladores desarrollados por la comunidad OpenKinect ya que tienen las prestaciones necesarias y están en constante desarrollo.
- Se recomienda la continuación de la investigación realizada implementando un brazo robótico real que pueda alcanzar a un objeto a partir de las demostraciones de este trabajo.
- Se recomienda usar las nuevas generaciones de Kinect ya que estas resuelven en gran manera el desfase de "visión doble" que tiene la usada para esta investigación.
- Antes de continuar con la siguiente etapa de esta investigación se recomienda realizar un estudio de arte actualizado sobre los avances en este campo. Ya que el Kinect es actualmente una gran oportunidad para la visión artificial.
- Se recomienda motivar que el proceso de la investigaciones e implementaciones se hagan en plataforma de Open Source ya que tiene muchas ventajas y se requiere de la masificación general para que en un futuro las próximas investigaciones tengan mayor documentación de estudio y mayor probabilidad de profundidad en sus trabajos.

RESUMEN

La investigación consistió en evaluar algoritmos de Tracking 3D, aplicado a la simulación de un brazo robótico, usando Kinect; en la Escuela de Ingeniería Electrónica de la Escuela Superior Politécnica de Chimborazo.

La evaluación se realizó a tres algoritmos: CAMShift, Algoritmo por Color y Algoritmo por Umbralización, con los cuales se implementó la misma aplicación de seguimiento de objeto (tracking). Escribiendo el programa en C++ y usando los recursos de los Software OpenCV y Libfreenect. Para evaluar los resultados se aplicaron métodos estadísticos como medición y error. Una vez tabulado los resultados, se escogió al Algoritmo por Umbralización como el más adecuado para la aplicación.

El siguiente paso fue realizar un estudio matemático sobre la cinemática de un brazo robótico. Este estudio se realizó en el Software Matlab, mismo que arrojó los parámetros buscados para el uso del robot. Luego, se implementó la simulación del brazo robótico imprimiendo los resultados sobre la imagen capturada en tiempo real.

La parte física del sistema es el Kinect, un sensor compuesto de: cámara RGB, sensor infrarrojo. Su función principal es capturar en tiempo real una escena y obtener las dimensiones de profundidad de todos los componentes que pertenecen a la misma. Toda la programación esta desarrollado en el sistema operativo Linux Ubuntu 14.04. Se usó el lenguaje de programación C++, trabajando en conjunto con las librerías OpenCV y Libfreenect, que son librería para el uso del Kinect y la V.A..

Se corroboró que el sistema tiene un 98,83 de precisión en los datos.

Concluimos que el algoritmo de Umbralización es el más adecuado para una aplicación de Tracking usando Kinect.

Se recomienda a los investigadores de la Escuela de Ingeniería Electrónica, tomar los resultados de este trabajo como punto de partida para implementaciones físicas de brazos robóticos que se realicen posteriormente.

Palabras Clave: <SOFTWARE MATLAB> <DISPOSITIVO KINECT>
<TRACKING> <ALGORITMO> <SOFTWARE OPENCV> <SOFTWARE LIBFREENECT> <SISTEMA OPERATIVO LINUX> <UBUNTU> <ALGORITMO CAMSHIFT> <SOFTWARE> <TRIDIMENSIONAL>

ABSTRACT.

The research consisted of evaluating Tracking algorithms 3D, applied to the simulation of a robotic arm, using Kinect; in the Electronic Engineering School of Escuela Superior Politécnica de Chimborazo.

The evaluation was realized to three algorithms, CAMShift, Algorithm by Color and thresholding algorithm, with them the application of the object monitoring (tracking) was implemented. Writing the program in C++ and using the resources of the Software OpenCV and Libfreenect. The results were evaluated applying statistical methods as measurement and error. When the results were tabulated the Algorithm by thresholding was chosen as the most adapted for the application.

A mathematical study on the cinematic of a robotic arm was the following step. This study was realized in the Software Matlab, the same that gave the parameters researched for the robot use. Then, the robotic arm simulation was implemented printing the results on the image captured in real time.

The physical system par is the Kinect, a sensor composed of camera RGB, infrared sensor. Its main function is to capture in real time a scene and obtaining the depth dimensions of all the components that belonging it. All programming is developed in the Linux Ubuntu 14.04 operating system. The programming language C++ was used, working as a whole with the bookstores OpenCV and Libfreenect, which serve for the use of Kinect and artificial vision.

It was corroborated that the system has the 98,83%, of precision in the information.

It is concluded that the thresholding algorithm is the most adapted for a Tracking application using Kinect.

It is recommended to the researches from Electronics Engineering School, to take the results of this work as a starting point for physical implementations of robotics arms that will perform subsequently.

Key Words : SOFTWARE MATLAB, DEVICE KINECT, TRACKING, ALGORITHM, SOFTWARE OPENCV, SOFTWARE LIBFRENECT, OPERATING SYSTEM LINUX, UBUNTU, ALGORITHM CAMSHIFT, SOFWARE, THREE DIMENSIONAL.

BIBLIOGRAFÍA

- 1. GRUPO DE INVESTIGACIÓN EDMANS** Técnicas y algoritmos básicos de visión artificial., Universidad de La Rioja., Servicio de Publicaciones., 2006. pp. 1-91.

- 2. ARBONA A.,** Análisis de Imágenes de profundidad para aplicaciones de realidad aumentada mediante el uso de la Kinect., Valencia - España., Universitat Politècnica de València., 2013., pp. 1-17

- 3. ALBA A.,** Ejercicios con OpenCV., Facultad de Ciencias, UASLP., Septiembre 2011., pp. 1-5.

- 4. COMPUTER VISIÓN AND ROBOTIC GROUP.,** Operaciones Morfológicas, Universitat de Girona, pp. 41- 59.

- 5. DIAZ CELIS C. Y ROMERO MOLANO C.,** Navegación de Robot móvil usando Kinect, OpenCV y Arduino., Meta - Colombia., Universidad de Llanos., 2012., pp. 1-8

- 4. GRUPO DE INTELIGENCIA ARTIFICIAL Y ROBÓTICA.,** Algoritmo de seguimiento de objetos basado en visión asistida por computador en tiempo real utilizando CAMShift e histogramas ponderados., Buenos Aires - Argentina., Universidad Tecnológica Fac. Regional Bs As. pp. 1-6

- 5. ESPAÑA,** Escuela Universitaria de Ingeniería Técnica Industrial de Madrid., Técnicas de Segmentación de Imágenes., pp. 5.6 - 5.9.

6. PÉREZ T., Control y Programación de un robot industrial usando Microsoft Kinect., Proyecto de fin de carrera., 2013., pp. 1-12

7. RAMIREZ A., Reporte de Búsqueda, Detección y Conteo de Objetos., Guanajuato - México, Centro de Investigación en Matemáticas A.C ., pp. 1-23

8. ILVAY R. Sistema de Educación para niño de 3-5 años, mediante un robot controlado por el sensor Kinect., Tesis Ing Electrónica., Riobamba-Ecuador., Escuela Superior Politécnica de Chimborazo., 2014., pp. 17-36

9. HERNANDEZ L. y HERRERA J., Análisis y estudio de los códigos **FUENTE** SDK (Kit de Desarrollo de Software) e implementación de una aplicación demostrativa que registre la captación de movimientos de manos y brazos del cuerpo humano a través de led's indicadores mediante la utilización del sensor Kinect para XBOX 360., Tesis Ing. Electrónica., 2013., pp 4 - 8.

10. PRIETO A. Desarrollo de un prototipo de un api para la interacción de un usuario con aplicaciones con contenido 3d utilizando Kinect., Tesis Ing Ciencias Computacionales., Guayaquil - Ecuador ., 2013., pp. xii -12.

11. ÑACATO D., Diseño e implementación de un sistema de tele operación para controlar un robot humanoide mediante un sensor kinect., Tesis Ing. Electrónica., Riobamba - Ecuador ., 2014., pp. 16 - 20

12. KINECT .

< <https://msexpertos.wordpress.com/2012/10/page/2/>>

[12 de Enero 2015]

< <https://github.com/OpenKinect/libfreenect>>

[13 de Enero 2015]

< <http://blogthinkbig.com/kinect-mas-alla-del-ocio/>>

[20 de Enero 2015].

13. OPENCV.

<<http://www.samontab.com/web/2014/06/installing-opencv-2-4-9-in-ubuntu-14-04-lts/>>

[20 de Enero del 2015].2015-01-20

< <http://acodigo.blogspot.com/p/tutorial-opencv.html>>

[25 de Enero del 2015].

< <http://opencv.org/>>

[17 de Enero del 2015>

14. ROBOT

< http://estefaniaospina774p.blogspot.com/2012_10_01_archive.html>

[15 de Marzo del 2015]

< <https://sites.google.com/site/proyectosroboticos/cinematica-inversa-i>

[10 de Marzo del 2015]

<http://www-pagines.fib.upc.es/~rob/protegit/treballs/Q2_03-04/general/carmorf.htm >

[30 de Marzo del 2015]

http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/sistema/transmisiones.htm

[12 de Abril 2015]