UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

VISIÓN ESTEREOSCÓPICA EN UN ROBOT HUMANOIDE PARA LA DETECCIÓN DE OBJETOS SOBRE UN PLANO

TESIS

Que para obtener el título de

Ingeniero Eléctrico - Electrónico

P R E S E N T A

Héctor Rodrigo Arce González

DIRECTOR DE TESIS

Dr. Jesús Savage Carmona

Ciudad Universitaria, Cd. Mx., 2016

Jurado Asignado

PRESIDENTE:	DR. CARLOS RIVERA RIVERA
VOCAL:	DR. JESUS SAVAGE CARMONA
SECRETARIO:	ING. ROMAN VICTORIANO OSORIO COMPARAN
1er SUPLENTE:	M.I. RUBEN ANAYA GARCIA
2do SUPLENTE:	MTRO. MARCO ANTONIO NEGRETE VILLANUEVA

Agradecimientos

A mi familia por el apoyo que me han brindado a lo largo de los años.

Universidad Nacional Autónoma de México

Facultad de Ingeniería

y Laboratorio de Bio-robótica

Por la formación brindada y permitirme realizar mis estudios hasta este punto

Al PAPIME PE102115 “Prácticas para la materia de Diseño de Casas Inteligentes” por el apoyo monetario brindado

Índice general

Índice de figuras

Índice de cuadros

CAPÍTULO 1

Introducción

1.1. Justificación

Actualmente la tecnología avanza cada vez mas rápido y se ha convertido en una parte importante para la vida de las personas, derivando en una mayor necesidad de trabajos especializados para sustentar el avance de la sociedad. Es debido a esta necesidad que la robótica se ha vuelto una disciplina tan socorrida, pues estos mecanismos automatizados se pueden aplicar directamente a ciertas labores humanas, que pueden ser consideradas como triviales desde el punto de vista que no resultan productivas para el desarrollo de las personas o la sociedad. En esta tesis se aborda la aplicación sobre un tipo especifico de estructura robótica, la del robot humanoide. Este tipo de robot se caracteriza por tener un aspecto físico similar al humano, del cual se hablara mas adelante, y la investigación sobre este tipo de mecanismo se justifica a partir del fin funcional de interactuar con el entorno social existente sin ningún problema y poder así realizar varias de las tareas humanas que otro tipo de robot no podría.

La implementación que se hará sobre el robot humanoide, es la de visión estereoscópica, la cual es una técnica de visión computacional basada en la visión frontal humana, la cual plantea que al tener 2 imágenes, se puede obtener una representación tridimensional del ambiente percibido. Es importante la percepción en un robot ya que este debe moverse e interactuar con el entorno, la razón por la cual se busca implementar un sistema de visión estereoscópica por sobre algún otro, es que al ser un método para obtener información del medio, que parece ser similar al utilizado por los seres humanos, además de ventajas computacionales y electrónicas mencionadas mas adelante, le da al robot la capacidad de interactuar de forma mas natural con el medio ambiente.

1.2. Objetivo

Con este trabajo se busca entender e implementar un sistema de visión estereoscópica, en especifico al robot humanoide "Nimbro-OP", de tal forma que se simplifique la tarea de localizar objetos a partir de su distancia. Se utilizara como referencia para la implementación de la tesis las reglas de la competencia "RoboCup", mas en especifico, la categoría de soccer para robots humanoides.En este caso de estudio se sabe que la visión estereoscópica nos deberá ayudar para reconocer objetos localizados sobre un plano definido como por ejemplo; la pelota, porterías, y posibles obstáculos como los robots oponentes, así como su distancia al centro de visión.

Los sub-objetivos que se abarcan en el ya mencionado son los siguientes:

- Investigar y elegir una cámara estereoscópica debido a que es el requerimiento obvio para la obtención de los datos a procesar.
- Comparar y seleccionar una computadora que sea viable para la implementación del trabajo realizado.
- Implementar un algoritmo para detección de objetos a partir de una representación basada en distancias, como una "nube de puntos"o "mapa de distancias".
- Integrar en el robot a nivel computacional el sistema de procesamiento de visión y el sistema motriz ya existente.
- Probar experimentalmente el reconocimiento de objetos con el sistema de procesamiento aplicado al robot humanoide.

Cabe destacar que considerare como objeto a una representación tridimensional con distancia conocida y cierta continuidad, esta definición abarca solo la detección de los objetos, mas no su reconocimiento, tal y como se indica también en el titulo de la tesis.

1.3. Prologo

Este trabajo escrito se sustenta directamente como una aplicación al robot humanoide "Nimbro-OP" perteneciente en parte al laboratorio de Bio-Robótica de la Universidad Nacional Autónoma De México, adquirido en colaboración con la Universidad La Salle y el Instituto Tecnológico de Monterrey campus Estado de México. Sin embargo, el trabajo realizado no esta restringido solamente a esta aplicación debido a su versatilidad y posible utilización en otros sistemas automatizados. Debido a esto es que la estructura de la tesis desarrolla primero las características del robot humanoide y posteriormente las del sistema

de visión. Los capítulos siguientes (posteriores a esta Introducción), se desarrollan de la siguiente forma:

En el Capítulo 2 se abordan los antecedentes teóricos así como las definiciones necesarias para la realización de este trabajo, como por ejemplo los antecedentes de visión computacional y sistemas robóticos.

El Capítulo 3 describe al robot humanoide y el como puede implementarse el sistema de visión en el, así como los requisitos para realizarlo, ya sea de forma computacional como es el caso del sistema operativo y los programas necesarios, y también se presenta el caso de los requisitos de hardware como la computadora

El Capítulo 4 se presentan la caracterización de la cámara estereoscópica elegida, las ecuaciones y modelos utilizados para caracterizar la obtención de los datos, así como su implementación en el sistema robótico.

Los resultados del sistema de visión aplicado experimentalmente al robot se presentan en el Capítulo 5

Para el Capítulo 6 se presentan las conclusiones finales resultantes de la investigación, así como algunos de los trabajos posibles planteados a futuro.

Finalmente, se presentan los apéndices mencionados a lo largo de la tesis, que son referentes a las adaptaciones para que el sistema final funcione correctamente. Cabe mencionar que en estos apéndices solo las configuraciones de los archivos finales tienen titulo, omitiéndolo en las secciones que solo son pasos a seguir, además de la notación de color que he elegido, las instrucciones en consola se encuentran en fondo negro, mientras que las modificaciones a archivos en blanco..

CAPÍTULO 2

Marco teórico

2.1. Descripción del Robot

2.1.1. Autonomía

Actualmente se considera que un robot es una entidad artificial o electro-mecánica, controlada generalmente por una sistema computacional. Desde un punto de vista de su comportamiento los robots pueden ser de 2 tipos principiante:

- Autónomo
- Semi-autónomo

El robot semi-autónomo tiene algunas funciones pre-programadas en algún controlador, pero además tiene una gran dependencia de la entrada o control efectuado por un usuario externo, y es a partir de esta intervención que realiza sus funciones. A diferencia de esto, un robot autónomo es aquel que puede realizar tareas con un alto nivel de independencia hacia un control manual externo y operar indefinidamente de esta manera, esto es útil para diversas áreas como la industria, exploración de ambientes extraños, mantenimiento del hogar, vigilancia, entre varios otros. Una de las ventajas mas evidentes de un sistema autónomo sobre uno semi-autónomo, es la capacidad del primero para compensar algún posible error ocasionado por situaciones inesperadas.

Un ejemplo común actualmente de este tipo de robots autónomos son una variante de los brazos robóticos que se puede encontrar en la industria, como los de la linea de ensamblaje de una fabrica automotriz, los cuales se dedican a colocar las partes de un chasis en su lugar correspondiente, estos robots tienen pre-programadas la ubicación donde las piezas deben ser

colocadas, pero además deben tener la capacidad de estimar de forma autónoma la posición real del chasis en construcción, es esta ultima característica la que hace la diferencia entre un robot industrial autónomo y uno semi-autónomo, ya que el diseño del comportamiento de un robot semi-autónomo se basa en la confianza de todos los componentes se encuentran en el lugar que deben y esto desprecia cualquier efecto caótico imprevisto.

Es debido a esta capacidad de interactuar con un ambiente dinámico que los robots autónomos son un tema de investigación tan importante y recurrente en la actualidad, cabe destacar que el interés en estos sistemas no es del todo nuevo, sino que ha sido un largo camino el recorrido que ha tenido esta área, desde el primer uso registrado de la palabra "Robot" como referente a una entidad artificial en 1921, por el escritor Checo Kael Čapek. Sin embargo, hay que denotar que actualmente los robots totalmente autónomos son inexistentes, ya que aun no son capaces de realizar la función para la que fueron diseñados de forma indefinida, requisito que como mencionaba previamente resulta indispensable para este tipo de robots, y debido a la aun existente limitación es que nacen proyectos de investigación y colaboración como el "RoboCup".

Actualmente los robots autónomos no tienen ninguna limitación de diseño para caer dentro de esta categoría, sino que al contrario, poseen muchos requisitos, además del sistema de computo en el que las ya mencionadas posibles funciones del robot se almacenan, también se necesita un hardware para que el robot interactúe con el entorno físico que lo rodea, este sistema debe tener la capacidad de adquirir datos del entorno (sistema de sensado), y similarmente poseer la capacidad de actuar físicamente (sistema actuador), agregando estas 2 características adicionales al ya mencionado requisito de actuar autónomamente, tenemos 3 clasificaciones de características necesarias las cuales a continuación describo brevemente:

- Sistema actuador: este sistema se compone por todo aquel elemento que genere una cambio físico en el entorno como por ejemplo un motor, servomotor, etc.
- Sistema de sensado: este sistema se compone de varios elementos que registraran el valor de diversas variables físicas del entorno y las convertirán a una referencia eléctrica, la cual podrá ser procesada digitalmente para cualquier interacción con las funciones programadas del robot. Un ejemplo de las variables que mas comúnmente son leídas por distintos sensores son: Luz, temperatura, campo magnético gravitacional, aceleración, giro respecto a una referencia, imagen, etc.

  A su vez este sistema se divide en 2 tipos de percepción, la primera mide las variables dependientes al estado interno del robot (propiocepción), y la segunda depende de variables dependientes del medio ambiente externo al robot (exterocepción).
- Sistema de planeación o planificación, es aquel que realiza la relación entre el sistema de acción y de sensado, este sistema se dedica a ejecutar cierta acción en cierto tiempo óptimo previamente definido o programado, y esto lo realiza a partir de reglas y la información obtenida del sistema de sensado.

Ahora cabe denotar que las formas en las que se relacionan los clasificaciones anteriores se les conoce como "paradigmas de la robótica", además, dentro de este contexto a las mismas clasificaciones se les denominan "primitivas", pues se considera que son la base mas simple para efectuar un sistema de comportamiento en un robot, cualquiera que sea el tipo (móvil, autónomo, semi-autónomo, etc). Estos paradigmas actualmente se listan dentro de 3 categorías, sin embargo algunos autores consideran una cuarta, a continuación una breve descripción de cada uno de ellas:

1. Paradigma Jerárquico o control basado en planeación:

   A este paradigma se le conoce como el modelo mas tradicional de inteligencia, prevaleciendo entre 1967 y 1990 (Murphy, 2000).
   Este modelo se basa en un "modelo introspectivo" de pensamiento, es decir, el procesamiento se desarrolla al rededor de una representación interna del "mundo" en el que el robot actúa, esta representación puede ser creada a partir de un modelo físico del "mundo", así como una opinión desde el punto de vista de la maquina: zonas de peligro, zonas seguras, etc. Debido al modelo realizado del mundo, el robot tiene la capacidad de resolver una gran cantidad de tareas, pues su restricción de datos disponibles es menor, sin embargo, esto también acarrea la mayor desventaja de este paradigma, que es el costo computacional de realizar y analizar el modelo del "mundo" debido a su gran cantidad de datos a considerar, sin mencionar que debido a que el robot basa su conocimiento en este modelo y no el "mundo real" (o externo), hay cierto grado de error posible al realizar acciones. Considerando el desarrollo de este modelo se puede notar que el razonamiento se lleva a cabo de manera lineal entre las primitivas, quedando de la siguiente forma:

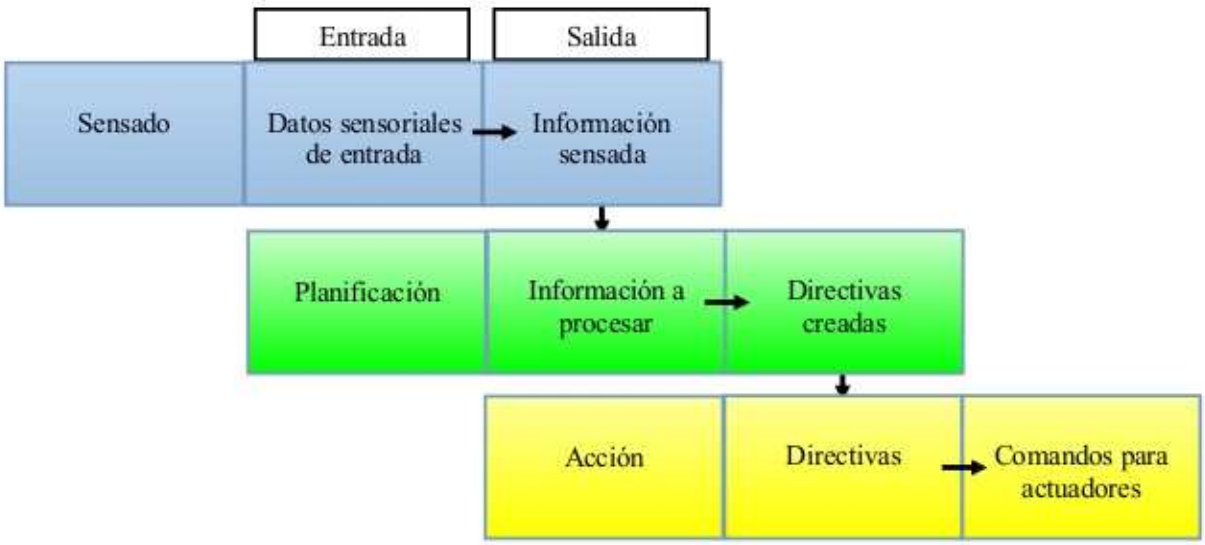


Figura 2.1: Esquema del paradigma jerárquico

Como se puede ver, el primer paso en este paradigma es la obtención de datos del

mundo externo, a través de un sistema sensorial del que se hablo previamente, estos datos posteriormente se procesan y acumulan en un modelo global del "mundo" para planear la acción siguiente, una vez planeada es efectuada por el sistema de acción. Estos 3 pasos se realizan hasta que el robot alcance su objetivo programado.

2. Paradigma Reactivo :

   Este paradigma originalmente se fundamento principalmente en la creación de un modelo de comportamiento mas "animal", o en cierta forma impulsivo, por lo que este es un diseño basado en la etología, con el cual se afirma que el estudio del comportamiento animal aplicado a la robótica puede dar respuestas valorables relacionadas al comportamiento humano. (Arkin, 1998)

   Desde este punto de vista etiológico usualmente el comportamiento animal es aparentemente estructurado, por ejemplo el de una presa al ser perseguida, de forma muy simple podría señalarse que esta huye, se oculta, detecta al depredador y así sucesivamente hasta detenerse si la persecución ha terminado. Es a partir de esta premisa de estructuración que se puede resumir el esquema de comportamiento del paradigma reactivo de la siguiente forma:
   - Comportamiento seccionado en reacciones
   - Activadores seleccionados para las reacciones
   - Secuencias de acciones y sensado
   - Secuencias implementadas de forma iterativa

Figura 2.2: Esquema del paradigma reactivo

3. Paradigma Híbrido/Deliberativo:

   El paradigma híbrido se basa principalmente en el paradigma reactivo, incorporando el elemento de planeación del paradigma jerárquico, recordando que para la planeación se requiere una interpretación del "mundo" y poder así interactuar de esta forma con el. La ventaja principal de combinar ambos modelos, es que se puede reaccionar de forma inmediata a perturbaciones debidas al "mundo real", y a su vez, generar planeación para resolver tareas complejas que de forma reactiva no se lograrían completar. De

forma funcional, los datos sensados por el robot están disponibles tanto para el sistema de acción como para el planeador, pudiendo ambos reaccionar de cierta manera y con alguna prioridad basada en la tarea actual.

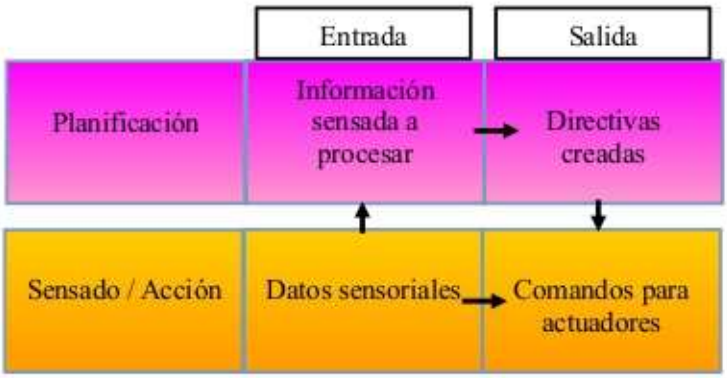


Figura 2.3: Esquema del paradigma híbrido

4. Control basado en comportamientos:

   Estos sistemas son muy similares a los del paradigma reactivo, sin embargo el control basado en comportamientos incluye la característica de poder guardar representaciones (similarmente al paradigma jerárquico). Este paradigma une los campos de inteligencia artificial y ciencias cognitivas para obtener una arquitectura de agentes inteligentes, y su fin es desarrollar métodos para controlar sistemas artificiales y desarrollar robots para el modelado y mejor entendimiento de sistemas biológicos.

2.1.2. Movilidad

Además de la clasificación de un robot por su grado de autonomía, también puede ser clasificado por su capacidad de trasladarse de un lado a otro, un robot que es capaz de locomoción se clasifica como "móvil". Un ejemplo de un robot móvil sencillo es uno que mecánicamente se describiría como un vehículo con ruedas, mientras que el ejemplo del tipo estacionario (no móvil), puede ser el brazo robótico como el mostrado en la imagen 2.4 .

Figura 2.4: Brazo robótico estacionario.

Figura 2.5: Robot móvil: vehículo con ruedas.

En conjunto con la clasificación de autonomía, tenemos las siguientes subdivisiones:

- Robot móvil y semi-autónomo, o Autónomo guiado:
  Estos dependen de una guía externa para cumplir una ruta predeterminada y trasladarse en un espacio relativamente controlado.
- Robot móvil autónomo:
  Son aquellos que navegan en un espacio no controlado sin necesidad de una guía.

Un robot clasificado como humanoide es un robot móvil, que tiene la disposición física de sus actuadores en una forma similar a la del cuerpo humano, en pocas palabras un robot humanoide es aquel que tiene las siguientes partes, o al menos la mayoría de ellas: un torso, 2 piernas, 2 brazos y cabeza. El primer intento documentado de realizar un robot humanoide data del año 1495 por Leonardo Da Vinci.
En el caso del robot utilizado para esta tesis, es un robot humanoide diseñado con todas las extremidades anteriores, conocido como "Nimbro-OP".

Originalmente el interés en desarrollar robots con apariencia humanoide surgió a partir de la necesidad de construir mejores prótesis y órtesis para el ser humano, ya que al tener un organismo que imite físicamente las condiciones humanas, se tiene la facilidad de realizar experimentación y desarrollos sin incurrir en métodos directos al cuerpo humano con la certeza de que el resultado final sera compatible con las condiciones humanas, y al mismo tiempo obtener un mayor entendimiento del cuerpo humano al simularlo. Actualmente los robots humanoides son utilizados principalmente como herramienta de investigación, en el caso de la ingeniería algunas de las áreas de interés son control, inteligencia artificial, entre otras. Pero se considera que de forma practica las aplicaciones de los robots humanoides, debido

a la arquitectura de estos, abarcaran una buena cantidad de tareas físicas que actualmente realizan los seres humanos.

Figura 2.6: Hasta el frente, HRP-4 y sus versiones anteriores.

Fuente: global.kawada.jp/mechatronics/hrp4.html

Figura 2.7: Asimo, el robot humanoide mas conocido.

Fuente: asimo.honda.com

Figura 2.8: Justin, diseñado para incursiones espaciales.

Fuente: http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-4789/7945_read-12712/

Para que el robot humanoide mantenga una arquitectura física similar a los humanos se utilizan actuadores que puedan emular los rangos de movimiento efectuados por los miembros humanos, aunque no con la misma composición, un ejemplo de esto es un brazo, que mientras en el cuerpo humano su movimiento lo realiza a través de músculos compuestos por tejido, en el caso de un robot humanoide un brazo puede ser accionado por actuadores rotatorios (como motores) o por un arreglo de actuadores lineales (pistones). Cabe mencionar que los métodos para accionar estos actuadores pueden ser con varios tipos de efectores físicos, como los eléctricos, hidráulicos, neumáticos, entre otros.

Debido a las características físicas de los robots humanoides, es una tarea crucial el considerar la planeación y el control de los movimientos de las extremidades y su interacción con el "mundo real", pues a diferencia de un robot con ruedas, la mala planeación de un movimiento puede derivar en que el robot caiga o incluso se genere un daño permanente en su estructura mecánica. Por lo que para generar este control se necesita una cantidad considerable de datos de la interacción del robot con el medio externo y su propio cuerpo a través de un sistema de sensado.

2.1.3. Sensado

Como se mencionaba previamente, el robot humanoide requiere de una gran cantidad de datos para su correcto funcionamiento, debido a esto es que se requieren una buena variedad de sensores diferentes que cumplan con el requisito de abarcar cada una de las

variables físicas de interés, que como ya se menciono previamente, se dividen en 2 tipos: las otorgadas por del medio ambiente (exterocepción), y las pertenecientes al estado del robot mismo (propiocepción), basado en estas 2 ultimas divisiones y el conocimiento ya descrito de la arquitectura física del robot, se pueden dar las siguientes divisiones y definiciones de la primitiva sensorial, específicamente de un robot humanoide:

- Propiocepción. A partir del diseño mismo del robot y su necesidad de calcular la trayectoria de sus propios miembros en el espacio, ya que deben co-actuar en armonía:

  - Dada la exigencia de saber la posición actual de cada actuador respecto al miembro que compone y el robot, se utilizan usualmente se utilizan sensores denominados "encoders", los cuales entregan una medida numérica de la posición del actuador respecto a una referencia, los hay tanto rotatorios como lineales. Estos mismos sensores pueden otorgar la velocidad individual de cada actuador.

  - Debido a que el robot necesita conocer su balance, orientación y velocidad, se utilizan sensores de aceleración (acelerómetros) e inclinación respecto a cierta referencia ambiental(giroscopio), además también pueden utilizarse sensores de orientación magnética, los cuales aunque miden el campo magnético de la tierra como referencia externa, son utilizados para auxiliar la orientación del robot.

- Exterocepción. Dado que el robot necesita conocer su ubicación en el espacio, debido a que es un robot móvil, y la ubicación de objetos externos ya que debe cumplir ciertas tareas tenemos:

  - Para satisfacer la necesidad de localizar fuentes de sonido, así como de recibir instrucciones sonoras de diferentes tipos, se emplean usualmente micrófonos para imitar el sentido del oído.

  - En el caso de requerir detectar el calor del ambiente se pueden utilizar sensores de temperatura, los cuales fácilmente pueden entregar los datos deseados.

  - La percepción de luminosidad en el ambiente se puede realizar a través de diferentes tipos de sensores diseñados para este fin, como foto-resistencias, o sensores que utilizan elementos semi-conductores.

  - Para la identificación de objetos y distancias hay varios tipos de sistemas de sensado que pueden utilizarse, por ejemplo, se pueden utilizar sensores que entregan directamente distancias como sensores ultrasónicos o infrarrojos, sin embargo al emplearlos individualmente solo puede obtenerse información geométrica del entorno, pues naturalmente entregan datos de distancia.
  Otro tipo de sistema mas sofisticado para identificar objetos es por medio de sensores de imagen, los cuales usualmente se encuentran en cámaras, estos sensores detectan emisiones de radiación electromagnética, por lo que pueden ser de imagen a color, imagen térmica, sonar, etc., la ventaja de estos sensores sobre los sensores de distancia es que obtienen muchas características de los objetos detectados, sin embargo, la distancia no se obtiene tan directamente, pues se requiere una mayor cantidad de cálculos

computacionales.
Además de los 2 sistemas mencionados previamente, actualmente en el mercado existen ya sensores que fusionan el hardware de ambos, así como sus ventajas, para aligerar el peso computacional en su implementación.

2.2. Visión computacional

En este apartado se abordan conocimientos base para comprender la aplicación y teoría de la visión estereoscópica, partiendo de los conceptos de luz y visión humana, para seguir a la representación digital de una imagen y la representación tridimensional a partir de distancias, así como la forma de obtenerla.

2.2.1. Conceptos físicos de luz y visión humana

Concepto de Luz

La luz es energía radiante, una onda electromagnética como tal, la cual se diferencia de las ondas electromagnéticas por el hecho de que el ojo es sensible a ella, y es con esta que los órganos de visión humana pueden realizar la función de percibir el entorno. De forma teórica se tiene ya bastante conocimiento de la luz, sin embargo, los conocimientos biológicos de los órganos que interactúan con ella son mayormente derivados de la experimentación, pero hasta ahora han resultado útiles para realizar modelos funcionales. Como tal esta energía radiante se mide por longitud de onda λ, la cual emite en un intervalo (ancho de banda) entre aproximadamente los 350 y 780 nanometros. Una forma de cuantificar la luz es a partir de la cantidad emitida con cierta longitud de onda, osea por su capacidad como fuente de luz, de acuerdo con Lim (1990) las fuentes de luz pueden clasificarse en dos:

1. Fuente primaria de luz: Aquella que emite su propia luz, como por ejemplo el sol, una lampara, o el fuego.
2. Fuente secundaria de luz: Es la que refleja la luz emitida por otra fuente, este es el caso de la mayoría de los objetos existentes.

Una onda electromagnética transporta energía, y esta energía al atravesar por un plano espacial puede ser representada con la función de irradiancia por longitud de onda:

$$c(x, y, t, \lambda)$$

Donde x y y son las variables espaciales en 2 dimensiones, t es el tiempo y λ es la ya mencionada longitud de onda. La longitud de onda y la frecuencia se relacionan entonces de la siguiente manera:

$$\lambda = \frac{c}{f}$$

Donde c es la velocidad de una onda electromagnética, aproximadamente $3 * 10^8$ m/sec en el vacío.

Percepción de la luz

La percepción humana de la luz medida por $c(\lambda)$ se puede descomponer en tres componentes principales para describirla:

1. Brillo:
   Se refiere a la intensidad luminosa percibida sobre el cierto objeto. Cabe destacar que dicha percepción es relativa al observador o punto de observación, ya que el brillo cambia dependiendo la reflexión del material del área vista y la posición de la fuente de luz respecto al objeto.

2. Tono:
   Es la propiedad para distinguir entre lo que se conoce como color, de forma física el color se clasifica a partir de la longitud de onda. La luz se puede descomponer en estos colores a partir de un prisma, tal y como muestra la siguiente imagen:

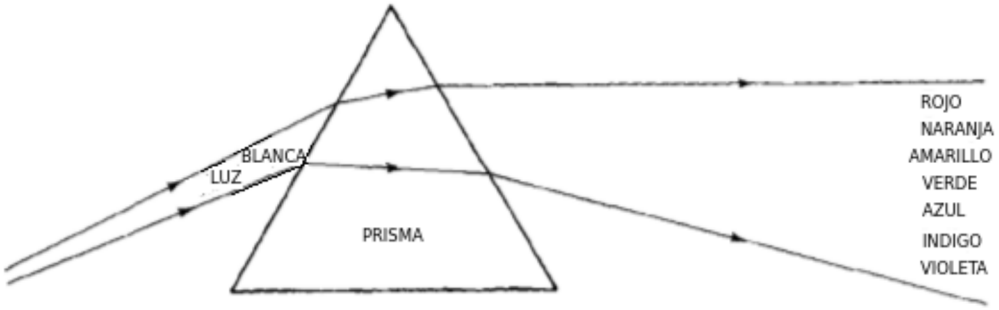


Figura 2.9: Descomposición de la luz en colores

Fuente: Pratt (2001)

Esto fue realizado por primera vez por Issac Newton en 1666, él dividió la luz visible en siete categorías: Rojo, Naranja, Amarillo, Verde, Azul, Índigo y Violeta, respectivamente ordenados de de mayor a menor longitud de onda. Debido a esta división puede

ser posible relacionar algunos colores con su longitud de onda, considerando que ciertos colores se componen de mezclar los colores descritos por su λ. A continuación muestro una imagen de la posición de cada color en longitud de onda dentro del espectro visible, el cual a su vez se encuentra dentro del espectro electromagnético.

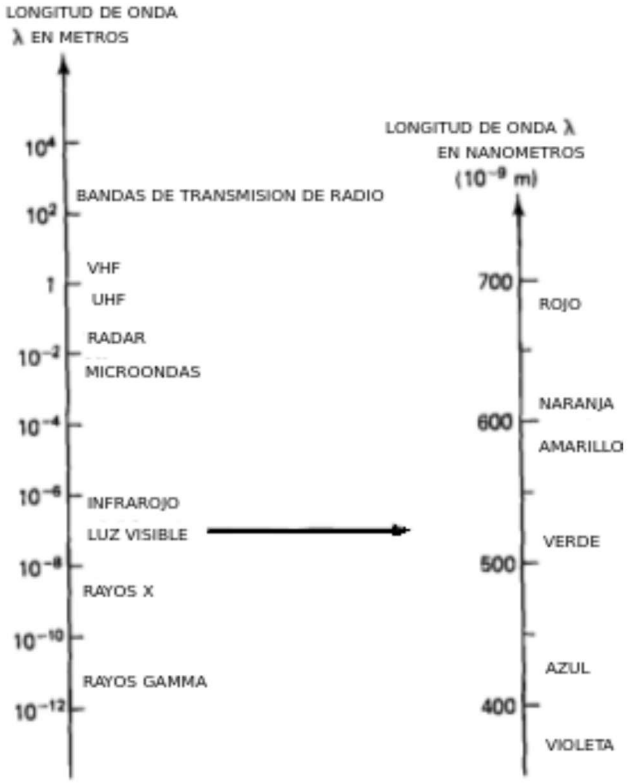


Figura 2.10: Ubicación del espectro visible dentro del espectro electromagnético

Fuente: Lim (1990)

3. Saturación: Se refiere a la "blancura" de una fuente de luz. Esta característica ayuda a definir que tan pálido u opaco se percibe un objeto a partir de su color. Entre menos vivido se percibe el color, se dice que es menos saturado.

Sistemas de color

Recordando la función de irradiancia por λ, y que el color se clasifica cada cierta longitud de onda, podemos considerar dos luces de cierto color $C_1(\lambda)$ y $C_2(\lambda)$, para operarlas de la siguiente forma:

$$C_1(\lambda) + C_2(\lambda) = C(\lambda)$$

Siendo $C(\lambda)$ la luz resultante.
A la operación anterior se le conoce como sistema aditivo de color. Como ya había mencionado brevemente, se tiende a describir algunos colores por combinación de otros existentes, por lo que igualmente al mezclar dos fuentes de luz con diferentes longitudes de onda, se crea una fuente de luz de color distinto a las mezcladas. Para generar otros colores se ha adquirido una convención en la que se definen tres colores primarios a partir de los cuales se pueden obtener las respectivas combinaciones de los otros colores.

En el caso del apenas comentado sistema aditivo, los colores primarios son el Rojo (R), Verde(G) y Azul (B), con longitudes de onda establecidas en 700nm, 546.1nm y 435.8nm respectivamente. Al mezclar de forma uniforme e individual un color primario con otro, se obtiene otro conocido como color secundario del sistema, en el caso de este sistema aditivo los colores secundarios son el Magenta, Cían y Amarillo. Y al mezclar de forma uniforme todos los colores primarios se obtiene el color blanco. Un ejemplo de la implementación de este sistema de colores son los televisores, en los cuales al combinar los tres colores básicos en diferentes intensidades se obtienen otros muchos colores, y al mezclar varias fuentes de luz se obtiene una imagen.

El sistema anterior de color puede explicarse y aplicarse desde el punto de vista de una fuente de luz primaria, sin embargo, cuando se trata de una fuente de luz secundaria los colores primarios y secundarios se invierten, la razón de esto es que en la naturaleza cuando la luz incide en un objeto, el objeto absorbe ciertas longitudes de onda a través de moléculas llamadas pigmentos, a esta absorción se le puede considerar una sustracción de longitudes de onda. Como resultado de esta sustracción la luz reflejada $c(\lambda)$ se percibe de cierto color, por ejemplo en una manzana roja, su color se percibe como rojo debido a que sus pigmentos absorben las demás componentes espectrales de la luz. Al sistema que describe los colores a partir de la absorción se le conoce como sistema de color sustractivo, tal que:

$$C(\lambda) - C_n(\lambda) = C_m(\lambda)$$

Siendo $C(\lambda)$ el espectro de luz visible, $C_n(\lambda)$ Todas las componentes absorbidas por el objeto, y $C_m(\lambda)$ las componentes reflejadas. Entonces como ya comentaba, los colores primarios quedan como Amarillo, Magenta y Cían. Al realizar diferentes combinaciones igualmente se pueden generar un gran rango de colores, y al mezclar a los 3 en proporciones iguales el color resultante es el negro.
A continuación dos diagramas que ilustran tanto el sistema aditivo como el sustractivo con sus respectivos colores primarios y secundarios:

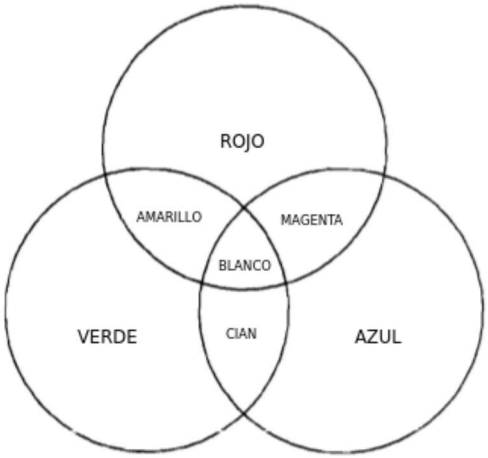


Figura 2.11: Espacio de color aditivo (RGB).

Fuente: Lim (1990)

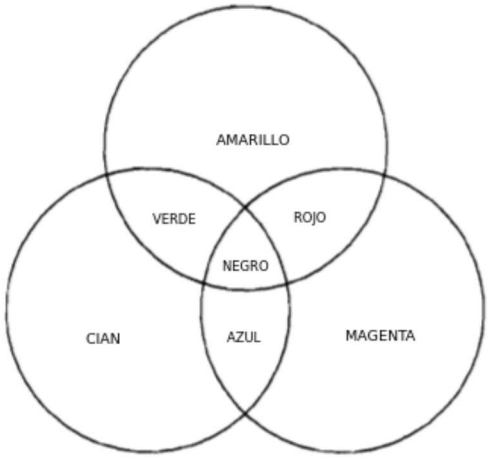


Figura 2.12: Espacio de color sustractivo (YCM).

Fuente: Lim (1990)

Se puede ver que en efecto los espacios de color son de características inversas. En el espacio aditivo el negro corresponde a ausencia de luz, y se agregan colores hasta tener una mayor cantidad de longitudes de onda, terminando en el color blanco. Inversamente en el espacio sustractivo el blanco corresponde a ausencia de pigmentos, y se va disminuyendo la propagación de las longitudes de onda hasta terminar en la luz reflejada, teniendo como menor propagación al negro.

Desde un punto de vista del procesamiento de señales, en un sistema aditivo de colores, se puede considerar que el color azul, rojo y verde se obtienen después de filtrar la luz blanca con sus respectivos tipos de filtros paso-banda, y que la combinación de dos colores se obtiene al realizar la luz blanca por un arreglo en paralelo de dos filtros del tipo ya mencionado. En contraste en un sistema sustractivo el color amarillo, cían y magenta resultan de travesar la luz blanca por los respectivos filtros elimina-banda (notch), y para la combinación de colores se realiza el arreglo de estos últimos filtros mencionados en configuración de cascada o serie.

El ojo humano

El ojo humano es un sistema biológico muy complicado, no solo por el proceso que realiza de transformar la luz a una señal neurológica, sino también por la cantidad y tamaño de los elementos que lo componen, debido a esto es que el conocimiento sobre este sistema se basa en un modelo derivado mayormente de estudios prácticos ejercidos sobre la funcionalidad del ojo, desde este punto de vista funcional se puede decir que el ojo es un dispositivo esférico que proyecta la luz visualizada en la parte posterior del mismo, para después convertirla a

señales cerebrales. A continuación se muestra una imagen transversal del modelo de un ojo:

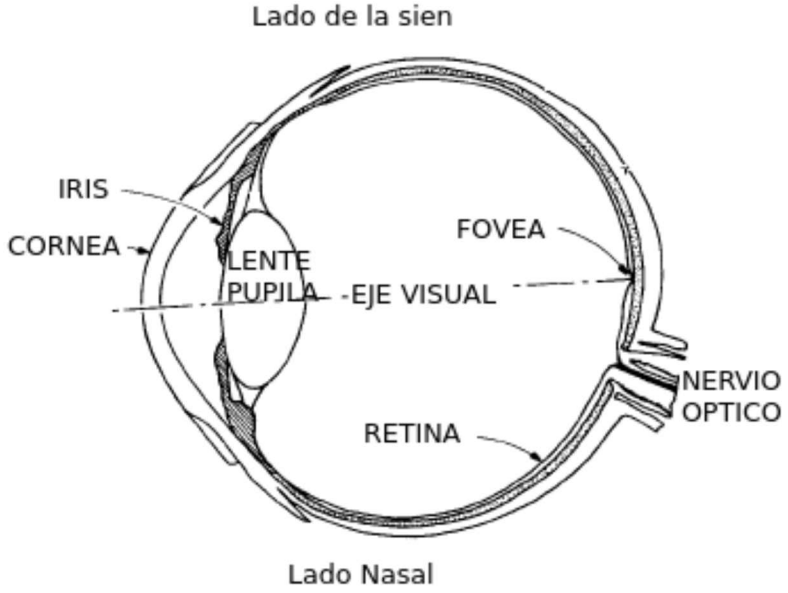


Figura 2.13: Diagrama trasversal del ojo

Fuente: Gonzalez y Woods (2002)

La composición del ojo se puede listar de la siguiente forma:

- Hasta el frente del ojo se encuentra la cornea, la cual es una membrana transparente la cual tiene la función de refractar la luz, pues debido a su forma redondeada puede funcionar como una lente convexa.
- A continuación sigue la pupila, la cual se encarga de cambiar de tamaño la abertura que tiene en el centro, a esta abertura se le conoce como iris, y sirve para regular la cantidad de luz que logro pasar al ojo.
- Atravesando el iris se encuentra el cristalino (o lente), su función es la de enfocar de forma correcta la luz que atravesó las capas anteriores, el cristalino cambia su forma debido a la presión efectuada por músculos que lo rodean y con esto es capas de cambiar el foco de la escena actualmente vista
- Las capas anteriormente descritas son para acondicionar la imagen debida a la luz, la capa siguiente es la retina y a diferencia de las anteriores esta sirve para atrapar la luz acondicionada en su superficie, esta capa receptiva se compone de dos tipos de células foto-sensibles:

  Bastones: Ofrecen visión escotópica, la cual es la que percibe niveles muy bajos de iluminación. Con este tipo de visión no se puede discriminar el color, debido a esto

los bastones solo ven en escala monocromática. Además de la sensibilidad a la luz, otra característica de los bastones es que al encontrarse muy cerca de la periferia de la retina, generan una visión periférica. Hay aproximadamente 100 millones de bastones en la retina.

Conos: Son menos sensibles que los bastones, por lo que responden a un nivel alto de iluminación, generando la visión fotópica. Con esta visión se interpreta el color de la luz proyectada dentro del ojo, existen como tal tres tipos de conos, cada uno con una diferente respuesta a la λ, por lo que detectan diferentes colores, teniendo conos para el color rojo, verde y azul. Como se puede ver en la sección transversal hay una zona central llamada fóvea, entre mas cerca se esta de esta zona, mas aumenta la densidad de conos y disminuye la de bastones, lo que en contraste con la organización de los conos y su visión periférica, se puede decir que los conos ofrecen una visión mas detallada y concentrada. Hay aproximadamente 7 millones de conos en la retina. A continuación una gráfica que muestra las diferentes capacidades en λ para sensar la luz transmitida al ojo por parte de los conos y bastones:

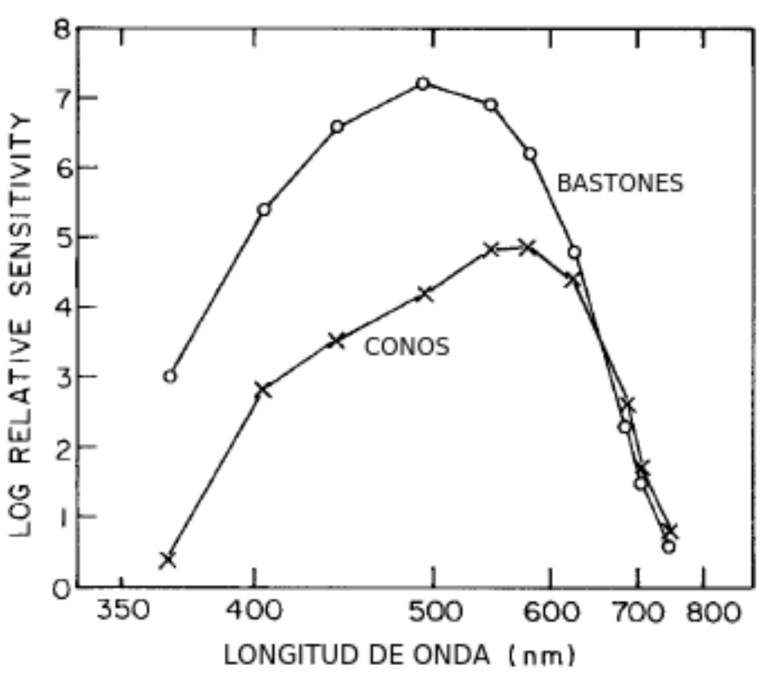


Figura 2.14: Rango de luz absorbida por los conos y bastones

Fuente: Pratt (2001)

- Por ultimo quisiera destacar la presencia del nervio óptico, el cual sale a través de un pequeño agujero en el ojo, lo que genera la existencia de un punto ciego pues es imposible que existan conos o bastones en esa zona. En el diagrama siguiente se muestra la localización de dicho punto ciego, así como la distribución de los conos y bastones existentes en la retina:

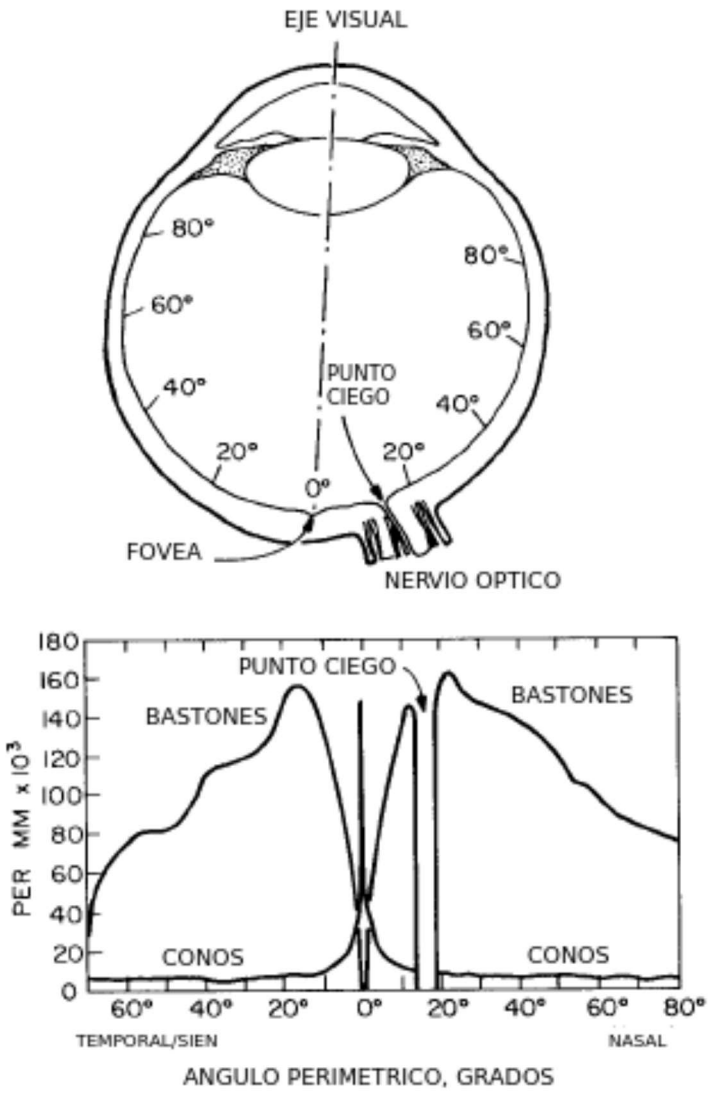


Figura 2.15: Distribución del ángulo de visión humana

Fuente: Pratt (2001)

Una vez descrito el ojo, queda por mencionar la forma en la que este se comunica con el cerebro para procesar los datos de la visión. Como ya se menciono, las células de la retina procesan la información entregada por la luz, dicha información se transporta a través de conexiones de nervios ópticos de los cuales se sabe existen aproximadamente 1 millón, y como similarmente se tiene conocido que existen mas de 100 millones de las mencionadas células, resulta lógica la deducción de que se comparten cierta cantidad de nervios. Lo que se piensa actualmente es que los conos sobre la fóvea están conectados individualmente para tener una mayor agudeza visual, mientras los bastones siempre comparten su conexión, por lo que a causa de esto la agudeza visual nocturna (coordinada por la visión escotópica) no es muy buena pese a que hay mas bastones que conos. Posteriormente a las interconexiones,

se sabe que los nervios de cada ojo se juntan en una rama, esta rama individual de cada ojo se entre cruza con la rama del ojo opuesto, se cree que esta particularidad es en parte responsable de nuestra capacidad de visión estereoscópica, pero debido al poco conocimiento de la propagación de las señales nerviosas a través de los nervios ópticos, no se tiene un modelo determinístico del proceso de visualización. A continuación un diagrama que muestra al sistema de visión humana desde el punto de vista de dos sistemas en cascada, teniendo como primer sistema al nivel periférico compuesto por el ojo y sus ya mencionadas partes, las cuales desembocan en un segundo sistema el cual se encarga de procesar a nivel neuronal para extraer la información necesaria.

Figura 2.16: Diagrama del sistema de visión humano

Fuente: Jähne (2002)

2.2.2. Representación digital de una imagen

A partir del ultimo diagrama de la sección anterior, se puede componer una analogía desde el punto de vista del procesamiento digital de señales, teniendo como entrada una imagen o escena natural, la cual se transforma en una señal eléctrica la cual se digitaliza, para que posteriormente pueda procesarse con algunos algoritmos y mostrar finalmente solo ciertas características deseadas:

Figura 2.17: Diagrama de un sistema de procesamiento digital de imágenes

Fuente: Jähne (2002)

Para obtener una representación digital de una imagen, se suelen tener los siguientes apartados de acciones necesarias previas a la representación final...

Dispositivo digitalizador o de entrada

Este dispositivo de entrada es el primer paso del sistema, es el que se encarga de convertir la imagen visualizada en impulsos eléctricos, hay varias formas de realizar esto a partir de la

recepción de la luz en un ambiente tridimensional, ya sea irradiada o reflejada. Para detectar la energía irradiada se utilizan sensores elaborados con material sensible a cambios generados por cierta longitud de onda electromagnética, en este caso del espectro de luz visible.

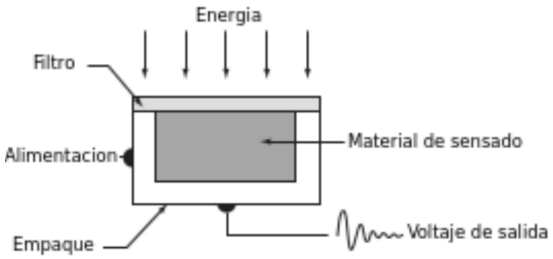


Figura 2.18: Modelo básico ejemplificando un sensor CCD

Fuente: Gonzalez y Woods (2002)

actualmente un sensor muy utilizado para la detección de los cambios energéticos de la luz, es el conocido como "Dispositivo de Carga Acoplada" (CCD por sus siglas en ingles), este sensor de estado solido al exponerse a la luz carga paquetes eléctricos de forma proporcional a la intensidad de luz que incide en él. Para crear un sensor de imagen completo a partir de este CCD lo que se hace es una matriz bidimensional discreta con los sensores, con aproximadamente 4000∗4000 elementos o mas, teniendo que así cada sensor es un punto de la imagen a generar. La ventaja de realizar esto es que todo el tiempo se tiene conocimiento de la posición del sensor y por tanto de la posición de los elementos que conforman gráficamente a la imagen, además de que al estar todos sobre el mismo arreglo, todos se activan respecto a la luz vista en la misma escena y la imagen se genera a partir de esa activación simultanea, sin tener que preocuparse de tener puntos muertos en la adquisición de datos.

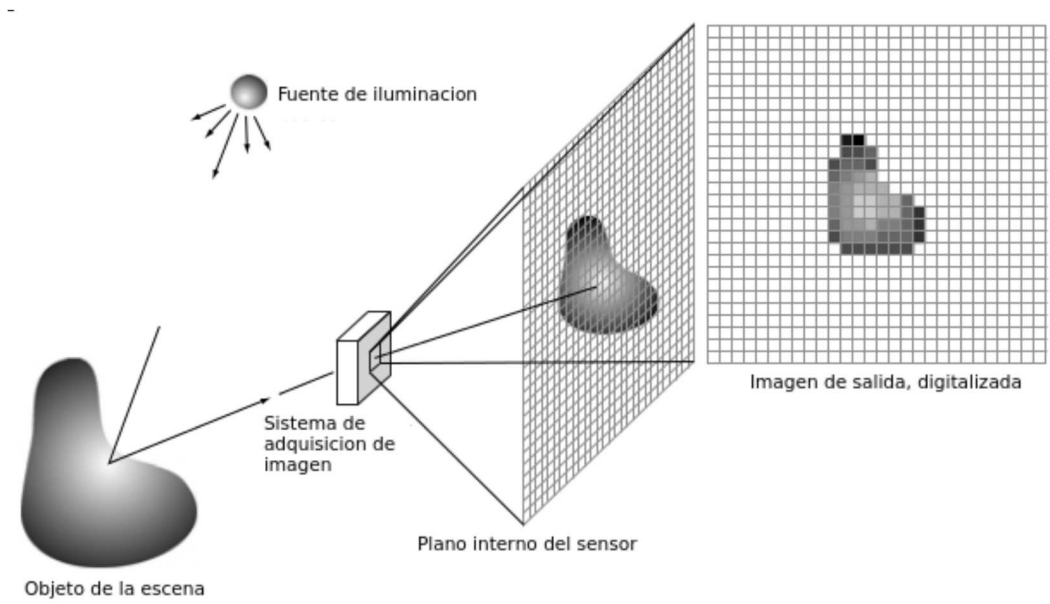


Figura 2.19: Diagrama de una matriz de sensores CCD y su recepción de datos

Fuente: Gonzalez y Woods (2002)

Algo a contemplar sobre estas mencionadas matrices de sensores es que usualmente se tiende a tener varios arreglos puestos uno sobre otro, de tal forma que el arreglo mas externo sensa directamente al ambiente, mientras que los posteriores funcionan como buffer, una vez que se sensa algo, estos datos pasan al siguiente arreglo y así sucesivamente, esto sirve para ir formando una imagen mas nítida en tiempo sin depender tanto del tamaño espacial del arreglo de sensores, ya que entre mas superficie tenga la matriz de sensores mayor sera la resolución, además de esto, la orden de sensado en la matriz principal debe ser ejecutada cada cierto tiempo para generar dicha imagen, al intervalo de tiempo entre cada sensado se le conoce como tiempo de muestreo de la imagen.

Muestreo y Cuantización

Como se puede ver a partir de la , de una escena continua, se quiere obtener una representación digital. Desde el plano de visión del sensor, dicha escena tiene componentes en las coordenadas x y y, así como una amplitud marcada por la intensidad de los diferentes datos enviados por la fuente de luz, para convertir todo esto digitalmente, se puede decir que el digitalizar los valores en el eje coordenado es el muestreo, mientras que la versión digital de los valores de amplitud de la luz son la cuantización.

Desde el punto de vista de la practica el método de muestreo se determina por el sistema de sensado utilizado para generar la imagen, como ya mencione, la matriz de sensores CCD

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Figura 2.20: Representación matricial de una imagen digital

sensan de forma simultanea toda la escena, por lo que se tendrá una recolección de datos homogénea en el eje coordenado, además el limite de muestreo para el eje x y y para una imagen es establecido por el numero de sensores en el arreglo, esto es relativamente un problema ya que la teorema del muestreo dicta que el muestreo mínimo debe ser el doble de la frecuencia de la señal, característica que en este caso viene determinada por una cantidad espacial y no temporal de frecuencia, por lo que existen muchas formas distintas de compensar esto a nivel de adquisición de la imagen.

Cuando se sensa con la matriz CCD, no hay movimiento sobre los sensores, por lo que puede realizarse un almacenamiento del valor de las localidades de cada elemento, para que cada cierta cantidad de muestras se promedie el valor de cada sensor individual, teniendo así una cuantización mas exacta que genere una imagen mas nítida, pues al realizar este tipo de compensaciones se evita tener elementos extraños a causa de movimiento en la escena.

Representación de la imagen

Posteriormente a la adquisición de datos la imagen obtenida se representa ahora en forma de un arreglo matricial bidimensional, el cual tiene un valor de localidad y una posición establecida análoga a la posición del sensor que capturo esa parte de la escena. El arreglo es de tamaño $M * N$, donde M es la cantidad de datos en la imagen para el eje y, y N es la cantidad para el eje x, a cada uno de estos datos dentro del arreglo se le denomina pixel. Los parámetros en los que se puede dividir el conocimiento sobre un píxel se pueden clasificar dentro de dos categorías, el "Donde" y el "Que"; el primero se refiere a la ubicación espacial que le corresponde dentro de la representación digital de la imagen bidimensional, este parámetro se puede expresar sencillamente en función del M y N que dan la posición del elemento del píxel dentro de la matriz. Para el caso del "Que", se refiere a los datos que se representan en esa zona de la imagen, los cuales generalmente van relacionados con la ya comentada percepción de la luz, y además de esto se pueden asignar datos correspondientes a otro tipo de consideraciones que se tengan para cualquiera que sea el proceso que se lleve a cabo, como por ejemplo, peso probabilístico, profundidad, ángulos, etc.

Un ejemplo del contenido del píxel es el color de la luz percibida, este se representa por 24 bits teniendo 8 bits para cada una de las componentes primarias del espacio de color en el que se este trabajando, si solo se representara una imagen en escala de grises entonces solo se

requerirían 8 bits, los cuales podrían representar 256 diferentes niveles de gris. Para calcular la cantidad total de bits para almacenar una imagen conociendo su tamaño espacial, y la cantidad de bits para las características contenidas, se puede utilizar la siguiente formula:

$$b = M * N * k$$

Con k como la cantidad de bits en los que se pueden almacenar los ya mencionados datos y b como la cantidad de bits necesarias para su almacenamiento.

2.2.3. Imagen tridimensional

Cuando se proyecta una escena del medio ambiente, al proyectarla en una imagen convencional no se considera la componente de proyección visual que otorga a los objetos el volumen que se percibe en el entorno físico, por lo que se pueden generar confusiones en ciertos análisis de las imágenes si lo que se quiere es interactuar con el medio externo a la cámara. Como tal, la caracterización tridimensional de una escena física se pueden obtener a partir de dos representaciones: De profundidad y volumétrica, la segunda siendo una versión mas complementada que la primera, al basarse en esta y añadir otras características. A continuación explico brevemente las bases de estas representaciones.

Representación por profundidad

Para obtener la profundidad de una superficie se requiere mas información además del brillo obtenido en cada píxel, dicha información puede adquirirse al aplicar técnicas basadas en los siguientes tres principios:

- Profundidad por triangulación
  Si se observa un objeto desde dos puntos de vista separados por cierta distancia sobre una base, el objeto tendrá un ángulo diferente hacia dicha base a partir de ambos puntos de vista. Un ejemplo de aplicación de este principio como técnica es la estereometría, de la que hablare mas adelante.

- Profundidad por tiempo de vuelo
  Se parte de tener una señal que se propaga desde cierta fuente y dicha señal lleva una velocidad característica hacia el objeto a medir, al llegar a él la señal se refleja hacia un receptor con lo que posteriormente se tiene que el tiempo total del viaje es proporcional a la suma de las distancia entre la fuente y el objeto con la distancia entre el objeto y el receptor. Usualmente el emisor y el receptor se colocan uno a lado del otro, de forma que la distancia es casi la misma y por tanto el tiempo total de viaje es la mitad de

dicha suma. Un sensor que aplica directamente este principio para medir distancias es el ultrasónico.

- Profundidad por fase o interferometría
  Se basa en parte en el principio anterior, solo que en este caso se mide la distancia como múltiplo de la longitud de onda de la onda radiada, al medir no solo la amplitud de esta sino también su fase. Es posible medir la fase a partir de la superposición de la radiación coherente generada cuando dos frentes de onda sobrepuestos tienen un desfase de 180º.

- Perfil por sombreado
  El perfil de una superficie puede ser determinada a partir de la orientación de la sombra proyectada, debido a que es matemáticamente posible calcular el ángulo de incidencia de la fuente de iluminación a partir de una imagen, con lo que se obtendría una profundidad aproximada del perfil calculado. Este principio como técnica es usada entre otras usualmente cuando se cuenta con un sistema de visión monocular o tradicional.

Representación volumétrica

Para crear un volumen se debe contar con una técnica que pueda medir varias profundidades al mismo tiempo, de tal forma que sea posible reconstruir un objeto visto. Además de los ya mencionados principios para obtener profundidad, se aplican otros dos principios para una representación de volumen:

- Corte por iluminación.
  Este principio parte de que si podemos mover activamente la fuente de luz, de tal forma que se sepa en que zona del objeto se esta variando, entonces es posible conocer que zonas de profundidad existen debido a los cambios de sombra que se registraran al mover dicha fuente, una vez que se tienen estas zonas se puede obtener una imagen volumétrica a partir del patrón de irradiación de luz.

- Profundidad por múltiples proyecciones o tomografía
  Una proyección de profundidad solo contiene información parcial de un objeto volumétrico la cual es correspondiente a solo una vista o cara del mismo. Para obtener como tal el volumen de dicho objeto es posible combinar diferentes piezas parciales de tal forma que se complete la imagen tridimensional.

2.2.4. Estereoscopía y representación de profundidad

De acuerdo con Castleman (1995) es la determinación de la distancia hacia un objeto a partir de la observación de una escena desde dos diferentes puntos de vista. Similarmente, a

la configuración de dos sensores de imagen sobre un mismo eje se le llama sistema "estereo" o estereoscopio, el cual se basa en la analogía a muchos sistemas de visión biológicos, estos sensores deben estar colocados con los ejes ópticos paralelos entre si, y al vector de distancia entre ambos se le llama base estereoscópica o linea base.

Al juntar los datos obtenidos, el objeto sera proyectado en diferentes posiciones de un mismo plano de imagen, esto es debido a que es visto dese diferentes ángulos a partir de la linea base. Esta diferencia en las posiciones del mismo objeto se le conoce como disparidad o paralelaje, se puede calcular de la siguiente forma:

$$p = b * (\frac{d'}{X_3})$$

Se puede decir entonces que el paralaje (p) es inversamente proporcional a la distancia (X_3) hacia el objeto desde la base estereoscópica de longitud (b), y es directamente proporcional a dicha base y la distancia focal de las cámaras (d'). De esto se puede deducir que entre mas lejano este el objeto mas difícil se vuelve encontrar dicha distancia, si se tiene una distancia X_3 infinita, el paralelaje tiende a cero. Expresándolo con la ley de la propagación del error, se tiene que:

$$X_3 = b * \frac{d'}{p} \rightarrow \sigma_{X_3} = b * \frac{d'}{p^2}\sigma_p = \frac{(X_3)^2}{b*d'}\sigma_p$$

Por lo que la sensibilidad absoluta para la estimación de profundidad de la distancia X_3, disminuye proporcionalmente a la distancia misma al cuadrado. Por ejemplo, si se tiene un sistema estereoscópico con linea base de 200mm y lentes con una longitud focal de 100mm entonces a una distancia de 10m, el cambio en el paralelaje sera de 200 μm/m o 20 píxeles/m, lo que es aun computable visto desde el punto de vista de la representación de una imagen. Mientras que si con las mismas especificaciones se intenta calcular una distancia de 100m, el cambio en el paralelaje sera de 2 μm/m o 0.2 píxeles/m.

Una forma de representar la distancias observadas a partir de disparidad es sobreponiendo las imágenes del sensor izquierdo y derecho, cada una con una diferente tonalidad, obteniendo así una representación simple de un modelo tridimensional, sin embargo esto no tiene mucha utilizad si se analizan imágenes a partir de su color. Debido a situaciones como la anterior se desarrollo un estándar conceptual de una imagen, llamado mapa de profundidad, el cual se basa en tomar una imagen en algún espacio de color o escala de grises, y la relaciona a algún punto de vista con eje coordenado definido, por lo que se almacenara en su representación no solo la ubicación de los píxeles y su contenido cromático, sino también los datos de profundidad correspondientes a cada píxel, en distancias de X, Y, Z y los ángulos entre ellas. Una forma mas simple de representar la profundidad es considerando la linea base de donde parte la visión estereoscópica como el eje coordenado, viendo de frente a la imagen, teniendo que para interactuar con los datos no sera necesario definir los ángulos.

CAPÍTULO 3

El robot Nimbro-OP y las adaptaciones realizadas

Como se menciono anteriormente, este trabajo se aplico a un robot humanoide similar a los previamente descritos, este robot, conocido como "Nimbro-OP", o simplemente Nimbro, fue desarrollado por el Instituto para Ciencias Computacionales de la Universität Bonn (2012), localizada en Alemania, y presentado por primera vez al publico en exhibiciones el año 2012. Actualmente es la versión prototipo para el robot "igus" de la misma universidad. Nimbro se creo como una iniciativa de la ya mencionada universidad como una plataforma abierta para la investigación y desarrollo, por lo que además de poder adquirirse mediante una compra, también fácilmente se puede tener acceso a sus especificaciones de manufactura, hardware y software, lo que facilito la adaptación de este trabajo a la arquitectura ya existente del robot.

3.1. Hardware

El hardware del robot se encuentra contenido en una estructura a modo de cuerpo hecha con fibra de carbono, aluminio y algunas pieza de plástico ABS, y las partes que requieren alimentación eléctrica la obtienen a través de una batería de litio polímero (LiPo) de 14.8 volts , 3.6 amperios por hora. Mas haya de esto, el demás hardware que lo conforma se puede dividir en la aplicación de las 3 primitivas de la robótica ya antes mencionadas:

3.1.1. Actuadores

Los actuadores que este robot posee son servo-motores marca Dynamixel, los cuales poseen encoders para ayudar en la propiocepción, pues estos entregan la posición en la que cada actuador se encuentra, de tal forma que sea posible realizar el control de las

extremidades del mismo, la estructura de Nimbro se conforma por dos tipos de actuadores: el Dynamixel MX-106 con torque máximo de 10 Nm (Newton por metro) y el Dynamixel MX-64 con torque máximo de 7.3 Nm.

Así mismo, el robot solamente posee 20 actuadores, de los cuales doce son del tipo MX-106 localizándose 6 en cada pierna, con los demás de tipo MX-64, teniendo 3 por brazo y 2 en la cabeza. La distribución de los actuadores se debe principalmente a las diferentes concentraciones de esfuerzos en la estructura.

Figura 3.1: Dynamixel MX-64

Figura 3.2: Dynamixel MX-106

Además de estos actuadores, el robot cuenta con un sub-controlador que realiza la función de interconectar a los anteriores con la unidad central de procesamiento, ya que este recibe las ordenes de la computadora para posteriormente traducirlas y enviarlas a cada motor. Este sub-controlador es de la marca Dynamixel, con modelo "CM-730", el cual fue originalmente creado para otro robot, conocido como "Darwin-OP", sin embargo los desarrolladores de Nimbro realizaron las adecuaciones necesarias para que funcionase con el.

Figura 3.3: Tarjeta sub-controladora CM730

Dentro del contexto de esta Tesis no se realizo ninguna modificación o integración nueva a este sistema de actuadores.

3.1.2. Procesamiento

El robot cuenta con una mini-computadora, la cual realiza todas las funciones de procesamiento del robot, debido a las nuevas implementaciones que se realizaron al robot tanto en materia de esta tesis como también otras desarrolladas, en el laboratorio de Bio-Robótica de la U.N.A.M. al que pertenece, se decidió remplazar la mini-computadora ya existente en el robot con una que cumpliera la demanda de la cantidad de procesamiento necesario. Para elegir la nueva computadora además de la ya mencionada necesidad de un mayor poder de procesamiento, se tomo como criterio principal que la nueva PC igualara las características de alimentación eléctrica a la ya existente, y además tuviera un tamaño similar o menor a la misma de tal forma que no se tuviese que modificar el hardware del robot, ya que como se menciono poco antes, los actuadores no dependen de la mini-PC sino de un sistema intermediario, por lo que la decisión de cambiar el sistema de procesamiento no afecto el funcionamiento del sistema de actuadores.
A continuación en la tabla 3.1 se muestra una comparación de las características y especificaciones de la computadora que originalmente tenia el robot Nimbro (Zotac ZBOX nano XS),

con algunas computadoras propuestas que cumplian los requisitos necesarios, entre ellas la PC que fue elegida (MintBox Min).

computadoras	*Procesador*	*Gráficos*	*Memoria RAM*	*Memoria interna*	*Precio*
Zotac ZBOX nano XS	AMD Dual-Core Processor E-450	AMD Radeon HD 6320	2GB SO-DIMM DDR3-1333	64GB	359 USD
MintBox Mini	AMD A4 Micro-6400T SoC	AMD Radeon R3	4GB SO-DIMM DDR3L	64GB	295 USD
Intel NUC6i5SYK	intel core i56260U	Intel Iris 540	Ranura DDR4	Ranura SDXC	377 USD
fitlet-iA10 Barebone	AMD A10 Micro-6700T	AMD Radeon R3	Ranura SO-DIMM DDR3	Ranura mSATA	299 USD

Cuadro 3.1: Diferentes mini computadoras consideradas para el robot humanoide

Al final la elección fue la mintbox mini debido a que además de cumplir los requisitos necesarios mencionados, tenia un precio menor al de la computadora original duplicando las características de esta, teniendo así una mejora sobre el robot manteniendo incluso una consistente mejora en el precio de sus componentes. El hecho de haber encontrado una PC que mejorara a la ya existente en precio y características de operación se debe al cambio de la tecnología desde que el robot fue diseñado hasta la implementación de proyectos sobre él por parte del laboratorio.

Figura 3.4: miniPC Zotac ZBOX

Figura 3.5: miniPC MintBox Mini

La única adaptación necesaria para colocar la miniPC en el robot, es el de adaptar el área de montaje de la PC, ya que esta es mas chica que la que el robot contenía originalmente, basado en esto y el conocimiento de las medidas del robot, pude hacer una base para la mintbox y montarla en el robot, esta base se detalla en el ultimo apéndice.

3.1.3. Sensado

El robot Nimbro posee los siguientes sensores:

- Encoder rotatorio: Como tal este sensor viene ya incorporado en cada uno de los actuadores marca Dynamixel previamente mencionados, estos sensores se encargan de entregar la posición de cada servo-motor entre los 0 y 360 grados geométricos.

- Acelerómetro: Este sensor viene incorporado en la tarjeta CM-730, el sistema de control lo utiliza para poder obtener la aceleración en sus 3 ejes de movimiento a partir del centro del robot, con lo que se podría realizar un algoritmo para compensar cualquier des-balance en el desplazamiento del mismo.
- Giroscopio: También incorporado en la CM-730, este sensor se utiliza mayormente para reportar la orientación del robot respecto al plano en el que se desplaza, y si este se encuentra en posición horizontal se ejecuta algún algoritmo de levantamiento físico para incorporarse en posición vertical.

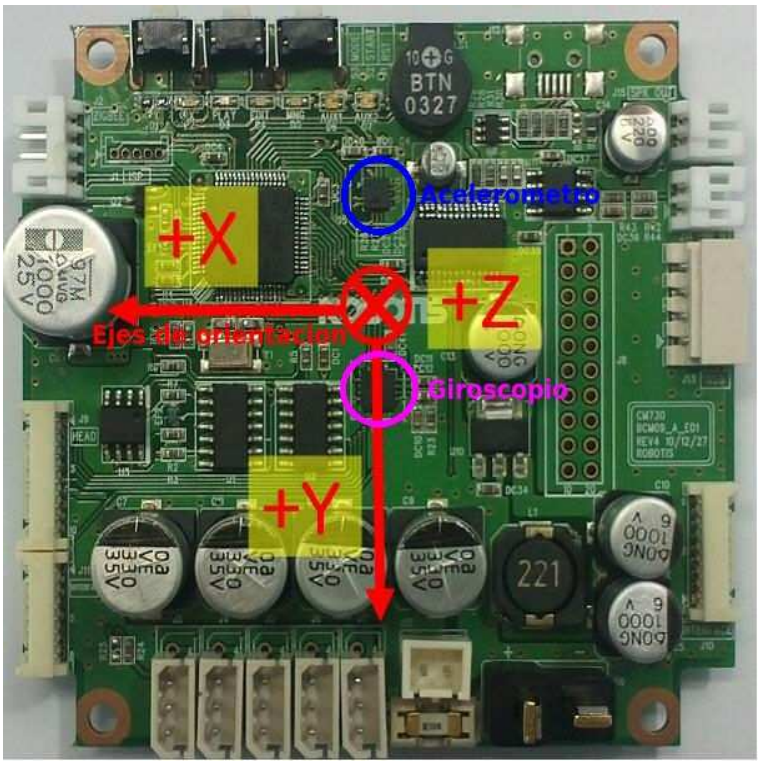


Figura 3.6: Sensores acelerómetro y giroscopio

- Cámara: El robot originalmente contaba con una webcam portátil de la marca *Logitech*, modelo *C905*, esta cámara como tal es bastante buena, sin embargo debido a la ya explicada necesidad de una visión estereoscópica, y por tanto una cámara estereoscópica, procedí a investigar sobre posibles reemplazos.

  Una cámara estereoscópica es un instrumento que tiene dos cámaras montadas de forma rígida sobre una plataforma, la cual las mantiene con una separación constante conocida como linea base, que es usualmente considerada como una similitud con la separación entre los ojos humanos. Además estas cámaras usualmente tienen un sistema de transmisión que es capaz de manejar una gran cantidad de datos, dígase de otra forma, un ancho de banda alto, esto es debido al envío de datos simultáneos por las dos cámaras, por lo que usualmente cuentan con el estándar IEEE 1394 Firewire. Además

de lo anterior, este tipo de cámara al ser compuesta, debe tener una sincronización por hardware, lo que significa que un solo pulso activa a ambas cámaras para iniciar su transmisión, esto es esencial pues de no estar sincronizadas no podrán utilizarse en un entorno altamente dinámico pues siempre estarán desfasadas.

Para seleccionar el reemplazo considere 3 cosas principalmente: Precio accesible, compatibilidad con el sistema operativo (Existencia de drivers principalmente), y compatibilidad con el hardware, esto ultimo se debe mayormente a que las mini-pc difícilmente se consiguen con puerto firewire, por lo que este tipo de cámaras quedaron descartadas automáticamente. A continuación en 3.2, las cámaras consideradas y una configuración estereoscópica realizada con la original:

cámaras	*Resolución máxima @ 30fps*	*Tamaño*	*Peso*	*Fabricante*	*Precio*
C905 x2 (original x2)	1600 x 1200	-	632 g	Logitech	268 USD
Minoru 3D	640 x 480	8.3 x 5.7 x 3.8	250 g	P.D.T.	24 USD
ZED	3840 x 1080	6.89 x 1.18 x 1.3	159 g	StereoLabs	595 USD
DUO M	752 x 480	2.1 x 1 x 0.5	6.5 g	CodeLabs	299 USD

Cuadro 3.2: Diferentes cámaras estereoscópicas consideradas para el robot humanoide

Figura 3.7: Cámara monocular original.

Figura 3.8: Cámara estereoscópica elegida.

Tras la búsqueda de cámaras estereoscópicas que cumpliesen el requisito, solo una poca cantidad resultaron no ser firewire, sin mencionar que al parecer la única ventaja del diseño de esos dispositivos es la velocidad de transmisión pues su resolución máxima no es muy diferente a la de las demás mostradas en las tablas. La mayoría de la cámaras realmente estereoscópicas tienen un precio bastante elevado, con excepción de la cámara "Minoru 3D", la cual, aunque no tiene una resolución muy impresionante, es una cámara estereoscópica ya ensamblada la cual evita algún posible problema de hardware sobre la linea base por intentar unir dos cámaras, además de que "Minoru 3D" ya incluye circuitos internos para enviar la imagen sincronizada por un solo puerto USB, lo que ahorra programar la sincronización de cámaras individuales.

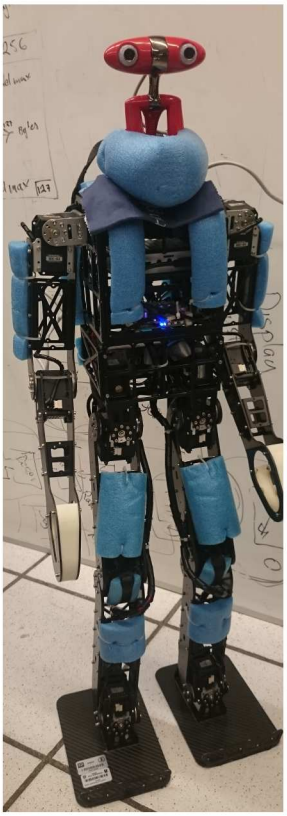


Figura 3.9: Nimbro después de las modificaciones

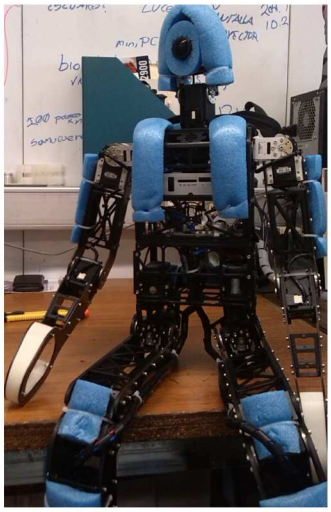


Figura 3.10: Nimbro con su hardware original

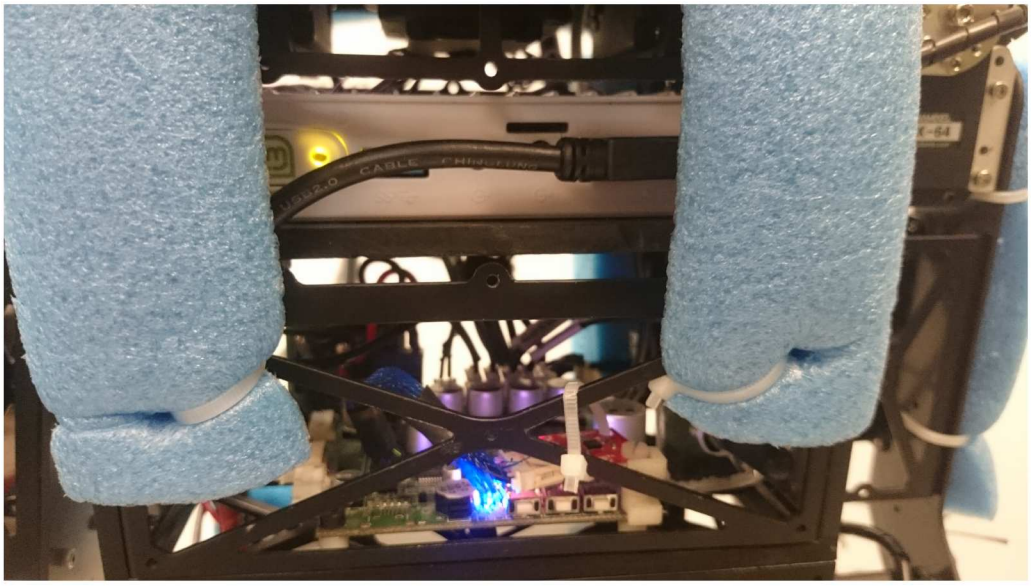


Figura 3.11: Mintbox colocada en el robot

3.2. Software

El software con el que contaba originalmente el robot y sus cambios se pueden dividir en las siguientes categorías:

- Middleware ROS:
  Debido a que el robot posee una gran cantidad de subsistemas , se requiere de algún programa que los auxilie en su interacción, este es el caso del sistema "ROS" (Por *Robotics Operative System)*, que aunque no es como tal un sistema operativo dedicado a la robótica, es actualmente un software muy utilizado en el área ya que además de proveer ya algunos módulos de ejecución, se encarga de realizar la respectiva interconexión entre los subsistemas necesarios previamente mencionados.
  Cabe mencionar que Nimbro cuenta con ciertos paquetes específicamente para la versión de ROS "hydro", los cuales ya se encargan de comunicarse con la tarjeta CM-730, los actuadores, la PC, la visión computacional y sensores. Estos paquetes especiales y ROS Hydro fueron instalados en el robot posteriormente a la modificación de la PC que se menciono, su instalación se describe mayormente en la sección de Apéndices A y B.

- Sistema operativo:
  Originalmente se contaba con una distribución de Linux Ubuntu 12.10, debido a que era la versión mas reciente de esa distribución cuando los creadores de Nimbro integraron el software, sin embargo, no era una versión de soporte extendido, por lo que para cuando el robot Nimbro llego al laboratorio en el que desarrolle la tesis, el sistema ya no tenia soporte por parte de los desarrolladores. Debido a esto, instale la distribución de Linux Ubuntu 12.04, que aunque es de numeración menor a la que el robot contaba, fue creada como versión de soporte extendido, por tanto aun me fue posible integrar varias librerías que necesité para el desarrollo de la visión estereoscópica.

  La razón de integrar un sistema operativo de numeración menor con soporte, en lugar de uno simplemente mas reciente, es que la versión de ROS antes mencionada no es compatible con las versiones recientes de dicho SO, y aunque hay nuevas versiones de ROS, son los paquetes para Nimbro los que requieren esa versión especifica de ROS. Por lo que, en resumidas cuentas, una edición mas actual de sistema operativo no hubiese permitido reutilizar los controladores ya existentes del robot y se hubiese necesitado mas desarrollo en esa área, pero mantener la versión 12.10 del SO tampoco era opción por el ya descontinuado mantenimiento, por lo que la respuesta de utilizar Ubuntu 12.04 con soporte extendido resulto lógica.

- Entorno gráfico:
  Tanto el sistema operativo original del robot como el actual poseen el entorno gráfico "LXDE", este entorno y su elección podrían considerarse no tan importantes, pues solo se utiliza para desarrollar las aplicaciones, pues se desactiva en la ejecución prolonga-

da de programas por el robot. Este entorno gráfico se caracteriza por su ligereza en términos de memoria y procesamiento necesarios para ejecutarlo.

- Configuraciones para comunicarse con el sub-controlador CM-730
  Como mencione posteriormente, existe un conjunto de programas (ROS) que se encargan de interconectar los subsistemas del robot, entre ellos el sub-controlador, lo único que se necesita realizar son configuraciones para que el sistema operativo detecte esta tarjeta electrónica tal y como lo haría con cualquier periférico conectado por USB, una vez detectada correctamente, ROS se encarga de enviar los parámetros necesarios para su funcionamiento. Dicha configuración se encuentra en el apéndice G.

Hay que constatar que el software anteriormente listado es el necesario para que el robot funcione posteriormente a las modificaciones de hardware, de tal forma que el funcionamiento se mantenga igual al estado original del robot. El software necesario para la realización de la visión estereoscópica lo menciono en la siguiente sección del presente trabajo.

A continuación un diagrama funcional del software y hardware de Nimbro:

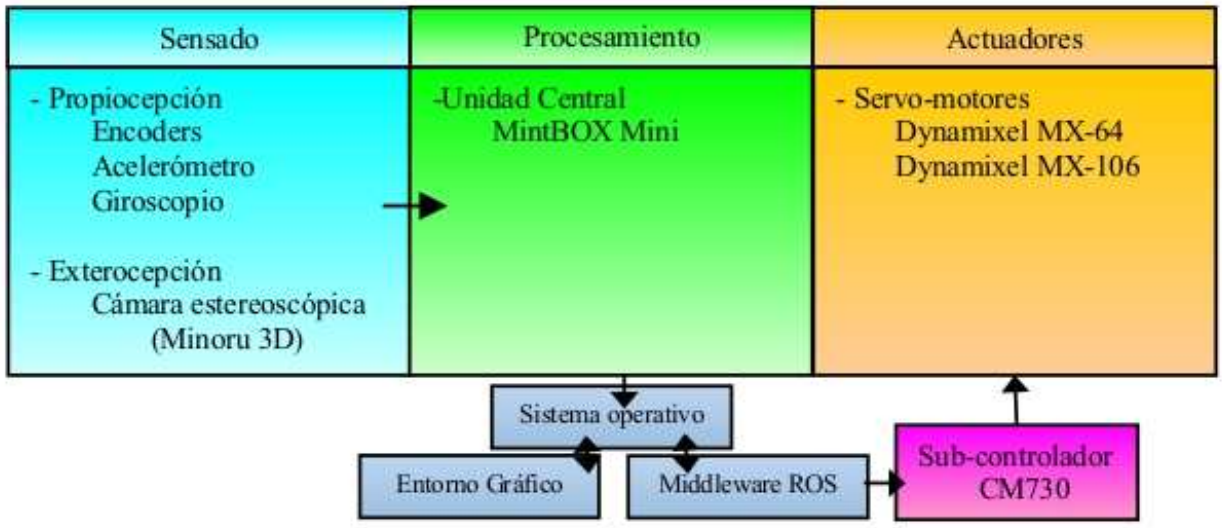


Figura 3.12: Diagrama de los componentes de Nimbro.

Por ultimo a este apartado de hardware, quisiera agregar que además de haber generado las modificaciones necesarias al robot, y la aplicación del nuevo sistema de visión, implemente un sistema de acceso distancia a la computadora del robot, por medio de SSH, VNC y una pantalla táctil, de tal forma que el mantenimiento del sistema del robot pudiera realizarse mas fácilmente. Estas configuraciones las abordo en los apéndices D, E y F.

CAPÍTULO 4

Visión estereoscópica

4.1. Caracterización de la cámara estereoscópica

Como se menciono previamente, el ensamblaje de una cámara estereoscópica en términos simples se compone por dos cámaras simples (o monoculares), las cuales se unen con una distancia entre si llamada linea base, y estas deben apuntar hacia el frente de la linea en todo momento. Desde el punto de vista de la transmisión, deben estar sincronizadas para el envío de imágenes. Respecto a estas características mencionadas, las correspondientes a la cámara Minoru 3D son las siguientes:

Descripción	*Valor*
Ángulo visual	40 grados
Longitud focal	3 milímetros
Linea base	60 milímetros
Formato	1/3 pulgadas
Apertura	F2.0
Resolución máxima	640x480 a 15 F.P.S.

Cuadro 4.1: Características de la cámara Minoru 3D

Algunos de estos parámetros son importantes debido a la necesidad de conocer la óptica de la cámara para hacer un calculo correcto de la imagen obtenida, tal y como comentaba en el marco teórico, sin embargo, a continuación detallo el proceso que lleve a cabo para obtener de una manera mas computacional las relaciones que de otra forma deberían ser obtenidas a partir de la geometría de la cámara.

4.2. Procedimiento teórico

Primero se describirá el procedimiento realizado para generar la nube de puntos, esta se obtuvo a partir de las imágenes obtenidas por las cámaras que componen a la "Minoru 3D", posteriormente se obtiene el plano principal de la escena tomada en la nube de puntos, y finalmente se discriminan los objetos restantes sobre el ya encontrado plano. A grandes rasgos los pasos fueron los siguientes:

1. Calibración o corrección de la distorsión: Esto se efectúa para eliminar la distorsión radial y tangencial debido a los lentes de la cámara y la geometría entre ambos.
2. Rectificación: Se realiza una transformación para proyectar las imágenes de ambas cámaras en un mismo plano de tal forma que queden en la configuración paralela frontal necesaria. Este paso requiere que las cámaras se ajusten en ángulo y distancia, debido a que "Minoru 3D" ya se encuentra ajustada físicamente el error no es tan grande, sin embargo siempre es necesario realizar la rectificación debido a cuestiones de fabricación y óptica del lente. Al terminar este proceso se contara con un par de imágenes alineadas en sus filas.
3. Correspondencia: En este paso se encuentran las mismas características en ambas imágenes obtenidas por la cámara, y las alinea de tal forma que queden en la misma columna. Al termino de este paso se obtiene el llamado mapa de disparidad.
4. Re-proyección: Al conocer la estructura geométrica de las cámaras se puede realizar triangulación sobre el mapa de disparidad y así obtener una representación de los puntos sobre las geometrías detectadas, y su distancia hacia la cámara.
5. Obtención del plano: Aquí elegí aplicar el algoritmo RANSAC sobre la nube de puntos ordenados previamente obtenida.
6. Detección de objetos: Esta parte se divide principalmente en dos...

   Segmentación escena-plano: Debido a que ahora se conoce el plano mas grande, procedí a restar el plano a la escena completa.

   Detección de objetos sobre el plano: Finalmente, con el conocimiento del plano dominante de la escena y los objetos existentes, se puede aplicar una eliminación de los objetos que no se encuentren directamente sobre dicho plano, además, las geometrías que no cumplan con cierta densidad son descartadas para evitar falsas detecciones.

4.2.1. Corrección de la distorsión

Para corregir la distorsión de una cámara, es necesario obtener primero algunos parámetros asociados con la configuración estereoscópica, estos son:

- Las matrices intrínsecas de ambas cámaras
- La matriz de distorsión debida a las cámaras
- El vector de rotación y translación entre las 2 cámaras (desfase estereoscópico)

Esto se puede hacer a través de una calibración de la cámara, al calibrarla estos parámetros ya mencionados se colocan en el punto óptimo esperado.

Calibración individual de las cámaras

Al calibrar una cámara, se encuentra su matriz intrínseca, coeficientes de distorsión, rotación y vectores de translación, todo respecto al sistema coordenado del objeto visto. Para este sistema de coordenadas se necesitan ciertos puntos del objeto que puedan ser aislados para interpretarse, esto ultimo se realiza usualmente al calibrar con la imagen de un tablero de ajedrez a partir de la implementación del algoritmo de Zhang (1999).

Una vez que se obtienen estos puntos mencionados, es necesario también establecer algunos puntos manualmente, de forma mas especifica el punto de origen, y una vez que se tienen estos datos es posible obtener los vectores de rotación y translación. La razón por la que se necesita que el usuario defina manualmente el origen, es debido a que a partir de este origen se calcularan todas las distancias del mundo real, tomando además como unidad de medida lo que el usuario especifique con ese origen. Por ejemplo, si se toma como origen la esquina del tablero de ajedrez, y al primer punto de los recuadros del tablero como la coordenada (1,0), entonces todo se medirá todo en unidades del tamaño de ese recuadro.

Para calibración se siguen los siguientes pasos:
Partimos de encontrar la matriz de homografía H desde la siguiente ecuación.

$$P_{dist} = Hp_{sec}$$

Donde P_{dst} y P_{scr} son las posiciones de un punto en el lector de imagen, y el plano de ajedrez respectivamente. Las homografías de todos los puntos aislados pueden ser encontradas a partir de esto, y posteriormente promediadas para obtener la homografía de la imagen. Recordando, una homografía es una transformación que determina la correspondencia entre dos figuras geométricas planas. Tenemos que entonces la matriz de homografía es:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$$

En donde "s" es el factor de escala y M la matriz intrínseca; h_1 y h_2 las homografías de los puntos previamente mencionadas; r_1 y r_2 los vectores de rotación sobre los ejes "X" y "Y" respectivamente, hay que notar que no se considera el eje "Z" ya que de inicio se supone que el plano visualizado es paralelo a nuestra cámara y por tanto no hay desplazamiento en Z para ninguno de nuestros puntos; "t" es el vector de translación. Estos 3 vectores últimos en mención representan parámetros entre la cámara y los orígenes del tablero de ajedrez.
Ahora, la matriz M...

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

f_x y f_y son las longitudes focales de la cámara, mientras que c_x y c_y representan el desplazamiento entre el punto donde el eje óptico tiene una intersección tanto el plano de la imagen como el centro lógico del plano de la imagen. Reemplazando M en H y despejando, tenemos lo siguiente:

$$r_1 = \nabla M^{-1} h_1$$
$$r_2 = \nabla M^{-1} h_2$$
$$t = \nabla M^{-1} h_3$$

Donde $\bigtriangledown = \frac{1}{s}$ debido al despeje realizado, donde s es el factor de escalamiento.
Ahora considerando la siguiente propiedad del producto interno; $v^T * v = v_{(1x1)}$ y a los dos vectores de rotación como ortonormales (r_1 y r_2), tenemos:

$$r_1^T r_1 = r_2^T r_2$$

Sustituyendo los vectores de rotación previamente despejados, tenemos finalmente 2 ecuaciones, que se utilizaran como restricciones de diseño para calcular nuestros puntos focales:

$$h_1^T M^{-T} M^{-1} h_2 = 0$$

$$h_1^T M^{-T} M^{-1} h_1 = h_2^T M^{-T} M^{-1} h_2$$

Ahora realizando la operación matricial $M^{-T} * M^{-1} = B$, tal que B tenga solución cerrada:

$$B = \begin{bmatrix} \frac{1}{f_x^2} & 0 & \frac{-c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & \frac{-c_y}{f_y^2} \\ \frac{-c_x}{f_x^2} & \frac{-c_y}{f_y^2} & \frac{c_x^2}{f_x^2} + \frac{c_y^2}{f_y^2} + 1 \end{bmatrix}$$

Recordando nuestras restricciones, podemos sustituir B tal que: $(h_j)^T B h_j$. Como pudimos ver justo antes, la matriz B es simétrica, y por propiedades de simetría, puede escribiste como un producto vectorial de seis dimensiones...

$$h_i^T B h_j = v_{ij}^T b = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix}$$

Ahora las restricciones, considerando que $v_{11} = (h_1)^T B h_1 =$, $v_{22} = (h_2)^T B h_2 =$ y $v_{12} = (h_1)^T B h_2 =$, se pueden escribir como:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0$$

Como se puede ver en las relaciones de $v_{ij}b$ que se acaban de definir como producto punto, podemos notar que para cumplir nuestras restricciones necesitamos varias homografía. Al aplicar estas restricciones en mas de 2 imágenes ya adquiridas se puede obtener b y posteriormente B, para finalmente despejar las longitudes focales y puntos centrales:

$$\begin{gathered} f_x = \sqrt{\lambda / B_{11}} \\ f_y = \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)} \\ c_x = -B_{13} f_x^2 / \lambda \\ c_y = (B_{12}B_{13} - B_{11}B_{23}) / (B_{11}B_{22} - B_{12}^3) \end{gathered}$$

Tal que

$$\lambda = B_{33} - (B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23}))/B_{11}$$

Con los centros y focos se pueden obtener los vectores de rotación y translación en función de M...

$$r_1 = \lambda M^{-1} h_1$$
$$r_2 = \lambda M^{-1} h_2$$
$$r_3 = r_1 x r_2$$
$$t = \lambda M^{-1} h_3$$

Así como el factor inverso de escalamiento

$$\nabla = 1/\left\| M^{-1} h_1 \right\|$$

Hay mayormente dos tipos de distorsión:

1. Radial: Debido a la forma de los lentes.

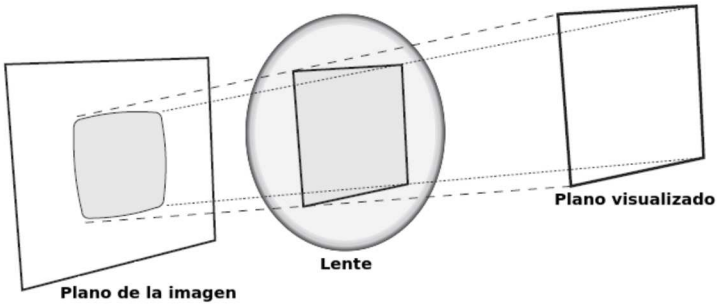


Figura 4.1: Distorsión radial

2. Tangencial: A causa de que los lentes no estén paralelos al sensor de imagen

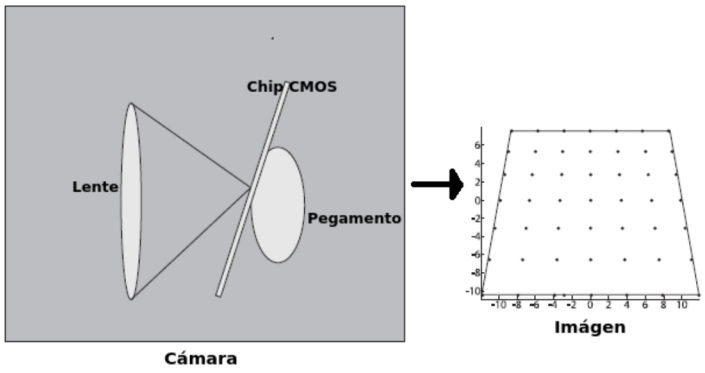


Figura 4.2: Distorsión tangencial

A continuación ambas distorsiones se modelan en una sola ecuación, la radial se representa por k_1, k_2 y k_3, mientras que la distorsión tangencial por p_1 y p_2. Considerando a (x_p, y_p) como la posición n-ésima de la cámara si esta fuese perfecta, y (x_d, y_d) la posición distorsionada, podemos calcular la distancia real usando los coeficientes recientemente mencionados:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ p_1(r^2 + 2y_d^2) + 2p_2 x_d y_d \end{bmatrix}$$

Calibración del desfase estereoscópico

Como ya mencione, para efectuar la calibración necesitamos la siguiente información:

- Los vectores de rotación R y translación T entre las dos cámaras
- Las matrices intrínsecas de ambas cámaras
- ...Y sus matrices de distorsión

Para cualquier punto P ubicado en coordenadas tri-dimensionales, se puede realizar la calibración individual de cada cámara para colocar a P en coordenadas de la misma, y al unirlas tendremos una representación tal que: $P_l = R_l P + T_l$ para la izquierda y $P_r = R_r P + T_r$ para la derecha. Ambos puntos de vista de P se relacionan a través de $P_l = R^T * (P_r - T)$, por lo que tomando esta ecuación y ambos puntos, podemos obtener las siguientes relaciones en términos de la rotación y translación.

$$R = R_r(R_l)^T$$
$$T = T_r - RT_l$$

Esta ecuación se puede aplicar a varias imágenes obtenidas del mismo tablero de ajedrez, calculando para cada una tanto R como T, aunque debido a posibles errores cada tablero tendrá diferentes R y T, podemos considerar a la mediana de esos valores como el valor de R y T del desfase estereoscópico, que en conjunto con la expresión de matriz intrínseca y de distorsión de la sección anterior, se utilizan para modelar completamente este desfase.

4.2.2. Rectificación estereoscópica

En este apartado se busca re-proyectar el plano de la imagen de ambas cámaras de tal forma que se encuentren en uno solo, y una vez que las imágenes estén en el mismo plano, se alinearan en la misma fila visual, para posteriormente recortar el plano mostrado a solo la parte en la que los dos planos originales se traslapan. La figura siguiente (4.3) muestra lo que intento expresar en este apartado:

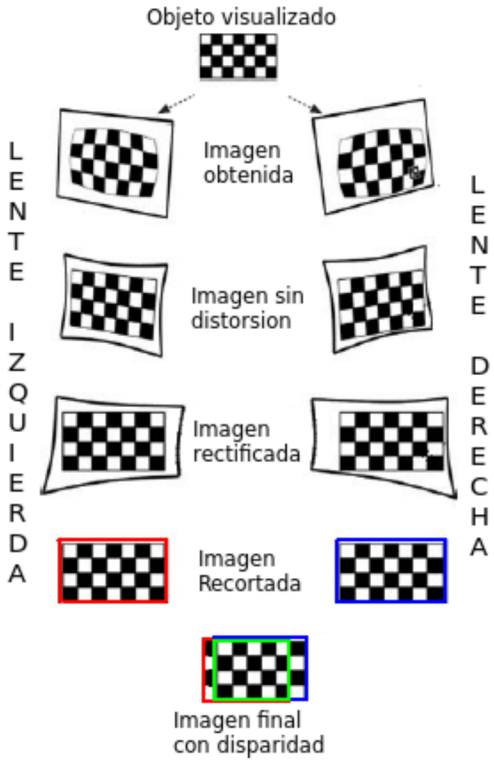


Figura 4.3: Rectificación de las 2 imágenes entregadas por la cámara estereoscópica

Se necesita delimitar el rango de procesamiento para que al buscar el objeto visto, se

pueda hacer de forma mas eficiente en un bloque de datos reducido y que además es coincidente en columnas. También porque al solo buscar en un área en el que ambas vistas se traslapan nos asegura una existencia del objeto en el rango de visión.

A continuación describo brevemente dos de los algoritmos existentes para realizar rectificación y al final un método para el mismo fin:

Algoritmo de Hartley

Este intenta encontrar homografías que lleven los epipolos al infinito así como a las lineas epipolares, minimizando las disparidades entre las dos imágenes debido a puntos de coincidencia.

- Ventajas:

  Reduce la complejidad computacional

  Se puede realizar solo observando los puntos de la escena

- Desventajas:

  No entrega representaciones de la escala

Algoritmo de Bouguet

Dada la matriz de rotación y translación ya obtenida previamente, este algoritmo trata de minimizar la cantidad de cambio que producirá la re-proyección de las imágenes al plano final, con lo que también disminuirán la cantidad de distorsiones finales y maximizando así el área visual obtenida.
Este algoritmo es el utilizado debido a que el resultado final obtenido debe ser mas preciso, propiedad que es muy necesaria debido a que el robot en el que se aplicara, basado en el caso de estudio del RoboCup, debe encontrar una pelota a la mayor distancia que sea posible.
El procedimiento del algoritmo es el siguiente:

1. Para minimizar la distorsión final, la matriz de rotación correspondiente al plano de la cámara derecha, que sera rotada sobre el plano izquierdo, es dividida a la mitad entre las 2 cámaras, de tal forma que ambos planos roten y no solo uno. Estas nuevas matrices de rotación se llamaran r_l y r_r respectivamente. Cada cámara rota entonces la mitad, de tal forma que sus proyecciones principales terminan paralelas al vector suma de donde estaban apuntando originalmente. A la rotación obtenida la nombramos como R_{rect}.

2. La rotación descrita anteriormente pone a las cámaras en alineación co-planar, pero no en correspondencia de sus filas. Para calcular la rotación que también realice esto, se calcula el R_{rect} que lleve el epipolo de la cámara izquierda (e_1) hasta infinito, de tal forma que las lineas epipolares queden horizontalmente, entonces la matriz de rotación se crea iniciando por la dirección del epipolo e_1 ya mencionado. Tomando como punto principal buscado al origen de la imagen izquierda (c_x, c_y), la dirección del epipolo atravesara directamente el vector de translación entre los centros de proyección de ambas cámaras, este vector se denominara "rayo principal", pues es el que esta justo en medio de ambos centros y debe proyectar al punto real visualizado.

$$\mathrm{e} = \frac{T}{\|T\|}$$

3. El siguiente vector por definir, es e_2, el cual sera ortogonal a e_1 y a su vez al "rayo principal". Esto se realiza con el producto cruz en la dirección del "rayo" y normalizando.

$$\mathrm{e}_2 = \frac{[-T_y T_x 0]}{\sqrt{T_x^2 + T_y^2}}$$

4. Ahora se calcula un tercer vector e_3 que sera ortogonal a los 2 anteriores

$$\mathrm{e}_3 = \mathrm{e}_1 x \mathrm{e}_2$$

5. Por lo que la matriz de rotación tan mencionada quedara finalmente como

$$R_{rect} = \begin{bmatrix} (\mathrm{e}_1)^T \\ (\mathrm{e}_2)^T \\ (\mathrm{e}_3)^T \end{bmatrix}$$

6. La alineación de ambas imágenes se obtiene al realizar la siguiente ecuación, recordando que r_l y r_r son las matrices de rotación individuales.

$$R_l = R_{rect} r_l$$
$$R_r = R_{rect} r_r$$

7. Ahora para las matrices de proyección izquierda y derecha, se tiene lo siguiente:

   - Izquierda.

$$P_l = M_{rect-l} P_l' = \begin{bmatrix} f_{x-l} & \alpha_l & c_{x-l} \\ 0 & f_{y-l} & c_{y-l} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Derecha.

$$P_r = M_{rect-r}P'_r = \begin{bmatrix} f_{x-r} & \alpha_r & c_{x-r} \\ 0 & f_{y-r} & c_{y-r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

M_{rect} es la matriz intrínseca pero de la rectificación, para recordar como se obtiene esta matriz, regresar al apartado referente a la matriz de homografía de una sola cámara. P' es una matriz diagonalizada de la proyección. Además, α_l y α_r son conocidos como el factor de "skew"(o distribución) para los píxeles, este factor usualmente es cero.
Con estas matrices de proyección se pueden transformar los puntos 3D a una representación en 2D, esto debido a que la primera esta ligara a una posición física real, mientras que la segunda a una posición "en pantalla"

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

Las coordenadas en pantalla se consideran como $(\frac{x}{w}, \frac{y}{w})$

8. Ahora inversamente, dado un punto bidimensional homogéneo con una disparidad asociada d, podemos proyectarlo en tres dimensiones con la siguiente matriz, conocida como de re-proyección, la cual es la inversa de la matriz de proyección. Ahora las coordenadas en 3D están dadas por $(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W})$.

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

$$Q \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c'_x)/T_x \end{bmatrix}$$

Mapa de rectificación

Para re-proyecciones muy grandes, es mas eficiente que el mapa de rectificación utilice la estructura de una tabla de búsqueda, en lugar de las ecuaciones matriciales, sin embargo

un mapa de este tipo se restringe al tipo de dato en el que este redactado, por lo que probablemente al realizar la búsqueda en los píxeles se tendrá un rango de perdidas debido al truncamiento de los datos, por lo que para evitar que la imagen rectificada tenga espacios "vacíos" se utiliza el promedio de los datos al rededor del píxel faltante, teniendo así una representación menos aproximada que con los modelos matemáticos.

4.2.3. Correspondencia estereoscópica

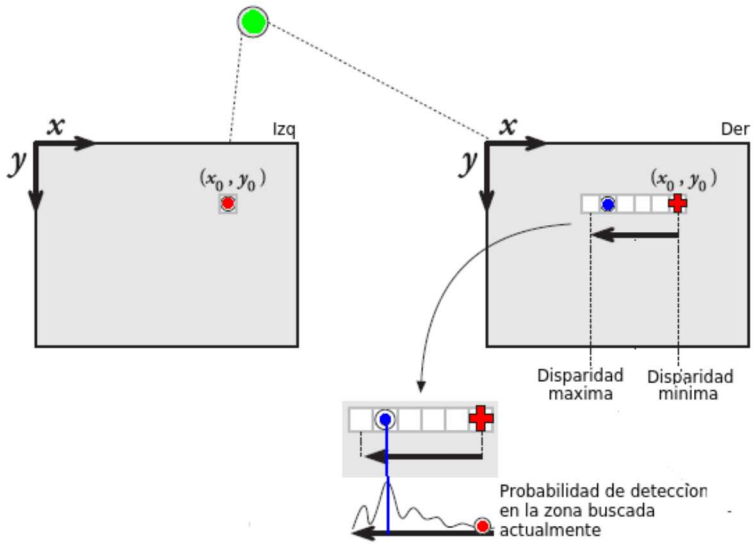
Una vez que las imágenes de ambas cámaras están en el mismo plano y alineadas, se encuentra la disparidad entre un punto que coincida en ambas imágenes. En este caso utilice el algoritmo "Block Matching", el cual encuentra correspondencias "fuertes" entre dos imágenes, basándose en el texturizado de la imagen. Este algoritmo conlleva los siguientes pasos:

1. Pre-filtrado: En esta etapa las imágenes se normalizan para reducir las diferencias de iluminación y así mejorar la textura percibida.

2. Correspondencia: Se calcula deslizando una ventana de "suma de diferencias absolutas" (Brown, Burschka, y Hager, 2003), descrita como:

$$\sum_{u,v} = \left| I_3(u,v) - I_2(u+d,v) \right|$$

   Para cada característica en la imagen izquierda, buscamos una en la imagen derecha dentro de la misma fila, ya que después de la rectificación, cada fila es una linea epipolar por lo que la posición en la imagen derecha deberá estar en la misma fila pero diferente columna en el lado izquierdo. En forma geométrica, si el píxel buscado se encuentra en una coordenada (X_0, Y_0) del lado izquierdo, entonces en el lado derecho se encontrara en la misma posición Y_0 con un desplazamiento hacia la izquierda a partir de X_0. Puede haber características no reconocidas, ya que los rangos de visión de cada cámara no son los mismos, a las características vistas en un lado pero no en el otro, se les denomina " inserciones ".
   El usuario puede configurar un valor mínimo y máximo de disparidad considerada, de tal forma que se tenga un rango para buscar dicha coincidencia. En la imagen siguiente se puede ver la representación de la utilidad del algoritmo de correspondencia.



El objeto visualizado es el verde, mientras que el rojo representa el visto por el lado izquierdo y el azul por el lado derecho, como puede verse hay cierto desfase, por lo que se requiere establecer un rango de disparidad a través del cual posteriormente se conocerá la disparidad real

4.2.4. Re-proyección

La disparidad obtenida al final del proceso anterior se puede utilizar para obtener la profundidad de dicho punto a partir de una triangulación, como se muestra en la siguiente imagen. Hay que recalcar que esto se realiza primero convirtiendo los puntos en la correspondencia 2D, a puntos 3D, a partir de emplear la matriz de re-proyección Q calculada previamente a partir del algoritmo de Bouguet.

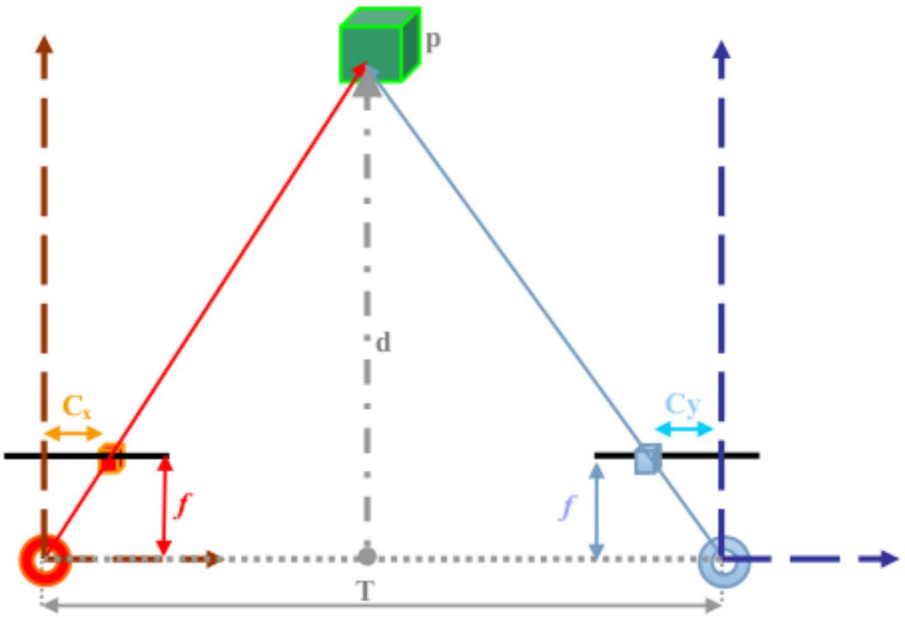


Figura 4.4: Diagrama de distancias en el sistema estereoscópico

Esta imagen muestra también los ejes ópticos y centros visuales de ambas cámaras. La longitud focal de cada uno, se denota por f; T es la linea base; C_y y C_x son las posiciones en el eje X, de la imagen proyectada en la izquierda y derecha respectivamente. d es la distancia desde la linea base hasta el objeto visualizado.

4.2.5. Obtención del plano y división de los objetos

Para obtener el plano, implemente el algoritmo "Random Sample Consensus(RANSAC)" para detección de planos, el cual, a partir de una muestra de 3 puntos, comienza a buscar el plano de mayor superficie dentro de la escena; la razón de que sean 3 puntos es que esta es la cantidad mínima para formar un plano. Posteriormente de definir los 3 puntos, RANSAC comienza a buscar cuantos puntos hay dentro de la superficie actualmente delimitada, para definir si un punto tiene o no pertenencia a dicha superficie se estima la distancia (ε) entre el punto probado, los que lo contienen y sus vecinos, a estos puntos internos a la superficie se le conocen como "inliers", este procedimiento se itera una cantidad de N veces y al final el plano mayor es el que contenga mas "inliers".
Para discriminar el cuando termina el plano computado se utiliza también la distancia ε previamente definida, teniendo que si un punto se encuentra muy lejos de otro obviamente no pertenecerá al mismo plano u objeto. Una forma sencilla de obtener esta distancia es, tal

y como expresa Baguley (2009), a través de la siguiente ecuación:

$$D_i(x,y,z) = \frac{|Ax+By+Cz+D|}{\sqrt{A^2+B^2+C^2}}$$

Donde A, B y C son vectores normales que definen al plano descrito por la ecuación geométrica $Ax + By + Cz + D = 0$, estos multiplican a la respectiva coordenada del punto (x, y, z) en evaluación de pertenencia. Siendo D un "offset" definido por el usuario.
Como se puede ver, la ecuación anterior calcula una distancia euclidiana para cada una de las componentes en el espacio en tres dimensiones pero solo entre un punto deseado y otros 3 vecinos. Este resultado se puede comparar posteriormente con un valor de umbral, y decidir finalmente si este punto pertenecerá o no al plano con el que se esta evaluando el punto.

$$Inlier(x,y,z) = \begin{cases} 1 & \textbf{if } D_i(x,y,z) \leq T_D \\ 0 & \textbf{if } D_i(x,y,z) > T_D \end{cases}$$

Donde T_D es el umbral de distancia.

Como tal las ecuaciones anteriores sirven para discriminar los puntos a través de sus distancias, sin embargo para representarlos de forma matricial y calcularlos con RANSAC, tal y como menciona Zuliani (2014), primero se plantea el sistema de ecuaciones de los puntos que representan al plano

$$\begin{cases} \theta_1 x_1^{(1)} + \theta_2 x_2^{(1)} + \theta_3 x_3^{(1)} + \theta_4 = 0 \\ \vdots \\ \theta_1 x_1^{(N)} + \theta_2 x_2^{(N)} + \theta_3 x_3^{(N)} + \theta_4 = 0 \end{cases}$$

Posteriormente se transforman a una representación matricial...

$$\begin{bmatrix} (x^{(1)})^T & 1 \\ \vdots & \vdots \\ (x^{(N)})^T & 1 \end{bmatrix} \theta = A\theta = 0$$

Y entonces se encuentra el vector que representa los parámetros del plano que contiene a los puntos $x_1, x_2 ... x_n$ de la matriz anterior.

$$\theta = argmin\|A\theta\|^2$$

A partir del vector de parámetros se puede obtener el "modelo espacial" del plano, sobre el cual operara RANSAC:

$$\mu = (\theta) \stackrel{\text{def}}{=} \left\{ x \in \Re^3 : \left[x^T 1 \right] \theta = 0 \right\}$$

En este caso, no se utiliza una vista simplemente frontal de la cámara, sino una del robot viendo ligeramente hacia abajo formando un triangulo entre el punto observado, la cámara y la posición del robot, recalco esto ya que es esta configuración la que permite que el algoritmo discrimine como plano mayor al suelo, teniendo así una forma viable y relativamente sencilla de obtener los puntos por sobre el plano observado sin tener que re-proyectar toda la nube de puntos.

Una vez detectado el plano, este se resta de la escena completa, teniendo como producto, una nube que contiene solo a los objetos que se encuentran por sobre este. Para encontrarlo primero se definen 3 puntos de la escena aleatoriamente, luego cada punto revisa a sus vecinos para saber si es que pertenecen al mismo plano, esto se realiza hasta obtener un plano cerrado. A este método se le denomina "RANSAC para planos".

Una vez obtenido el plano por este método, se resta a la imagen como ya se menciono, y a estos objetos restantes se les aplica una segunda segmentación que definí, esta es para evitar que los objetos restantes se traslapen entre si, dejando así solo objetos individuales, esto se realiza a partir de una segmentación de distancia por píxeles, la cual aunque tiene la desventaja de no considerar los colores, esta se vuelve irrelevante pues lo que se toma en cuenta es que tan lejana es la distancia euclidiana del punto evaluado respecto a los vecinos; Si un vecino tiene un valor de distancia mayor a un umbral definido, entonces ese píxel es descartado como parte de un mismo objeto, siguiendo así con el siguiente vecino hasta terminar esa región, pasando con la siguiente, hasta completar la imagen. Evidentemente si uno de los objetos posteriores a la extracción del plano tiene discontinuidades o partes de profundidad no uniforme, perderá partes con esta segunda segmentación, sin embargo no es un gran problema para el caso de estudio de la categoría de humanoides del RoboCup, ya que el balón que se quiere detectar al ser una esfera conlleva una continuidad homogénea de profundidad entre sus zonas, por lo que es un objeto que gracias a su geometría no es recortado siempre y cuando no se encuentre ocluido.

4.3. Procedimiento práctico: Programación

Debido a que "Minoru 3D" esta clasificada como "USB Video Class (UVC)", es posible utilizarla para desarrollar aplicaciones basadas en OpenCV, a continuación sigo los mismos pasos teóricos expresando ahora las funciones usadas para la practica, con excepción de la obtención del plano y su eliminación, pues esta parte la ejecute con la librería "v4l2stereo", cuya instalación describo en el Apéndice C:

4.3.1. Corrección de la distorsión

Las primeras dos funciones pertinentes a esta sección las aplico sobre varias imágenes del tablero de ajedrez, tomadas por ambas cámaras, de forma que estén calibradas tanto individualmente, como en conjunto a partir de la tercera función:

- cvFindChessboardCorners
  Esta función busca e identifica los bordes e intersecciones de los cuadrados encontrados dentro de un tablero de ajedrez, a partir de una imagen dada, para posteriormente entregar el numero de esquinas encontradas.
- cvFindCornerSubPix
  Esta función es similar a la anterior, pero antes convierte la imagen a escala de grises. Se utiliza para apoyar la cuenta de esquinas previamente obtenida, de tal forma que la ubicación de las mismas tenga una mayor precisión.
- cvStereoCalibrate
  Entrega la matriz intrínseca y los coeficientes de distorsión de ambas cámaras, esto se calcula a partir de un arreglo que describirá los puntos de las esquinas obtenidas de las varias imágenes procesadas con las 2 funciones anteriores.

4.3.2. Rectificación estereoscópica

- cvStereoRectify
  Para iniciar la rectificación, esta función encuentra y entrega los vectores de rotación y translación para colocar ambas imágenes en el mismo plano además de alinearlas. En conjunto a estos vectores, también se puede utilizar para obtener las matrices de proyección y de re-proyección pertenecientes al desfase estereoscópico.
- cvInitUndistortRectifyMap
  Crea un mapa que relacione los valores de la imagen obtenida con valores de posición en X y Y, esto se hace utilizando la matriz intrínseca, los coeficientes de distorsión, el vector de rotación y la matriz de proyección, de cada cámara respectivamente.
- cvRemap
  Re-ordena los valores de la imagen de tal forma que sean despreciables las curvaturas introducidas por el hardware de la cámara.

4.3.3. Correspondencia estereoscópica

- cvCreateStereoBMState
  Inicializa el algoritmo “Block Matching” que se utiliza para la correspondencia como

se menciono previamente.

- cvFindStereoCorrespondenceBM
  Genera el mapa de disparidad a partir de las imágenes rectificadas de las cámaras izquierda y derecha

4.3.4. Re-proyección

- cvReprojectImageTo3D
  Genera las coordenadas 3-D a partir de las coordenadas 2-D de la disparidad, usando los valores mismos de disparidad propia de los puntos existentes así como la matriz de re-proyección.

CAPÍTULO 5

Resultados

5.1. Visión obtenida

A continuación muestro el resultado de la visión en cada uno de los pasos realizados para obtener la representación por distancia, con el caso del balón de football utilizado por los robots humanoides:

- Imágenes obtenidas de forma simultanea directamente de la cámara

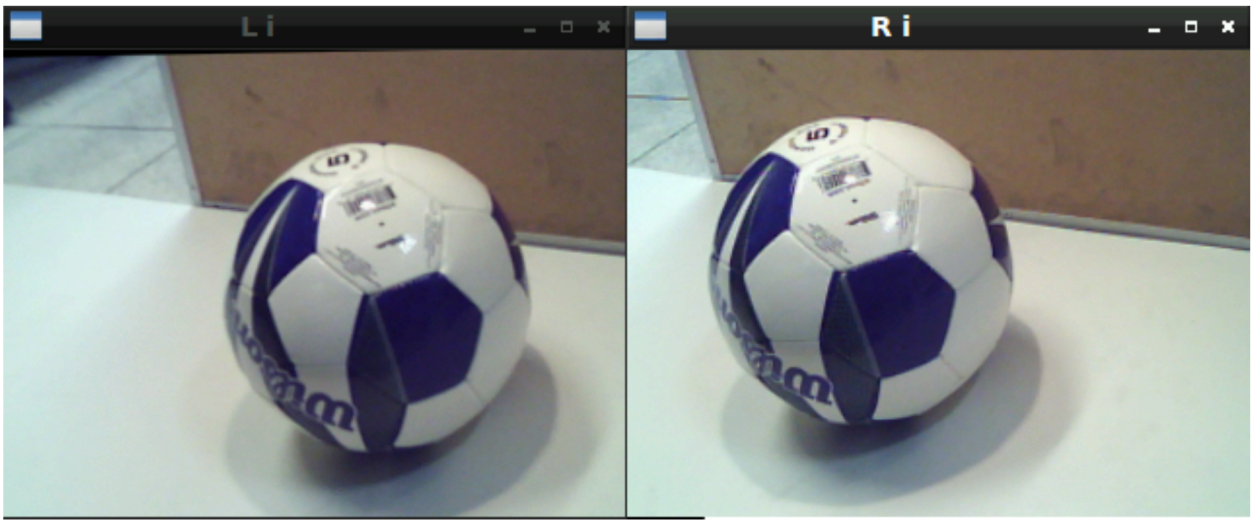


Figura 5.1: Imágenes obtenidas de la cámara estereoscópica

Como se puede ver, en efecto las imágenes tienen cierto desfase entre ellas, por lo que

se procede a rectificarlas y generar una representación por distancias.

- Nube de puntos

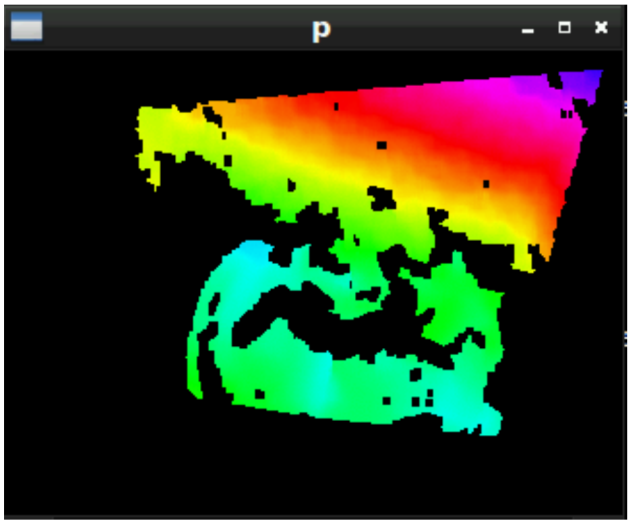


Figura 5.2: Nube de puntos

En esta imagen se puede ver el mapa de profundidad, marcado por color, yendo del azul como mas lejano, rojo intermedio, y verde mas cercano. Se puede notar que el plano aun se ve en la representación, por lo que tal vez el objeto no se distingue bien visualmente.

- Nube de puntos sin el plano

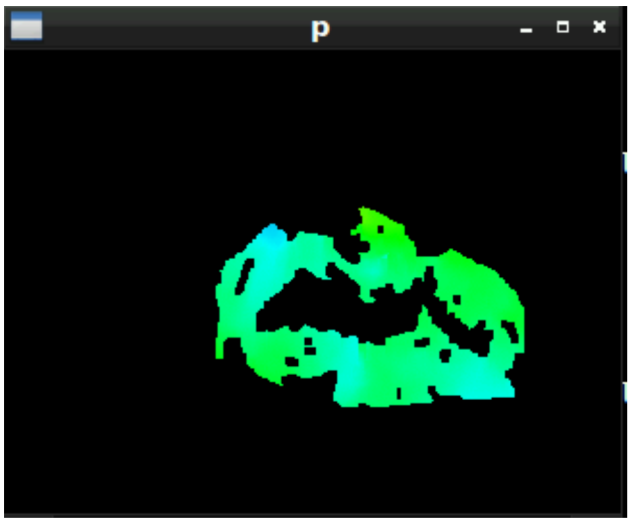


Figura 5.3: Nube de puntos sin el plano dominante

Una vez mas un mapa de profundidad con el mismo código de color, esta vez posteriormente a la extracción del plano por lo que se puede observar una menor cantidad de colores.

- Objeto sobre el plano

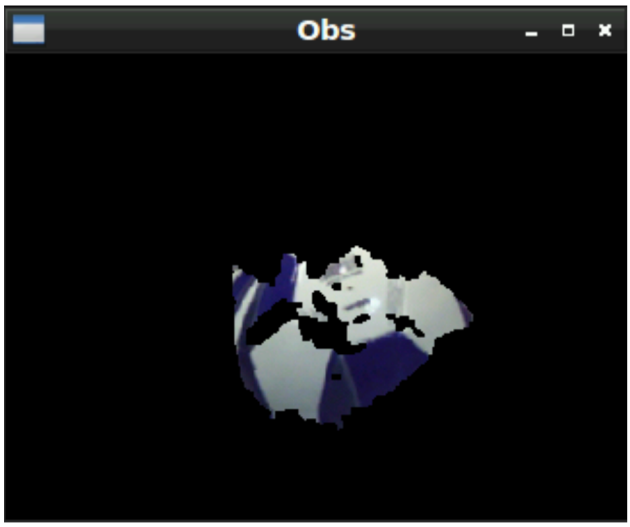


Figura 5.4: Objeto discriminado

Como puede verse finalmente en la imagen, una parte del objeto detectado (el balón), desaparece al terminar todos los procesos que definí, esto es debido principalmente a los parámetros utilizados para el detector de planos con RANSAC, la razón de esto es que el umbral de puntos causa que la parte del balón mas cercana al plano sea considerada como parte de este, por lo que la segmentación hacia el plano no es tan exacta, causando que en la resta del mismo esta parte del balón sea eliminada. En esta imagen final los datos de la nube de puntos final y la imagen de las cámaras se unen para tener una vista con color.

Para el umbral de distancia que elegí entre un objeto y el plano, así como entre objetos, considere un máximo de 0.26cm, los cuales calcule al despejar la variación de la sensibilidad de la disparidad dada por:

$$\sigma_{X_3} = \frac{(X_3)^2}{b*d'} \rightarrow X_3 = ((\sigma_{X_3}) * (b * d'))^{1/2}$$

Recordando que...

$$10\mu\text{m} = 1\text{píxel}$$

Considerando los parámetros de la cámara Minoru antes ya descritos, donde la linea base (b) de 60mm, longitud focal (d') de 3mm, y una sensibilidad de 10 píxeles. La razón de haber utilizado este calculo para determinar mi umbral de distancia máxima, es que esto me asegura que las distancias serán consistentes entre lo experimental y lo calculado de forma computacional, además de que estén en un rango entero de píxeles.

5.2. Diferencia entre distancia real y calculada

Colocando el objeto a distancias medidas experimentalmente, tome los datos de la distancia reportada en consola por el programa, y posteriormente realice una gráfica de puntos sobre la cual calcule una función de mínimos cuadrados para linealizar los resultados y poder obtener un modelo matemático de la distancia. El modelo obtenido es útil para poder calcular la relación con una distancia física real a partir de la distancia verdaderamente reportada. A continuación muestro dicha gráfica con la ecuación obtenida...

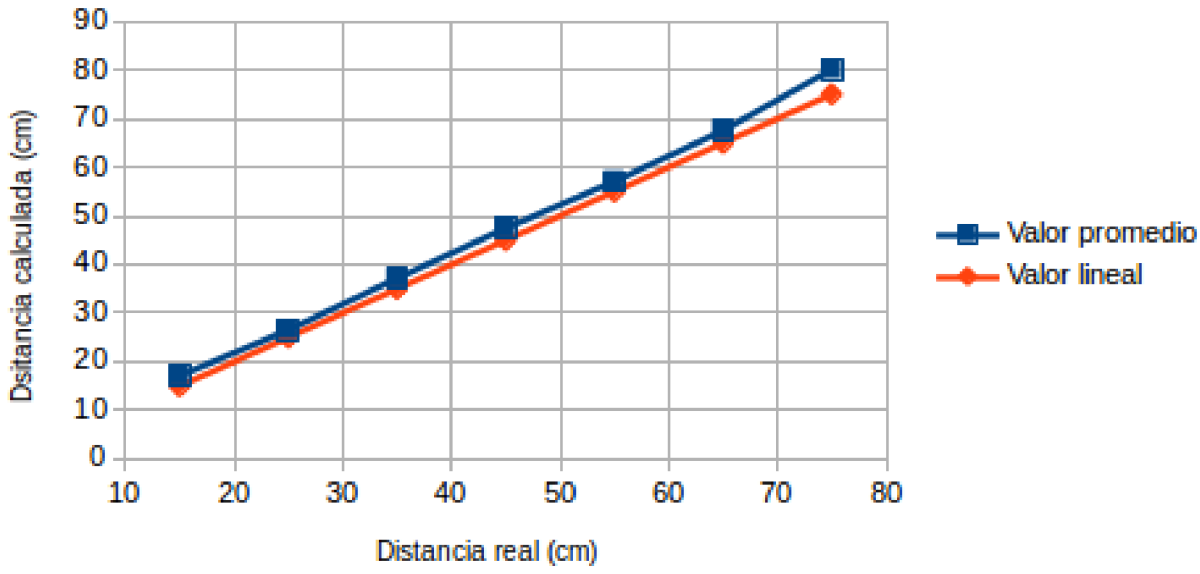


Figura 5.5: Gráfica comparativa de distancias

La distancia marcada como valor promedio se obtuvo a partir de 3 conjuntos de datos ya promediados, los cuales se obtuvieron a partir de 9 repeticiones sobre la misma distancia practica, teniendo que el conjunto total de muestras por distancia fue de 19. Mientras que el valor lineal es la recta que se hubiese obtenido si los valores coincidieran perfectamente con una relación 1 a 1. Se puede ver que los valores inician con un error casi constante, hasta llegar a los 65cm done comienza a aumentar. El conjunto de datos que obtuve de las pruebas realizadas es el siguiente:

Distancia medida	Promedio 1	Promedio 2	Promedio 3	Distancia promedio
15	17.25	17.62	16.58	17.15
25	26.42	26.54	26.61	26.5233333333
35	37.12	36.98	37.4	37.1666666667
45	47.59	47.23	47.63	47.4833333333
55	58.22	57.13	55.87	57.0733333333
65	68.7	69.57	64.39	67.5533333333
75	79.1	84.2	77.06	80.12

Cuadro 5.1: Tabla de datos obtenidos de distancias medida y calculada

El calculo de regresión lineal quedo de la siguiente forma, junto con la mencionada ecuación para calcular la distancia real:

$$b = 0{,}742200724$$

$$m = 0{,}962345678$$

$$y = mx + b \longrightarrow y = 0{,}961x - 0{,}7422$$

$$R^2 = 0{,}99868938$$

El valor del coeficiente de determinación de la regresión lineal es cercano a 1, por lo que la confiabilidad de la ecuación como método de linealización es muy bueno basándonos en la variación de los datos de la tabla, al menos hasta los 75cm de distancia, donde el error aumenta hasta los 5cm, de un intervalo inicial entre 2cm a 2.5cm máximo.

CAPÍTULO 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Pese a algunas dificultades encontradas al momento de implementar la visión estereoscópica en el hardware dedicado al robot, debidas principalmente a las versiones ya muy viejas del software original, después de realizar algunas adaptaciones a nivel computacional se logro que todo funcionase en armonía pudiendo así sustituir el sistema original del robot humanoide.

Gracias a que, como ya se menciono, no se requirió mucha modificación en cuestión del hardware del robot, la tesis se concentro principalmente en el desarrollo del sistema de visión estereoscópica, el cual se programo e implemento sin que el robot tuviese antecedente alguno de el dando resultados interesantes considerando las elecciones en cuestión de la cámara utilizada así como los métodos.

Dicha cámara aunque no posee una muy buena resolución se logro utilizar para conocer distancias con un nivel tolerable de error, lo que demostró que sin necesidad de cambiar sustancialmente el robot se podría realizar una mejora sobre la estimación de distancias, ya que anteriormente se dependía del mayor peso computacional generado al utilizar visión monocular. Sin embargo, aunque "minoru" sirvió para comprobar lo anterior sobre el robot humanoide, hay que mencionar que el rango de visión obtenido no es tan bueno como se esperaría, teniendo un rango de detección total de aproximadamente 45° (22.5° hacia la izquierda y derecha a partir del centro de las cámaras), por lo que en este rubro no se tiene una mejora sustancial, ya que el lente de la cámara original contaba con un ángulo de visión de 75°. Aun pese a esta desventaja, la cámara elegida como sustituto representa una opción mucho mas barata con la ventaja de poder utilizarse como sensor de distancia.

Pese a que efectivamente pudieron obtenerse representaciones tridimensionales de obje-

tos, la calidad de las nubes de puntos obtenidas no fueron tan buenas, sin mencionar que las perdidas por la resta del plano causa que posiblemente alguna geometría quede demasiado incompleta como para reconocerse por algún método posterior, o incluso que quede tan incompleta que el umbral de detección rechace la nube restante.

6.2. Trabajo futuro

Basado en los resultados y conclusiones reportadas, pude observar las siguientes posibles mejoras:

- Implementar un mejor algoritmo de detección, y aplicar reconocimiento.
  Hasta el desarrollo de esta tesis, solo se realizo la detección de objetos basados en la segmentación de la escena, sin embargo el siguiente paso seria realizar un reconocimiento de una base de objetos a partir del diseño de un clasificador, completando así un sistema de reconocimiento de patrones. Además, creo que es importante aplicar un método mas eficiente para la segmentación, ya que actualmente es una segmentación por distancia entre píxeles, pudiéndose extender al reconocimiento de otras características como bordes, primitivas, regiones, entre otras cosas. Creo que un punto de partida interesante podría ser la validación de lo mencionado por van Dijck, Korsten, y van der Heijden (1996) y Liu, Dai, y Xu (2010).

- Mejorar la calidad de la nube de puntos de los objetos detectados.
  Creo que esto se podría realizar combinando diferentes fuentes de información y técnicas, seria interesante probar si lo escrito por Diaz (1998) o Hsu, Lin, y Wu (2009) en realidad presenta alguna mejora sobre el análisis de la escena para la generación de las nubes de puntos. También seria óptimo generar e implementar algún método para tener una mejor descripción de la escena ya sea considerando de diferentes maneras las combinaciones de colores, texturas, formas, etc., obtenidas por la cámara.
  O bien también podría mejorarse la calidad implementando algún otro mecanismo de sensado de distancias, como podría ser un láser, sensor ultrasónico, etc., para posteriormente unir los datos sensoriales en un solo conjunto. El tener una mejor calidad en las distancias obtenidas deriva en una mayor probabilidad de detectar correctamente los objetos sin perder partes importantes de su geometría, ya que al tener una nube de puntos con características pobres se tiende a segmentar de forma incompleta.

- Disminuir el tiempo del sistema para la de detección de objetos.
  Una forma que parece optimizar bastante bien el procesamiento de la visión, es reducir previamente la nube de puntos adquirida, tal y como menciona Moenning y Dodgso (2003), aunque esto podría pensarse contrario al punto anterior, no es necesariamente así pues si se delimitan correctamente los objetos en la escena tomada por la cámara, se puede aplicar posteriormente algún algoritmo para suavizar las superficies detectadas

como el usado por Sumi et al. (1998), y/o disminuir la cantidad de puntos que las denotan sin arriesgar la integridad de la representación tri-dimensional. La reducción de datos de la nube de puntos la sugiero principalmente por la aplicación del algoritmo RANSAC en este trabajo, ya que al reducir la cantidad de puntos para buscar el plano dominante dentro de la escena, evidentemente reducirá el tiempo de computo.

- Mejorar el hardware pertinente a la visión estereoscópica.
  Desde el punto de vista del hardware, la cámara Minoru limita la visión estereoscópica a la posibilidad de hacer detección solo con la información de una imagen de resolución de 640x480 a 30 cuadros por segundo. En contraste la cámara "ZED" ya antes mencionada en la tabla 3.2, posee dentro de sus características reportadas un rango de reconocimiento de profundidad de hasta 20 metros de distancia desde la linea base, lo cual suena plausible considerando su resolución de 3840x1080 a 30 cuadros por segundo.
  (https://www.stereolabs.com/zed/docs/ZED_Datasheet_2016.pdf).

  Además de cambiar la cámara, se puede intentar una adaptación sobre el lente existente, ya que conociendo las medidas del lente de "Minoru", así como de su montura sobre el sensor de imagen, se puede ver que una lente con las siguientes características puede ser colocada en reemplazo de la original:

Descripción	*Valor*
Ángulo visual	180 grados
Longitud focal	1.8 milímetros
Longitud focal posterior	5.44 milímetros
Formato	1/3 pulgadas
Apertura	F2.0

Cuadro 6.1: Lente óptimo para remplazo en Minoru 3D

  Con esto se aumentaría el ángulo de visión, sin embargo una consecuencia inmediata de esto es que al aumentar el ángulo de visión de ambas cámaras, la disparidad disminuye, teniendo un rango menor de visión efectivamente estereoscópica. Por lo que se estaría intercambiando la distancia máxima por la posibilidad de una detección mas cercana pero amplia. La forma de compensar un campo de visión demasiado amplio para la cámara estereoscópica seria aumentar la linea base entre ambos lentes, sin embargo la cámara Minoru esta diseñada como una sola pieza.

Cabe mencionarse que aunque la cámara Minoru es bastante funcional a un precio asequible, para robots humanoides de un tamaño mas funcional (como el del humano), no seria un sensor de distancia muy útil pues la distancia entre la cabeza o montura de la cámara hacia el plano mas bajo que es el suelo aumentaría considerablemente, por lo que creo que un punto

importante a considerar es que eventualmente se deben trabajar algoritmos implementados en cámaras y sensores mas avanzados.

Apéndices

APÉNDICE A

Optimización del sistema operativo

- Eliminar tiempo de espera de la red:
  Con la siguiente linea, en una terminal de comandos, editar el archivo "failsafe.conf"...

```
sudo nano /etc/init/failsafe.conf
```

  comentar las 2 lineas que siguen a la frase...

```
"waiting for network configuration..."
```

  Al hacer esto se elimina la espera por la red al iniciar el S.O.

- Eliminar tiempo de espera del grub:
  Con las siguientes lineas, en una terminal de comandos, instalar "grub-customizer"...

```
sudo add-apt-repository ppa:danielrichter2007/grub-customizer
sudo apt-get update
sudo apt-get install grub-customize
```

  Ejecutar la aplicación, y en la pestaña "Configuración general" cambiar la opción

```
Boot default entry after
```

  A 0 segundos.

- Configurar auto-inicio de sesión:
  Con la siguiente linea, en una terminal de comandos, editar el archivo "custom.conf"...

```
sudo nano /etc/gdm/custom.conf
```

  Sobre-escribir todo el contenido, eliminar lo existente y agregar solamente las siguientes lineas:

```
[daemon]
TimedLoginEnable=true
TimedLogin=<nombre de usuario>
TimedLoginDelay=1
```

Archivo 1: /etc/gdm/custom.conf

- Predeterminar el idioma del teclado:
  Esto solo es necesario para el entorno de linux "LXDE", ya que en los demás entornos existe como tal un menú para configurar el teclado.

  1. Teclado en español:
     Con la siguiente linea, en una terminal de comandos, editar el archivo "autostart.conf"...

     ```
     sudo nano /etc/xdg/lxsession/Lubuntu/autostart
     ```

     Y agregar la siguiente linea hasta el final, de no existir simplemente agregar esta linea...

     ```
     @sudo setxkbmap -layout es
     ```

  2. Teclado en ingles:
     Editar el mismo archivo...

     ```
     sudo nano /etc/xdg/lxsession/Lubuntu/autostart
     ```

     Pero con la siguiente linea...

     ```
     @sudo setxkbmap -layout us
     ```

APÉNDICE B

Instalación de ROS para Nimbro-OP

Agregar la dirección del repositorio para la descarga de los paquetes de ROS

```
sudo sh -c 'echo ''deb http://packages.ros.org/ros/ubuntu precise main'' >
  /etc/apt/sources.list.d/ros-latest.list'
```

Actualiza los paquetes disponibles para instalación, y a su vez, procede a instalar ROS

```
sudo apt-get update && sudo apt-get install ros-groovy-desktop-full
```

Una vez terminada la descarga, se inicia la base de datos de ROS

```
sudo rosdep init
```

Ahora se procede a actualizar los paquetes dependientes de ROS

```
rosdep update
```

Así como las librerías puente para interacción con el robot

```
sudo apt-get install -y python-rosinstall ros-groovy-rqt-rviz ros-groovy-joy
  libqglviewer-qt4-dev libgsl0-dev git
```

Ahora se registra la instalación de ROS

```
export | grep ROS
```

ROS ya contiene un script para registrar un nuevo entorno de trabajo, se ejecuta de la siguiente forma

```
./opt/ros/groovy/setup.bash
```

Se crea el directorio que funcionara como espacio de trabajo, este deberá encontrarse dentro de la carpeta "home" del usuario

```
mkdir -p ~/catkin_ws/src
```

Una vez accediendo a este espacio de trabajo creado, se inicia la configuración del mismo

```
cd ~/catkin_ws/src
catkin_init_workspace
```

Ahora se registrara como espacio de trabajo con las variables de entorno

```
cd ~/catkin_ws/
catkin_make
source devel/setup.bash
echo $ROS_PACKAGE_PATH
```

Regresar a la carpeta del usuario, y agregar como ultimas lineas del archivo ".bashrc"

```
. /opt/ros/groovy/setup.bash
export NIMBRO_ROBOT_VARIANT=nimbro_op_hull
```

Archivo 2: .bashrc

Guardar el archivo modificado, y ahora escribir el siguiente comando para proceder a descargar el espacio de trabajo especifico para el robot "Nimbro"

```
git clone https://github.com/NimbRo/nimbro-op-ros
```

Una vez descargado, ejecutar el scrpit de instalación "setup.sh"

```
./nimbro-op-ros/src/nimbro/scripts/system_setup.sh
```

Por ultimo editar el archivo "/etc/hosts", pues se requiere para comunicarse con la tarjeta CM730, controladora de servomotores utilizada para el robot

```
127.0.0.1       localhost mycomp.local
```

Archivo 3: /etc/hosts

APÉNDICE C

Instalación de V4L2stereo

Debido a la librería utilizada para la visión estereoscópica, es necesario instalar la versión 2.3.1 de openCV, para eso, se instalan los pre-requisitos siguientes:

```
sudo apt-get install libcv4 libhighgui4 libcvaux4 libcv-dev libcvaux-dev
  libhighgui-dev libgstreamer-plugins-base0.10-dev libgst-dev
sudo apt-get install libglib2.0-dev
```

Posteriormente, descargar la versión de openCV mencionada Para instalarlo, extraer el archivo descargado, y escribir los siguientes comandos para instalarlo

```
mkdir build
cd build
cmake ..
sudo make && sudo make install
```

Ahora descargar la librería "libcvm57" de la pagina E instalarla con la siguiente linea de comando

```
sudo dpkg -i libcvm57.deb
```

Debido a que la versión de OpenCV instalada, busca sus propias versiones de las librerías instaladas anteriormente, es necesario realizar el vinculo entre las librerías buscadas y las existentes:

```
sudo ln -s libopencv_highgui.so.2.3.1 libhighgui.so
sudo ln -s libopencv_core.so.2.3.1 libcxcore.so
sudo ln -s libopencv_legacy.so.2.3.1 libcv.so
sudo ln -s libopencv_video.so.2.3.1 libcvaux.so
sudo ln -s libopencv_ml.so.2.3.1 libml.so
```

Ahora las librerías creadas se encontraran con los siguientes nombres, bajo el directorio "/usr/lib/"

/usr/lib/libhighgui.so
/usr/lib/libcxcore.so
/usr/lib/libcv.so
/usr/lib/libcvaux.so
/usr/lib/libopencv_calib3d.so.2.3.1

De la ultima pagina de internet mencionada, descargar las librerías para OpenCV de visión estereoscópica, contenidas en el paquete "V4L2stereo", posteriormente descomprimir el archivo "V4L2stereo" descargado, entrar a la carpeta, y modificar el archivo "Makefile" y en la linea correspondiente a "gstreamer", sobrescribir lo existente con lo siguiente:...

```
1    g++ -O3 -o v4l2stereo *.cpp calibration/*.cpp elas/*.cpp -I/usr/include/opencv
2      -I/usr/include/gstreamer-0.10 -L/usr/lib -L/usr/lib/gstreamer-0.10
3      `pkg-config --cflags --libs gstreamer-0.10` `pkg-config opencv --cflags
4      --libs` `pkg-config --cflags --libs glib-2.0` `pkg-config --cflags --libs
5      gstreamer-plugins-base-0.10` -msse3 -lgstapp-0.10 -Wall -pedantic -fopenmp
```

Archivo 4: Makefile (V4L2stereo)

APÉNDICE D

Instalación de VNCserver

Instalar el servidor para VNC

```
1       sudo apt-get install tightvncserver
```

Editar el siguiente archivo y sobre-escribirlo con la información siguiente

```
1       sudo nano /etc/init.d/tightvncserver
```

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          tightvncserver
# Required-Start:    $local_fs
# Required-Stop:     $local_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start/stop tightvncserver
### END INIT INFO
### Customize this entry
################# Set the USER ##################
export USER='pi'
### End customization required
eval cd ~$USER
case "$1" in
  start)
    su $USER -c '/usr/bin/tightvncserver :1'
    echo "Starting TightVNC server for $USER "
    ;;
  stop)
    pkill Xtightvnc
    echo "Tightvncserver stopped"
    ;;
  *)
    echo "Usage: /etc/init.d/tightvncserver {start|stop}"
    exit 1
    ;;
esac
exit 0
```

Archivo 5: /etc/init.d/tightvncserver

Ejecutar las siguientes lineas para cambiar los permisos de ejecución y que el servidor VNC pueda ejecutarse desde el inicio

```
sudo chown root:root /etc/init.d/tightvncserver
sudo chmod 755 /etc/init.d/tightvncserver
sudo update-rc.d tightvncserver defaults
```

Reiniciar el servicio de VNC

```
sudo /etc/init.d/tightvncserver start
sudo /etc/init.d/tightvncserver stop
```

Ahora en el lado del cliente, instalar el visor VNC

```
sudo apt-get install xtightvncviewer
```

Configurar ambas maquinas para conectarse en red estática, editar el siguiente archivo

```
sudo nano /etc/network/interfaces
```

Y sobre-escribir con las siguientes lineas

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 192.168.x.y
netmask 255.255.255.0
gateway 192.168.x.1
dns-nameservers 8.8.8.8 192.168.x.1
```

Archivo 6: /etc/network/interfaces

Por ultimo se editara el archivo para conexión VNC en el cliente, de tal forma que se conecte al entorno visual correcto, para esto, sobre-escribir el siguiente archivo

```
sudo nano ~/.vnc/xstartup
```

```
export XKL_XMODMAP_DISABLE=1
openbox &
/usr/bin/lxsession -s Lubuntu -e LXDE &
```

Archivo 7: /.vnc/xstartup

APÉNDICE E

Instalacion de SSH

Instalar el servidor SSH en la PC del robot

```
1 sudo apt-get install openssh-server
```

Copiar las configuraciones predeterminadas del SSH instalado

```
1 sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.factory-defaults
```

Cambiar los permisos de ejecucion de los archivos para que se puedan iniciar desde que la PC se enciende:

```
1 sudo chmod a-w /etc/ssh/sshd\_config.factory-defaults
```

Agregar proteccion del firewall para tiempo maximo de acceso

```
1 sudo ufw limit ssh
```

Editar el siguiente archivo

```
1 sudo nano /etc/ssh/sshd_config
```

Buscar la linea que dice lo siguiente

```
1 #Max login time
```

Y editar el valor al siguiente

```
1 LoginGraceTime 30
```

Reiniciar el servicio de SSH

```
1 sudo restart ssh
```

Ahora para agregar los permisos para montar USB via SSH o VNC, se necesita editar el siguiente archivo en la seccion especificada posteriormente:

```
1 sudo nano /usr/share/polkit-1/actions/org.freedesktop.udisks.policy
```

```
<action id=‘‘org.freedesktop.udisks.filesystem-mount’’>
<description>Mount a device</description>
<description xml:lang = ‘‘da’’>Monter en enhed</description>
<message>Authentication is required to mount the device</message>
<message xml:lang=‘‘da’’>Autorisering </message>
<defaults>
  <allow_any>no</allow_any>
  <allow_inactive>no</allow_inactive>
  <allow_active>yes</allow_active>
</defaults>
</action>
```

Archivo 8: /usr/share/polkit-1/actions/org.freedesktop.udisks.policy

Cambiar los 3 “no” en las ultimas lineas que se ven anteriormente por “yes”, al reiniciar el VNC ya se tendran permisos para acceder remotamente a los USB

APÉNDICE F

Instalación de pantalla táctil

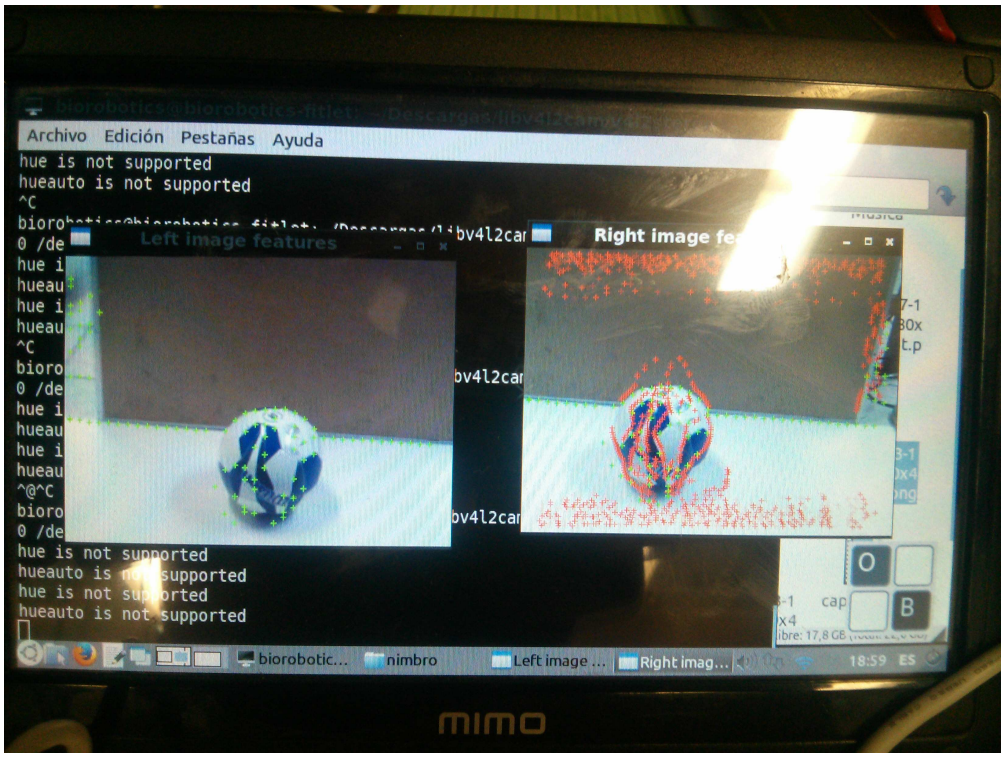


Figura F.1: Pantalla táctil en el robot mostrando resultados de la visión estereoscópica.

Respaldar el archivo de configuración "xorg.conf" para los periféricos conectados

```
1    sudo cp /etc/X11/xorg.conf ~
```

Instalar los drivers para pantalla táctil "displaylink" y posteriormente reinicia el sistema:

```
sudo apt-get install xserver-xorg-core xserver-xorg-video-displaylink
sudo reboot now
```

Conectar ahora la pantalla táctil, y teclear el siguiente comando

```
dmesg $|$ grep DisplayLink \newline
```

Copiar de la linea

```
udlfb: DisplayLink USB device...
```

la ubicación

```
/dev/fb*
```

En el caso de esta tesis, la ubicación es "/deb/fb0", en un archivo de texto, escribir la siguiente opción, que posteriormente se usara

```
Option      "fbdev" "/dev/fb0"
```

Aun con la pantalla táctil conectada, ejecutar el siguiente comando

```
sudo evtest
```

Esto mostrara una lista de dispositivos, seleccionar la opción que contenga "2ei" para posteriormente dar clic en la pantalla en las siguientes zonas:

Arriba izquierda
Y anotar los siguientes valores...
Valor mínimo ABS_X
Valor máximo ABS_Y
Arriba derecha
Y anotar los siguientes valores...
Valor máximo ABS_X
Valor mínimo ABS_Y

En el mismo archivo de texto previamente creado, agregar mínimos y máximos en la siguiente opción

```
Option "Calibration" "minX maxX minY maxY"
```

Donde minX, minY, maxX y maxY son los valores previamente obtenidos. En el caso especifico de esta tesis, fueron los siguientes:

Valor mínimo ABS_X: 808
Valor máximo ABS_Y: 31345
Valor máximo ABS_X: 31853
Valor mínimo ABS_Y: 1553

Finalmente sobre-escribir el siguiente archivo considerando los valores que se recopilaron previamente

```
sudo nano /etc/X11/xorg.conf
```

```
Section "Files"
  ModulePath      "/usr/lib/xorg/modules"
  ModulePath      "/usr/local/lib/xorg/modules"
  ModulePath      "/usr/local/lib/xorg/modules/drivers"
EndSection
Section "Device"
  Identifier      "Mimo"
  Driver          "displaylink"
  Option          "fbdev" "/dev/fb0"
EndSection
Section "Monitor"
  Identifier      "MimoMonitor"
EndSection
Section "Screen"
  Identifier      "MimoScreen"
  Device          "Mimo"
  Monitor         "MimoMonitor"
  DefaultDepth    16
EndSection
Section "InputClass"
  Identifier  "touchscreen"
  MatchProduct    "e2i Technology, Inc. USB Touchpanel"
  MatchDevicePath "/dev/input/event*"
  Option      "InvertY"       "true"
  Option "ReportingMode" "Raw"
  Option "SendCoreEvents" "On"
  Option "Calibration" "808 31853 1553 31345"
EndSection
```

Archivo 9: /etc/X11/xorg.conf

Para activar el clic derecho, editar el siguiente archivo y sobre-escribir con el texto posterior

```
sudo nano /usr/share/X11/xorg.conf.d/99-calibration.conf
```

```
Section "InputClass"
  Identifier  "touchscreen"
  Driver "evdev"
  MatchProduct    "e2i Technology, Inc. USB Touchpanel"
  Option "Calibration" "808 31853 1553 31345"
  Option "EmulateThirdButton" "1"
  Option "EmulateThirdButtonTimeout" "750"
  Option "EmulateThirdButtonMoveThreshold" "30"
EndSection
```

Archivo 10: /usr/share/X11/xorg.conf.d/99-calibration.conf

También se puede utilizar clic derecho con el teclado “on board”, instalarlo primero

```
sudo apt-get install onboard
```

Ejecutar el teclado y en “opciones“, desactivar ”auto escaneo del teclado“

Para que auto-inicie con el sistema, modificar el archivo siguiente

```
sudo nano /etc/xdg/autostartonboard-autostart.desktop
```

```
[Desktop Entry]
Name=Onboard
Exec=onboard
Type=Application
Terminal=false
```

Archivo 11: /etc/xdg/autostartonboard-autostart.desktop

Para regresar al monitor original re-configurar el archivo xorg con las lineas del ultimo bloque:

```
sudo nano /etc/X11/xorg.conf
```

```
Section "Screen"
  Identifier      "Default Screen"
  DefaultDepth    24
EndSection
Section "Module"
  Load    "glx"
EndSection
```

Archivo 12: /etc/X11/xorg.conf (monitor original)

APÉNDICE G

Instalación de la CM730

Ejecutar el script “robot_setup.sh” en la instalación de ROS previamente realizada

```
./catkin_ws/src/nimbro/scripts/robot_setup.sh
```

Conectar la tarjeta CM730 a la PC, y ejecutar

```
dmseg
```

Deberá aparecer un mensaje como el siguiente, señalando en la tercer linea el numero de serie de la CM730

```
[3003.946198] usb 1-1.1: Product: FT232R USB UART
[3003.946206] usb 1-1.1: Manufacturer: FTDI
[3003.946213] usb 1-1.1: SerialNumber: A4008alW
```

Después editar el siguiente archivo:

```
sudo nano /etc/udev/rules.d/90-cm730.rules
```

Al final del archivo, se debe modificar el 2o parametro de la linea siguiente

```
ENV{ID\SERIAL}=="FTDI\FT232R\USB\UART\XXXXXX"
```

En el caso especifico del robot utilizado, la linea quedo

```
ENV{ID\SERIAL}=="FTDI\FT232R\USB\UART\A4008alW"
```

Archivo 13: /etc/udev/rules.d/90-cm730.rules

APÉNDICE H

Adecuación para la mini-PC en Nimbro-OP

A continuación muestro las medidas del plano realizado para la pieza que fija la mintBox al robot Nimbro-OP

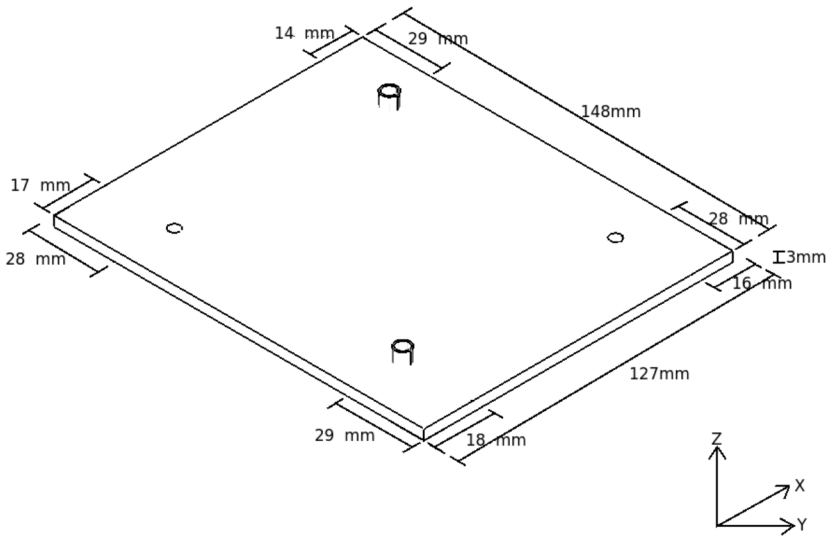


Figura H.1: CAD de la base para la mini-PC, vista axonométrica

El frente de la mini-PC y del robot corresponden con el eje X positivo.
Como se puede ver, hay 4 agujeros cuyas medidas en el plano solo coinciden en pares, esto es porque 2 son para la PC, y el otro par se utiliza para fijar esta base a la carcasa del robot Nimbro, a continuación en la vista lateral se puede ver que en efecto cada par sale en la respectiva dirección opuesta en Y.

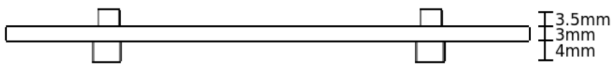
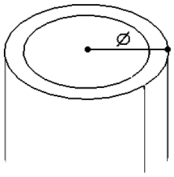


Figura H.2: CAD de la base para la mini-PC, vista lateral

Respecto a los agujeros para atornillar, estos fueron los radios empleados para evitar realizar perforaciones demasiado grandes o pequeñas



Radio	**interno [mm]**	**externo [mm]**
Separador de la mintBox PC	1.5	3
Separador de Nimbro-OP	2.5	5

Figura H.3 & Cuadro H.2: Separador y radios de la perforación.

Bibliografía

Arkin, Ronald C. (1998), *Behavior-Based Robotics*. Intelligent Robotics and Autonomous Agents, MIT Press, Cambridge.

Baguley, Greg (2009), *Stereo Tracking of Objects with respect to a Ground Plane*. Tesis, University of Canterbury, Christchurch, New Zealand.

Brown, M.Z., D. Burschka, y Hager (2003), "Advances in computational stereo." En *Trans. Pattern Analysis*, 993–1008, IEEE.

Castleman, Kenneth (1995), *Digital Image Processing*. Prentice Hall, U.S.A.

Diaz, Fernando A. (1998), *Un algoritmo de correspondencia entre imagenes estereo a multiples resoluciones*. Grado de ingeniero en computacion, UNAM, Facultad de Ingeniería.

Dynamixel (2010), "Especificaciones del servomotor mx-64 y mx-106." `http://support.robotis.com/en/product/dxl_main.htm`. Visitada: 2016-02-20.

Gonzalez, Rafael y Richard Woods (2002), *Digital Image Processing*, 2a edicion. Prentice Hall, New Jersey.

Hsu, Gee-Sern, Chyi-Yeu Lin, y Jia-Shan Wu (2009), "Real-time 3-d object recognition using scale invariant feature transform and stereo vision." En *4th International Conference on Autonomous Robots and Agents.*, 239–244, IEEE, Wellington.

Jähne, Bernd (2002), *Digital Image Processing*, 5a edicion. Springer, Germany.

Lim, Jae (1990), *Two-Dimensional Signal And Image Processing*. Prentice Hall, New Jersey.

Liu, Yebin, Qionghai Dai, y Wenli Xu (2010), "A point-cloud-based multiview stereo algorithm for free-viewpoint video." *IEEE Transactions on Visualization and Computer Graphics*, 16, 407–418.

Moenning, Carsten y Neil A. Dodgso (2003), "A new point cloud simplification algorithm." En *International conference on Visualization, image and image processing*, 1027–1033, IASTED, España.

Murphy, Robin R. (2000), *Introduction to AI Robotics*. Intelligent Robotics and Autonomous Agents, MIT Press, Cambridge.

Pratt, William (2001), *Digital Image Processing: PIKS Inside*, 3a edicion. Wiley, New York.

Sumi, Y., Y. Kawai, T. Yoshimi, y F. Tomita (1998), "Recognition of 3d free-form objects using segment-based stereo vision." En *Sixth International Conference on Computer Vision*, 668 – 674, IEEE.

Universität Bonn (2012), "Descripción del robot Nimbro-OP." `http://www.nimbro.net/OP/NimbRo-OP.html`. Visitada: 2016-02-20.

van Dijck, Harrie, Maarten Korsten, y Ferdi van der Heijden (1996), "Robust 3-dimensional recognition using stereo vision and geometric hashing." En *International Conference on Image Processing*, volumen 1, 329–332, IEEE, Laussane.

Vosselman, George (2009), "Advanced point cloud processing." En *Photogrammetrische Woche*, ITC, Enschede, the Netherlands.

Zhang, Zhengyou (1999), "A flexible new technique for camera calibration." Reporte teconico, Microsoft research.

Zuliani, Marco (2014), *RANSAC for Dummies*. University of California Santa Barbara.