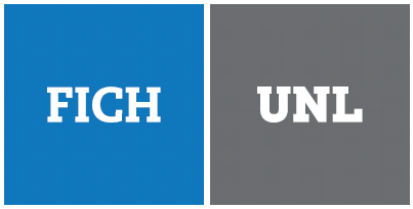


UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas

Desarrollo de un sistema de navegación basado en visión estéreo utilizando Lego® Mindstorms® NXT

Proyecto Final de Carrera:
Ingeniería Informática

Autor:
Alejandro Sartori

Director:
Dr. César Martínez

Co-Director:
Ing. Guido Sánchez

Santa Fe, Enero de 2014

Índice general

Resumen

La visión estereoscópica es una forma de sensado aplicable a diversos fines, entre ellos la robótica. Con ella se puede obtener medición sin contacto del entorno. Se consigue con la observación de la misma escena desde dos perspectivas diferentes, lo cual provee un medio para determinar la forma y su posición tridimensional.

En este proyecto se utiliza una cámara estereoscópica con la que se capturan pares de imágenes de la misma escena, con el fin de producir mapas de disparidad que sirvan al esquive de obstáculos por parte de un robot Lego® Mindstorms® NXT. Con tal capacidad, dicho robot debe ser capaz de dirigirse sin riesgo al destino de manera autónoma.

Dado que se extrae información métrica a partir de imágenes bidimensionales, primero se realiza un módulo para la calibración de cámaras, para corregir las distorsiones de las imágenes producidas por las características del dispositivo estereoscópico. Posteriormente se utiliza un método basado en áreas para la generación de los mapas de disparidad, los cuales son sujetos a interpretación para definir si existe un obstáculo frente al robot. Tales mapas se calculan a partir de los pares de imágenes luego de haber sido rectificadas, a los efectos de explotar las ventajas que brinda la geometría de este sistema, denominada geometría epipolar.

El manejo del robot se realiza según el paradigma reactivo dentro del marco de la arquitectura de control. La ejecución se lleva a cabo en una mini computadora denominada Raspberry Pi. Dicho sistema es capaz de responder en tiempo real a un entorno estático.

Los experimentos de odometría y trayectoria llevados a cabo muestran que el sistema es capaz de alcanzar un punto objetivo con precisión satisfactoria aún en la presencia de obstáculos dispuestos de formas diversas.

Capítulo 1

Introducción

Desde sus inicios los desarrollos de la visión por computadora han estado inspirados en el estudio del sistema visual humano. Representan una alternativa para la percepción del entorno para máquinas y permiten realizar mediciones sin contacto con el mismo[1]. Este tipo de desarrollos se está volviendo más popular en aplicaciones robóticas, donde se utilizan para el control basado en visión de robots móviles. Dicha visión puede dividirse en dos categorías de diseño: monocular y estéreo, cada uno con sus ventajas y desventajas, siendo la ventaja principal del diseño estéreo la posibilidad de generar información tridimensional.

Actualmente, es en estas aplicaciones robóticas móviles donde se está poniendo mayor énfasis, en gran medida debido a la necesidad de adaptación robótica a un entorno flexible, como por ejemplo en el caso del transporte de materiales entre cualesquiera de los puntos de una instalación fabril o en un lugar que resulte inhóspito para las personas, como ocurre luego de un desastre nuclear [1, 2].

La visión estéreo se consigue con la observación de la misma escena desde dos perspectivas diferentes, lo cual provee un medio para determinar la forma y su posición tridimensional. Esto es posible mediante la búsqueda de correspondencias de puntos en ambas vistas (es decir, capturar el mismo punto en las dos visualizaciones), lo que permite calcular diferencias de coordenadas del punto en cada una de ellas. Dicha diferencia se denomina disparidad y al conjunto total de diferencias mapa de disparidad, con el cual se pueden inferir distancias relativas. Para que esta búsqueda de correspondencias gane en precisión y eficiencia, se debe: por un lado obtener lentes libres de distorsión y por el otro hacer uso de la geometría asociada al sistema estéreo, a los fines de encontrar las relaciones espaciales entre ambos lentes. Estas relaciones permiten la rectificación de imágenes, lo cual posibilita que las búsquedas sean unidimensionales [3].

[1] www.tecnicaenlaboratorios.com/Nikon/Info_medicion_sin_contacto.htm

Para el movimiento del robot los mecanismos constan en general de un sistema de locomoción que puede basarse en tecnologías que se vienen utilizando desde hace siglos para el transporte, pudiendo ser ruedas, cadenas o hélices. Otras tecnologías buscan imitar a la naturaleza en la resolución de algún problema (biomímesis), como por ejemplo un sistema de locomoción que utilice piernas o que repte como una serpiente [2].

El sistema de navegación y esquive de obstáculos consiste en un algoritmo que se basa en la interpretación tridimensional de la escena visualizada. Debe contar con un módulo de navegación que sea capaz de tomar tal interpretación y responder en tiempo real, para cumplir con las restricciones temporales que permitan la interacción adecuada con el entorno.

1.1. Justificación

Existen diversas tecnologías para percibir el entorno, cada una de ellas con sus ventajas y desventajas. Se pueden usar sensores ópticos tales como fotorresistencias, fotodiodos o fototransistores. Los fototransistores pueden usarse como sensores infrarrojos de tipo On-Off, es decir pueden indicar solamente si el obstáculo está o no presente, para obtener información sobre distancia se debe recurrir a mecanismos más complejos [5].

Otro tipo de sensores muy utilizados son los de ultrasonido, los cuales hacen uso de las propiedades del sonido y del tiempo transcurrido entre emisión y recepción de la señal. Cuando se utiliza el mismo sensor para emisión y recepción tiene la desventaja de que aparece una “zona muerta”, esto es la distancia mínima a la que no se puede detectar obstáculos, la cual se determina por el tiempo que necesita el sensor para pasar del estado de emisión al de recepción. Otra desventaja que surge de la utilización de estos sensores es el denominado “eco falso”, provocado por el rebote de la señal en múltiples superficies antes de llegar al sensor, lo que implica que la estimación de la distancia al obstáculo sea mayor a la real [2].

Mediante el uso de un sensor de imágenes se podrían evitar las desventajas mencionadas. Además se contaría con la posibilidad de obtener otro tipo de información, no disponible con otros sensores, como por ejemplo color, reconocimiento de caracteres, objetos, etc. Por ello, este trabajo se realiza con la intención de brindar una solución al problema de navegación autónoma, basado en un sensor de imágenes [5].

Otra ventaja que presenta el uso de múltiples sensores de este tipo, es que permite obtener información tridimensional del entorno, información valiosa cuando la complejidad de la escena aumenta [6].

También debe notarse que desarrollar un sistema basado en visión artificial representa una evolución para la robótica, ya que dicho sistema está inspirado en el sentido más utilizado por la mayoría de los animales, la visión. En este trabajo se utilizará un

sensor de tipo estereoscópico. Este tipo de sensor está diseñado para la imitación de la forma en que la naturaleza resuelve un problema de diseño, ya que se está creando un sistema basado en visión condicionada por la disposición frontal de los ojos, como tenemos los humanos.

La utilización de sistemas de visión estereoscópicos en robótica móvil mejora en gran medida el desempeño gracias al uso de información tridimensional, particularmente útil en situaciones que involucran el reconocimiento de escenas no familiares. Considerando un sistema de navegación que utiliza visión estéreo, deben resolverse tres problemas: el primero es cómo obtener la información tridimensional de la escena con lo que ello implica (contrarrestar la distorsión inherente en toda cámara y conocer la disposición física del dispositivo de visión), el segundo es tener un mecanismo que pueda interpretar la información provista por tal sensor y el tercero es la estimación del movimiento del robot mediante un módulo de navegación. Dicho módulo de navegación debe lograr resolver el camino a seguir en tiempo real, para permitir que dicha navegación se realice sin riesgos para el entorno del robot y para sí mismo. Tal restricción temporal implica que el sistema sea predecible en cuanto al tiempo que demora en brindar una respuesta. El algoritmo de visión estéreo puede implementarse mediante hardware específico o en una computadora [1, 7, 8].

Este trabajo tiene como finalidad dotar a un robot de autonomía. Con ella, lo que se busca es reducir la intervención humana al mínimo, lo que representa un paso más para acercar la robótica a nuestra vida diaria. Hasta ahora los avances conseguidos contribuyeron a los primeros intentos de utilizar robots móviles en aplicaciones de la vida real, como asistentes personales para personas con discapacidades o ancianas por ejemplo, mascotas robóticas, entre otros. Un sistema de este tipo se puede aplicar en diversas situaciones, por ejemplo para un dispositivo explorador donde se requiera navegación autónoma sin riesgos para el mismo, con la ventaja agregada de que utiliza solamente un sensor para lograr el cometido, lo que deviene en menores costos para su implementación.

Cuando las innovaciones son como en este caso inspiradas en la naturaleza se acuñan bajo los términos “bioimitación” o “biomímesis”[2]. El argumento principal para apelar a estas soluciones es que en millones de años de evolución de la vida en la Tierra, la naturaleza seguramente ya encontró soluciones de diseño óptimas para enfrentar los desafíos a los que se enfrentan los humanos. Estas soluciones además conducen a diseños eficientes, prácticos y sostenibles desde un punto de vista ingenieril, y tienen además el potencial para descubrir nuevas tecnologías, materiales y procesos.

Se establece como interés central de este trabajo realizar una aplicación en el marco de la biomímesis. Dicha aplicación es posible a través del uso de un sistema de visión estéreo calibrado y rectificado como herramienta que genera información útil para un algoritmo que posibilita evitar obstáculos.

[2]http://www.athanor.es/reportajes/bioimitacion

1.2. Objetivos

General

Desarrollar un sistema de navegación basado en visión estéreo.

Específicos

- Desarrollar el módulo para calibración automática de cámaras utilizando un patrón conocido.
- Desarrollar el algoritmo para procesar imágenes ya calibradas, con las que se pueda generar los mapas de disparidades.
- Diseñar y desarrollar el módulo para navegación basado en procesamiento estereoscópico de imágenes.
- Implementar el módulo desarrollado en una mini computadora y dotar a un robot de navegación autónoma en tiempo real.
- Evaluar y analizar el desempeño del sistema desarrollado.

1.3. Alcances

Se desarrollará un software capaz de realizar la calibración individual de cada uno de los lentes de un dispositivo estereoscópico, a los efectos de eliminar las distorsiones que provoquen. La información de esta calibración se utilizará para realizar la calibración estéreo y rectificación de las imágenes que se capturen. El tipo de calibración será fotogramétrica mediante la utilización de un objeto plano. Como objeto de calibración se utilizará un tablero de ajedrez.

Por cada par de imágenes tomado desde el dispositivo ya calibrado y rectificado, se realizará una búsqueda de correspondencias de puntos para poder calcular diferencias de coordenadas y generar así un mapa de disparidad. Cada mapa de disparidad será interpretado a los efectos de detectar y esquivar obstáculos.

El Lego® Mindstorms® deberá aproximarse a un punto relativo al comienzo del recorrido. El desplazamiento se llevará a cabo en una escena estática con terreno plano y condiciones de iluminación controladas.

La computadora Raspberry Pi donde se instala el sistema debe estar conectada a la energía eléctrica.

Capítulo 2

Marco Teórico

Esta sección expone nociones básicas acerca de las transformaciones que ocurren cuando se capturan imágenes mediante una cámara. Asimismo se identifican algunos de los tipos de distorsión que introduce el uso de la misma y los modelos que los describen. También se muestran conceptos asociados a la visión estéreo, como obtenerla, utilizarla y su geometría asociada denominada Geometría Epipolar.

2.1. Transformaciones proyectivas y de cámara

La geometría proyectiva trata aquellas propiedades que se conservan bajo proyecciones. Tiene aplicaciones en visión artificial, funcionamiento de cámaras, etc. Es la geometría asociada al modo en que el ojo humano percibe el mundo. Por ello, se realiza una introducción breve a los conceptos generales de la misma.

2.1.1. De la geometría Euclideana a la proyectiva

La geometría Euclideana sirve para describir el mundo tridimensional en el que vivimos. En ella, los costados de los objetos tienen longitudes, líneas intersectándose determinan ángulos, y dos líneas son paralelas si se encuentran en el mismo plano y nunca se tocan.

Esas propiedades no cambian cuando se aplican transformaciones Euclideanas (rotación y traslación). Sin embargo, al estudiar el proceso de formación de imágenes en una cámara, se observa que longitudes y ángulos no se preservan, y que las líneas paralelas parecen intersectarse, por lo cual la geometría Euclideana resulta insuficiente. Por esta razón se mejora agregando estos puntos en el infinito donde se encuentran las

líneas paralelas, denominándolos “puntos ideales” [3, 9].

Agregando estos puntos en el infinito, el espacio Euclideano conocido se transforma en el espacio proyectivo. Este espacio incluye las propiedades del espacio Euclideano, como distancias, ángulos, puntos, líneas e incidencia. El espacio proyectivo, es una extensión del espacio Euclideano en el cual dos líneas siempre se encuentran en un punto, el punto en el infinito [9].

La geometría proyectiva modela bien el proceso de formación de imágenes en las cámaras porque permite un número mayor de tipos de transformaciones además de rotaciones y traslaciones, por ejemplo proyecciones perspectivas [10].

2.1.2. Representación de coordenadas

Un punto en el espacio bidimensional Euclideano se representa por un par ordenado de números reales (x, y). El mismo punto se puede representar agregando una tercer coordenada igual a 1 al final, dando origen a la tripleta $(x, y, 1)$. Tal tripleta se denomina coordenadas homogéneas del punto, y para volver a la representación original se debe dividir por la última coordenada. Por ello, las coordenadas $(x, y, 1)$ y $(2x, 2y, 2)$ representan al mismo punto, y en general (kx, ky, k) también lo hace, para todo valor k distinto de cero. Es mediante estas coordenadas homogéneas, que el espacio Euclideano $\mathbb{R}^n$ se extiende al espacio proyectivo $\mathbb{P}^n$.

Cuando la última coordenada es 0, se obtiene el punto $(x/0, y/0)$, lo que representa el infinito. Es así como surgen los puntos en el infinito, los que se representan por coordenadas homogéneas con la última coordenada igual a cero [3].

2.1.3. Cambio de coordenadas

Se puede considerar un cambio de coordenadas para el espacio Euclideano, ya sea cuando los ejes se trasladan y rotan a una posición diferente, o cuando el espacio mismo es trasladado y rotado a otra ubicación. La operación resultante se denomina transformación Euclideana. Esta transformación lineal del espacio Euclideano $\mathbb{R}^n$ se representa por una multiplicación matricial aplicada a las coordenadas del punto.

De la misma manera, una transformación proyectiva del espacio $\mathbb{P}^n$ es un mapeo de las coordenadas homogéneas que representan un punto (un vector de (n + 1) componentes), en el cual el vector coordenado se multiplica por una matriz no singular. Así, una transformación proyectiva del espacio proyectivo $\mathbb{P}^n$ se representa mediante una transformación lineal de coordenadas homogéneas según [3]

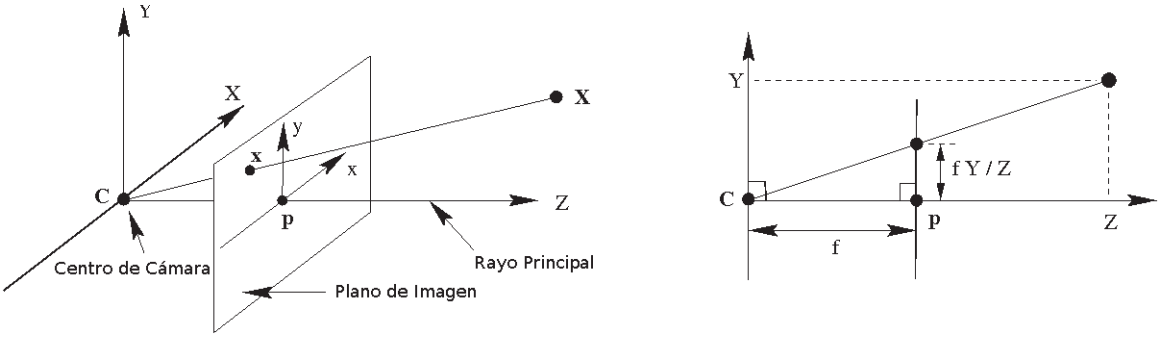


Figura 2.1: Geometría de cámara pinhole. **C** es el centro de cámara y **p** el punto principal. El centro de cámara está ubicado en el origen de coordenadas. Imagen modificada de [3].

$$X^{'} = H_{(n+1)\times(n+1)}X. \tag{2.1}$$

En problemas de visión por computadoras, el espacio proyectivo se utiliza como una manera conveniente de representar el espacio real 3D, extendiéndolo al espacio proyectivo tridimensional.

2.1.4. Proyecciones de cámara - Modelo Pinhole

El descenso del espacio tridimensional a una imagen bidimensional se modela mediante proyección central, en la cual se dibuja un rayo desde un punto en el espacio del mundo 3D hasta un punto fijo en el espacio, denominado centro de proyección. El rayo intersecta un plano específico en el espacio que se elige como el plano de la imagen. Esa intersección representa la imagen del punto. Si la estructura 3D se encuentra en un plano entonces no hay descenso en dimensión.

Este modelo concuerda con el modelo de cámara pinhole. En éste, se define como centro de proyección el origen de un sistema de coordenadas Euclideano. Dicha proyección se materializa en el plano $Z = f$, el plano de imagen. Bajo este modelo, un punto en el espacio Euclideano con coordenadas $\boldsymbol{X} = (X, Y, Z)^T$ se mapea al punto sobre el plano de imagen por el que pasa una rayo uniendo dicho punto en el espacio $\boldsymbol{X}$ con el centro de proyección, como se observa en la Figura 2.1 [3, 11].

Mediante triángulos semejantes[1], el punto $(X, Y, Z)^T$ se mapea al punto $(fX/Z, fY/Z, f)^T$ sobre el plano de imagen. Ignorando la coordenada de imagen final, se ve que

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T \tag{2.2}$$

[1]Dos triángulos son semejantes si tienen los lados proporcionales.

describe la proyección central desde coordenadas de espacio a coordenadas de imagen. Este es un mapeo de $\mathbb{R}^3$ a $\mathbb{R}^2$.

El centro de proyección se denomina centro de cámara. La línea desde el centro de cámara perpendicular al plano de imagen es el rayo principal, y el punto donde el rayo principal se encuentra al plano de imagen es el punto principal. El plano a través del centro de cámara paralelo al plano de imagen es el plano principal de la cámara [3].

Proyección central utilizando coordenadas homogéneas

Representando los puntos de imagen y del espacio mediante vectores homogéneos, la proyección central se expresa como un mapeo lineal entre coordenadas homogéneas. En particular, $(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T$ puede escribirse en términos de multiplicación de matrices según

$$\begin{vmatrix} X \\ Y \\ Z \\ 1 \end{vmatrix} \mapsto \begin{vmatrix} fX \\ fY \\ Z \end{vmatrix} = \begin{vmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{vmatrix} \begin{vmatrix} X \\ Y \\ Z \\ 1 \end{vmatrix}. \tag{2.3}$$

Se designa con X al punto del espacio representado por el vector homogéneo $(X, Y, Z, 1)^T$, x el punto de imagen representado por un vector homogéneo de tres elementos, y P la matriz de proyección homogénea de la cámara. De manera compacta se escribe

$$x = PX.$$

Matriz intrínseca

Para llevar el modelo pinhole a coordenadas de píxel, se deben introducir los parámetros c_x y c_y, que representan los desplazamientos en el sensor CCD[2] con respecto al eje óptico, ya que éste no necesariamente cae en el centro del mismo. Esto determina que un punto Q en el espacio con coordenadas (X, Y, Z) se proyecte en el píxel ($x_{pantalla}$,$y_{pantalla}$), cuyas coordenadas se definen por

$$x_{pantalla} = f_x \frac{X}{Z} + c_x, y_{pantalla} = f_y \frac{Y}{Z} + c_y. \tag{2.4}$$

Los parámetros f_x y f_y representan las longitudes focales. Se necesitan dos parámetros diferentes debido a que los píxeles individuales de una cámara económica a menudo son rectangulares en vez de cuadrados. Dicha longitud focal f_x es el producto

[2]El sensor CCD consiste en una grilla de fotodetectores que convierten la luz en información digital.

de la longitud focal física F y el tamaño s_x de cada uno de los elementos individuales del dispositivo. Debe observarse que las unidades de s_x están en píxeles por milímetro mientras la distancia focal física está en milímetros, lo que implica que f_x está en píxeles. De esta manera se calcula $fx = Fsx$. Las mismas afirmaciones son válidas para f_y y s_y.

Estos parámetros pueden organizarse en una matriz que representa la proyección de puntos del sistema de coordenadas del espacio al plano de imagen. Dicha matriz es la matriz intrínseca M. De esta manera el punto q en la pantalla puede expresarse como

$$q = MQ, q = \begin{vmatrix} x \\ y \\ w \end{vmatrix}, M = \begin{vmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{vmatrix}, Q = \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \tag{2.5}$$

2.2. Distorsiones del lente

Con el modelo pinhole ideal, se tiene un modelo útil para la geometría de la visión tridimensional. Sin embargo en la realidad toda cámara utiliza un lente, y este lente es responsable de introducir distorsiones que se deben corregir. La calibración de lentes en visión de computadoras para la navegación autónoma mediante visión estéreo es un tema de interés, ya que se requieren observaciones posicionales precisas [12].

Tipos de distorsión

Aquí se describen los dos tipos principales de distorsión y como modelarlos. Las distorsiones radiales surgen como consecuencia de la forma de la lente, mientras que las distorsiones tangenciales debido a defectos en el ensamblaje de la cámara.

Distorsión Radial

La distorsión radial causa que líneas rectas en el espacio se rendericen como líneas curvas en el sensor CCD. Este tipo de distorsión produce las deformaciones más severas y cuando ocurre, los lentes tienden a introducir distorsiones cercanas a los bordes del sensor. Este fenómeno es el que causa el efecto “ojo de pez” (Figura 2.3).

En general, produce que los rayos más alejados del centro del lente se tuerzan más que los cercanos. Provoca que la magnificación transversal[3] se comporte como una función de la distancia **r** al centro óptico, haciendo que sea 0 en el centro del sensor y se vaya incrementando hacia la periferia. Es el tipo de distorsión que genera que el centro de la imagen aparezca inflado (distorsión de barril) o pinchado (distorsión de cojín). El

[3]El aumento (o magnificación) transversal (o lateral) es el cociente entre las alturas (respecto al eje) de un punto imagen (h’) y de su punto conjugado (h): m = h’/h.

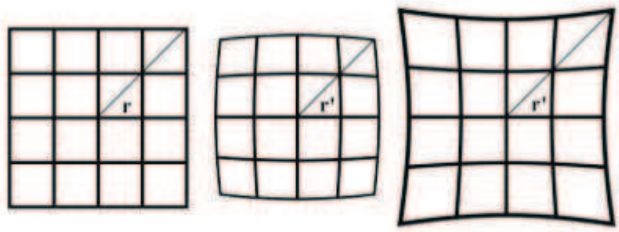


Figura 2.2: Distorsión radial. Izquierda: grilla sin distorsión. Centro: grilla con distorsión de barril. Derecha: grilla con distorsión de cojín. Imagen tomada de [14].

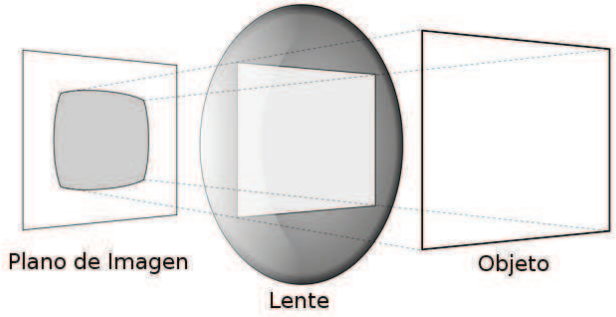


Figura 2.3: Proyección con distorsión radial. Ejemplo de distorsión de tipo barril que sufre un objeto cuadrado. Imagen modificada de [16].

sistema óptico sufre de distorsión de cojín cuando la magnificación transversal se incrementa con la distancia al centro óptico. En contraposición la distorsión de barril ocurre cuando la magnificación transversal decrece con tal distancia (Figura 2.2) [13, 14].

En general, ésta se modela por los primeros términos de la serie de Taylor alrededor de $r = 0$. Para cámaras baratas, se pueden caracterizar con los primeros dos términos, que se denominan convencionalmente k_1 y k_2. Para cámaras altamente distorsionadas como las de lente ojo de pez se agrega un término adicional k_3. La ubicación radial de un punto sobre el sensor se corrige según las ecuaciones [15]

$$x_{corregido} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), y_{corregido} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6). \quad (2.6)$$

donde (x, y) es la ubicación original (en el sensor) del punto distorsionado y $(x_{corregido}, y_{corregido})$ es la ubicación corregida.

Distorsión tangencial

El segundo tipo de distorsión más común es la tangencial. Ocurre debido a defectos en el proceso de fabricación que resultan en un lente no alineado exactamente al plano

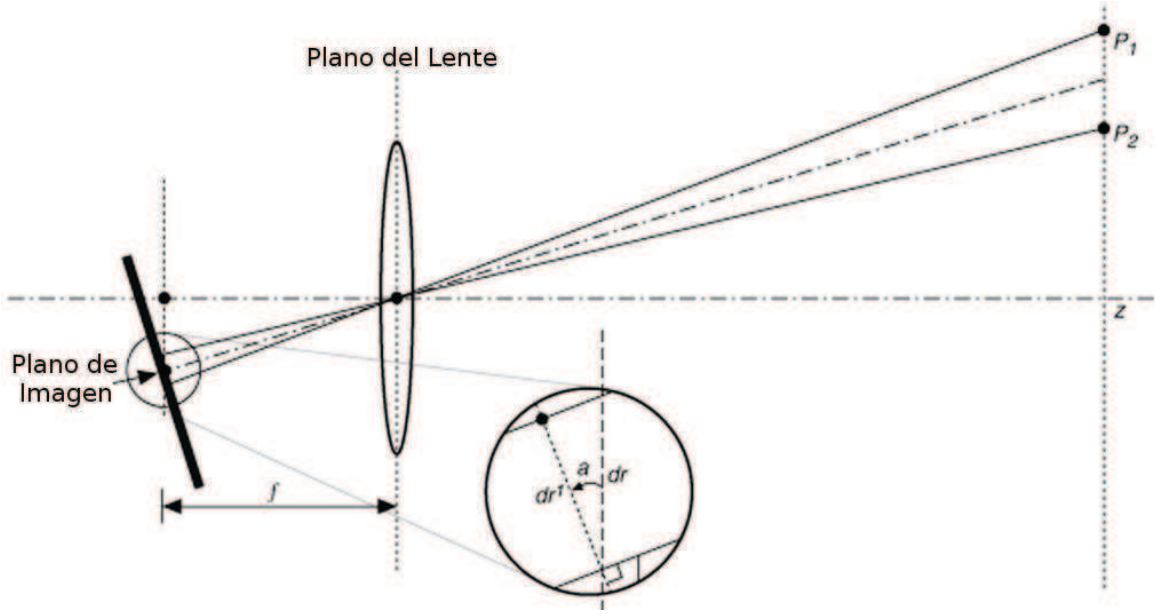


Figura 2.4: Distorsión tangencial. Imagen modificada de [17].

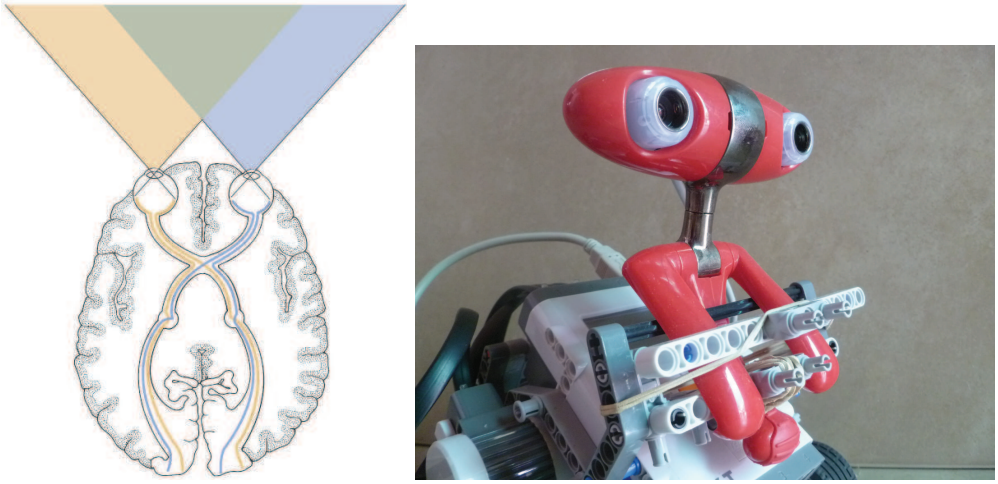
de imagen (Figura 2.4). La distorsión tangencial es caracterizada por dos parámetros adicionales, p_1 y p_2 según [15]

$$x_{corregido} = x + [2p_1xy + p_2(r^2 + 2x^2)], y_{corregido} = y + [p_1(r^2 + 2y^2) + 2p_2xy]. \quad (2.7)$$

Así, hay en total cinco coeficientes de distorsión. Dado que son comunes, las rutinas que trabajan con los mismos en general los almacenan en un vector de 5 x 1 [11].

2.3. Visión Estéreo

La capacidad de formación de imágenes estéreo que brindan nuestros ojos resulta familiar, el punto es hasta dónde se la puede emular a través de un sistema computacional. Dos cámaras que observan la misma escena desde diferentes puntos de vista generan un medio para determinar la forma tridimensional y la posición, brindando un mecanismo directo para la percepción de máquinas, porque permite técnicas de medición directas de profundidad. La Figura 2.5 muestra una comparación entre el esquema biológico del sistema visual humano (izquierda), donde en la parte superior se pueden ver los dos ojos y sus campos de recepción; mientras que a la derecha se muestra un dispositivo estéreo para visión computacional, una webcam dotada de 2 cámaras. Una vez que se tienen las imágenes estéreo, recuperar la profundidad mediante triangulación de puntos correspondientes resulta sencillo [18].



(a) Visión Humana (b) Webcam Estéreo

Figura 2.5: Visión estéreo.

Tales puntos correspondientes son proyecciones del mismo punto del espacio tridimensional hacia los diferentes planos de imagen. Con estas correspondencias y sabiendo la posición de los centros de proyección, la longitud focal, la orientación de los ejes ópticos, y el intervalo de muestreo de cada cámara, la profundidad se puede establecer mediante triangulación. Dicha búsqueda puede ser computacionalmente costosa, para evitarlo, se utiliza el conocimiento de la geometría del sistema, denominada geometría epipolar. Esta práctica ajusta el espacio de búsqueda tanto como es posible [11].

El primer paso involucrado en visión estéreo mediante el uso de dos cámaras consiste en eliminar matemáticamente las distorsiones radial y tangencial del lente de cada una, para obtener una imagen libre de distorsión. Luego se deben ajustar los ángulos y distancias entre cámaras, proceso denominado rectificación. La salida de este paso son imágenes de filas alineadas [4] y rectificadas. Posteriormente se deben buscar puntos que se correspondan en ambas vistas. La salida de este paso es el mapa de disparidad. Tales disparidades son las diferencias entre la coordenada horizontal x^l de cada punto izquierdo y la coordenada horizontal x^r de su punto correspondiente derecho [5], es decir x^l - x^r.

La correspondencia estéreo se puede resumir en términos de encontrar correspondencias reales, es decir características en la imagen generadas por la misma entidad física en el espacio. Dichas correspondencias reales satisfacen en general algunas restricciones [19].

[4] Filas alineadas significa que ambos planos de imagen (o planos proyectivos) son coplanares y que las filas de las imágenes están exactamente alineadas (en la misma dirección, teniendo las mismas coordenadas y).

[5] En este caso, dado el tipo de sensor que se utiliza, sólo se calcula disparidad en x. Mediante el uso de un sensor trifocal se podría calcular disparidades en x y y, con el consiguiente aumento en esfuerzo computacional.

- Restricción epipolar: dada una característica en una imagen, su par correspondiente estará a lo largo de una línea en la segunda imagen, denominada línea epipolar.
- Similaridad: puntos correspondientes tienen atributos o propiedades locales similares.
- Orden: la posición relativa de dos características se preserva en ambas vistas.
- Unicidad: cada punto en una imagen debe corresponderse con sólo un punto en la otra.

Modelo para la triangulación

Con las correspondencias obtenidas, ya se tiene un medio para triangular la profundidad. Para lograr tal cometido el dispositivo debe cumplir con ciertas restricciones [11]:

1. El dispositivo estéreo debe estar perfectamente calibrado, alineado y medido como se muestra en la Figura 2.6, con ambos planos de imagen coplanares, ejes ópticos paralelos separados por una distancia conocida T, y con la longitud focal izquierda f_l igual a la longitud focal derecha f_r.
2. Ambos puntos principales c_x^{left} y c_x^{right} se han calibrado para tener las mismas coordenadas de pixel en sus respectivas imágenes izquierda y derecha[6].
3. Las imágenes son de filas alineadas (cada fila de píxeles de una cámara se alinea exactamente con la correspondiente fila en la otra). Esta disposición de cámaras se denomina paralelo frontal.
4. Se puede encontrar un punto P del espacio físico en los puntos de vista izquierdo y derecho en p_l y p_r, de los cuales se tienen las coordenadas horizontales respectivas x^l y x^r.

Tomando x^l y x^r como las coordenadas horizontales de los puntos en cada cámara, se define como disparidad a la resta: $d = x^l - x^r$. El modelo resultante se muestra en la Figura 2.6, con el cual se puede derivar la profundidad Z usando triángulos semejantes. Con dicho modelo se realiza la reproyección mediante triangulación, la cual establece la relación entre profundidad y disparidad según

$$\frac{T - (x^l - x^r)}{Z - f} = \frac{T}{Z} \Longrightarrow Z = \frac{fT}{x^l - x^r}. \tag{2.8}$$

[6]No deben confundirse estos puntos principales con el centro de la imagen en píxeles. Un punto principal es donde el rayo principal intersecta el plano de imagen. Esta intersección depende del eje óptico del lente. Como existen distorsiones, el plano de imagen rara vez está alineado exactamente con el lente, por lo tanto el centro del sensor casi nunca coincide con el punto principal.

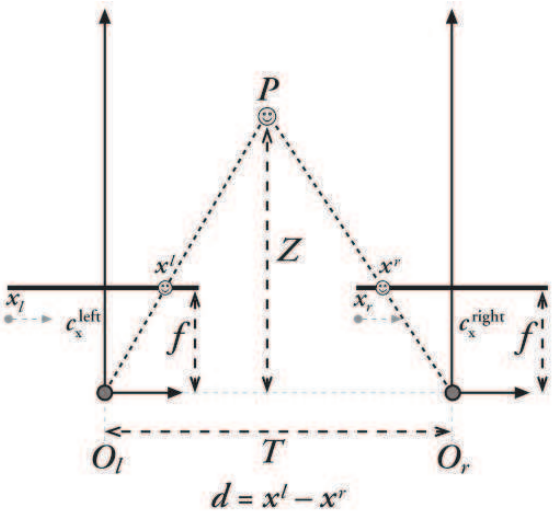


Figura 2.6: Modelo estándar para la triangulación. Imagen tomada de [11].

Dicha profundidad Z se obtiene cuando se conocen las coordenadas físicas de la cámara, de otra manera se calcula sólo hasta un factor de escala. La relación entre la disparidad y la profundidad es no lineal e inversamente proporcional. Cuando la disparidad es cercana a 0, pequeñas diferencias en disparidad generan grandes cambios en profundidad. Cuando la disparidad es grande, pequeñas diferencias en disparidad no afectan demasiado la profundidad, provocando que los sistemas de visión estéreo tengan gran resolución de profundidad a corta distancia, como se muestra en la Figura 2.7 [11].

La Figura 2.8 muestra los sistemas de coordenadas de un modelo para la visión estéreo. Es un sistema de mano derecha. En éste, los dispositivos tienen origen de imagen en la esquina superior izquierda, y los píxeles se denotan por (x_l,y_l) y (x_r,y_r) respectivamente. El centro de proyección está en O_l y O_r con rayos principales intersectando el plano de imagen en el punto principal (c_x,c_y).

Después de realizar la rectificación matemática, las cámaras son de filas alineadas (coplanares y horizontalmente alineadas), desplazadas una de otra por T, y con la misma distancia focal f. Se necesita lograr esta disposición para resolver distancias. Por esta razón se busca la manera de mapear una configuración de cámara de mundo real a esta configuración paralelo frontal, como se grafica en la Figura 2.8. En ésta, las coordenadas de píxel son relativas a la esquina superior izquierda de la imagen y las coordenadas de cámara, relativas al centro de proyección de la cámara izquierda.

Físicamente, las cámaras deben estar alineadas de esta manera al menos aproximadamente, para hacer las transformaciones matemáticas más tratables. De lo contrario, la alineación matemática resultante puede producir imágenes distorsionadas y así reducir el área de solapamiento estéreo de las imágenes resultantes[7]. Para resultados óptimos,

[7]La excepción a esta afirmación es cuando se requiere mayor resolución a corta distancia, para ello

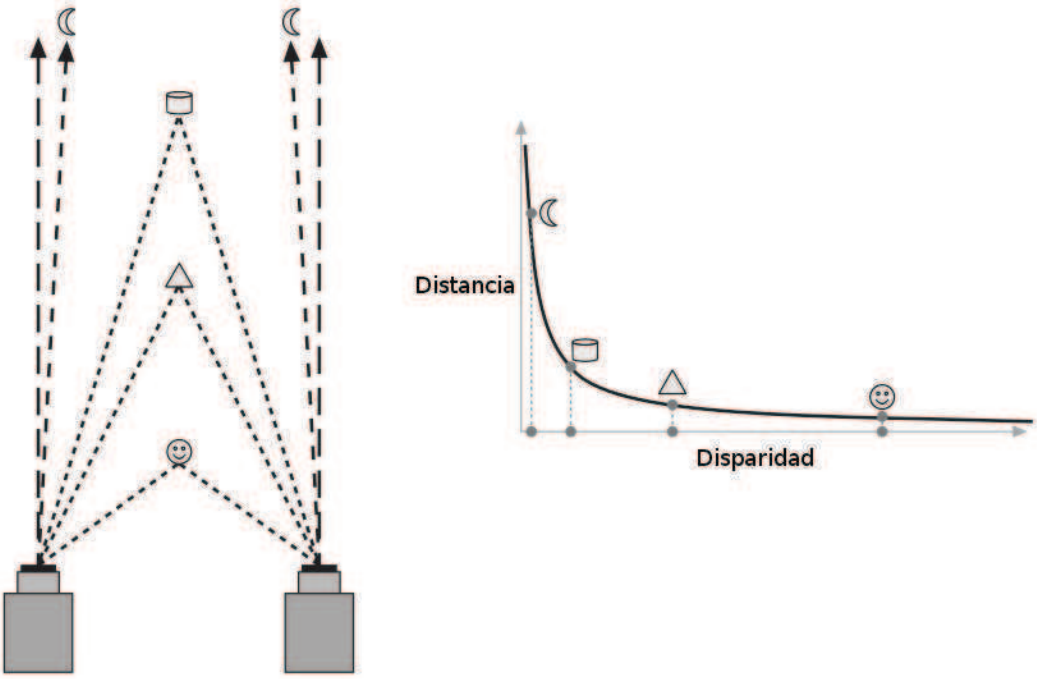


Figura 2.7: La profundidad y la disparidad están inversamente relacionadas, la medición de profundidad de mayor resolución se restringe a objetos cercanos. Imagen modificada de [11].

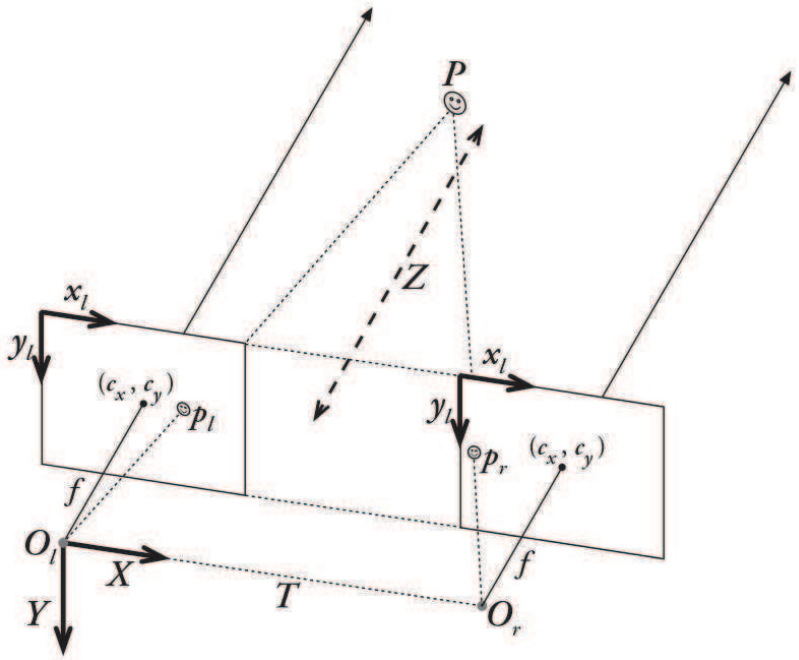


Figura 2.8: Sistema de coordenadas del modelo estéreo. Imagen modificada de [11].

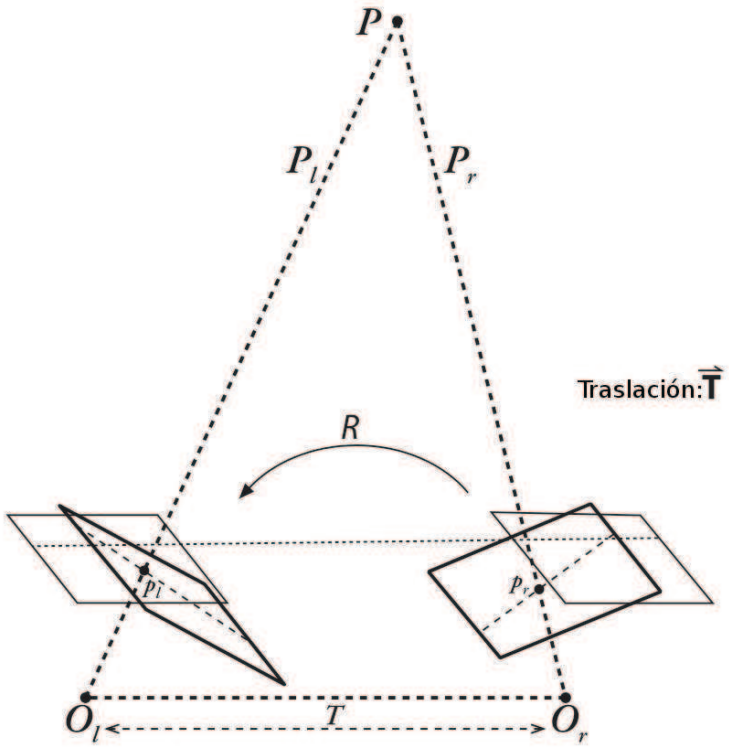


Figura 2.9: Alineación matemática de cámaras. Imagen modificada de [11].

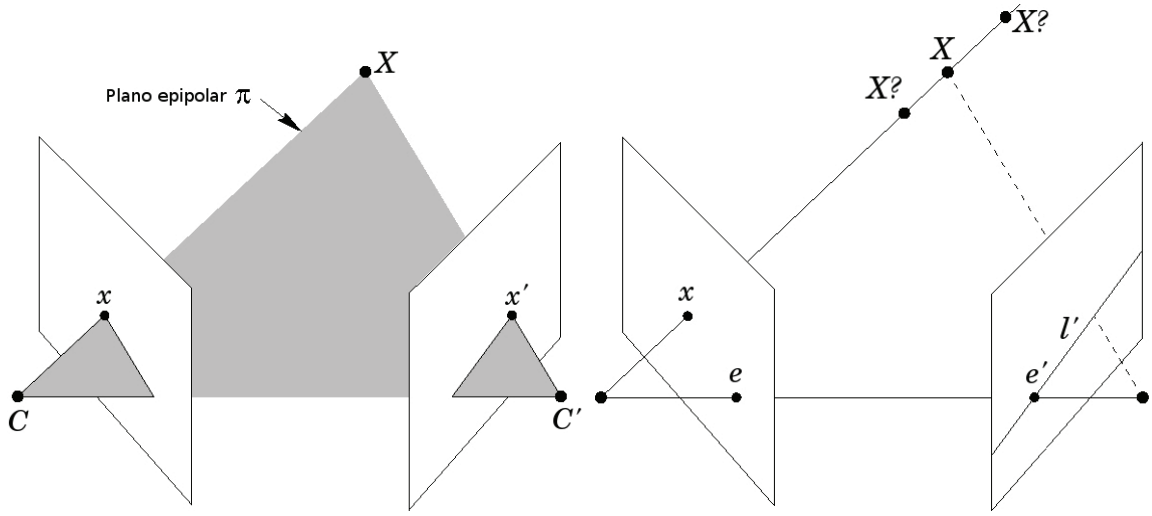
ambas cámaras deben estar sincronizadas. Si no se capturan sus imágenes al mismo tiempo, puede generar problemas cualquier cosa se está moviendo en la escena (incluyendo las cámaras en sí).

La Figura 2.9 demuestra la situación real entre dos cámaras y la alineación matemática que se busca. Para ejecutar esta alineación matemática, se deben utilizar el conocimiento de la geometría de dos cámaras viendo una escena.

2.4. Geometría Epipolar

En este trabajo para obtener las correspondencias estéreo se utiliza la restricción que brinda la geometría epipolar a fines de simplificar la búsqueda. Independientemente de la estructura de la escena, ésta es la geometría proyectiva intrínseca cuando dos vistas se solapan. Sólo depende de los parámetros internos de las cámaras y sus posiciones re-

se inclinan las cámaras ligeramente una hacia la otra de manera que sus rayos principales se intersecten a una distancia finita. Luego de la alineación matemática, el efecto de tal inclinación es introducir un x-offset que se substrae desde la disparidad. Esto puede resultar en disparidades negativas, pero se gana resolución en profundidad a distancias más cercanas.



(a) Definición de las cámaras (b) Reproyección al espacio

Figura 2.10: Geometría epipolar. Imagen modificada de [3].

lativas. Combina dos modelos pinhole (uno por cada cámara[8]). Esta geometría epipolar entre dos vistas es la geometría de la intersección de los planos de imagen con el haz de planos teniendo la línea base como eje (la línea que une los centros de cámara).

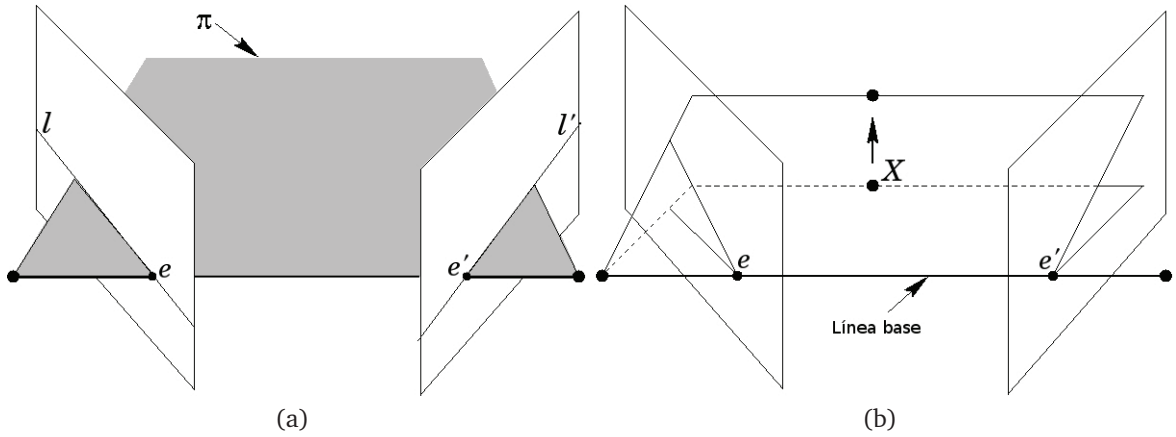
El plano determinado por ambos centros de cámara y el punto del espacio X es el plano epipolar π (Figura 2.10). Los puntos visualizados x y $x^{'}$ son coplanares sobre dicho plano, al igual que los rayos que se reproyectan desde x y $x^{'}$, intersectando a X. Todo punto del espacio X proyectará un punto al correspondiente plano de imagen si se encuentra dentro del cono de visibilidad de la cámara en particular [3, 20].

Las cámaras se definen por sus centros C, $C^{'}$ y sus planos de imagen (Figura 2.10a). Un punto de imagen x reproyecta un rayo al espacio tridimensional definido por el centro de la primer cámara, C, y por sí mismo. Dicho rayo se ve como la línea $l^{'}$ en la segunda vista. Dado que el punto X del espacio tridimensional que proyecta a x se encuentra sobre ese rayo, la imagen de X en la segunda vista debe encontrarse en $l^{'}$ (Figura 2.10b).

Los epipolos e y $e^{'}$ se definen por la intersección de la línea base con los planos de imagen (la línea base es la que une ambos centros de imagen). Un plano epipolar π es aquel plano que contenga tal línea base, intersectando ambos planos de imagen en las líneas epipolares l y $l^{'}$ (ver Figura 2.11a). A medida que el punto tridimensional X varía, los planos epipolares rotan alrededor de la línea base. Todas las líneas epipolares se intersectan en el epipolo (Figura 2.11b) [20].

Sabiendo la ubicación del punto x, se deduce de que manera está condicionado el

[8]Debido a que se utilizan lentes reales, es importante resolver la distorsión.



(a) (b)

Figura 2.11: Plano epipolar e intersección con la línea base. Imagen modificada de [3].

punto correspondiente $x^{'}$. El plano π se determina por la línea base y el rayo que pasa a través de x. Como se mencionó, el punto $x^{'}$ (aún desconocido) se encuentra en π, y dado que la intersección entre π y el segundo plano de imagen determina la línea $l^{'}$, $x^{'}$ se encuentra sobre dicha línea. En términos de un algoritmo de correspondencia estéreo el beneficio es que la búsqueda del punto correspondiente a x no necesita cubrir el plano de imagen por completo sino que se puede restringir a la línea $l^{'}$.

2.4.1. La matriz fundamental F

La matriz fundamental F constituye la principal herramienta en reconstrucción tridimensional, porque representa geométricamente la restricción epipolar. Dado un par de imágenes, para cada punto x en una imagen, existe una línea epipolar $l^{'}$ correspondiente en la imagen opuesta (Figura 2.10). Todo punto $x^{'}$ en la segunda imagen correspondiente al punto x debe encontrarse en dicha línea epipolar $l^{'}$. Además, ambos puntos, x en la primer vista, y $x^{'}$ en la segunda, satisfacen la relación $x^{'T}Fx = 0$ [10,11].

La línea epipolar es la proyección en la segunda imagen del rayo desde el punto x a través del centro de cámara C de la primer cámara. Entonces, existe un mapeo

$$x \mapsto l^{'}$$

de un punto en una imagen a su línea epipolar correspondiente en la otra imagen. Dicho mapeo es una correlación (singular), esto es un mapeo proyectivo de puntos a líneas, lo cual se representa mediante la matriz F, la matriz fundamental.

Propiedades de la matriz fundamental F

Dada la matriz fundamental F, cada par de puntos correspondientes $(x, x^{'})$ satisfacen la relación

$$x^{'T} F x = 0 \tag{2.9}$$

siendo F una matriz de 3 X 3 de rango 2, la ecuación 2.9 puede reescribirse como

$$x^{'T} l^{'} = 0 \tag{2.10}$$

donde

$$l^{'} = F x. \tag{2.11}$$

Es la línea epipolar asociada al punto de la vista izquierda x. La ecuación 2.10 establece que el punto $x^{'T}$ pertenece a la línea $l^{'}$. El cálculo de la matriz fundamental F es posible de dos maneras: a partir de un conjunto de puntos correspondientes en ambas vistas en el caso de no contar con cámaras calibradas o mediante reconstrucción métrica en caso contrario, siendo el segundo caso el que brinda resultados más precisos [11].

2.4.2. La matriz esencial E

La matriz esencial E es la especialización de la matriz fundamental F. Se denomina esencial porque establece la relación entre cámaras sin introducir la no esencial asunción del conocimiento de las matrices intrínsecas de ambas. Simplemente contiene información sobre traslación y rotación que relaciona ambas cámaras en el espacio físico (Figura 2.12). Por otro lado, la matriz fundamental F contiene la misma información que E, además de las intrínsecas de ambas cámaras[9]. Debido a que F embebe información acerca de parámetros intrínsecos, relaciona ambas cámaras en coordenadas de pixel [3, 11].

Relaciona la ubicación, en coordenadas físicas, del punto P visto desde la cámara izquierda con el punto correspondiente visto por la cámara derecha (es decir, relaciona p_l con p_r). La matriz fundamental F relaciona los puntos entre ambos planos de imagen en coordenadas de píxel (para lo cual se utiliza la notación q_l y q_r).

[9]La matriz E se describe prácticamente igual que una matriz de homografía H (matriz la que representa una transformación proyectiva lineal que define un mapeo entre planos) . Ambas se construyen a partir de información similar, pero no son lo mismo. Una homografía H puede relacionar un punto en un plano al punto sobre el plano de la cámara. La matriz E sólo es capaz de relacionar un punto en una imagen a una línea en el otro.

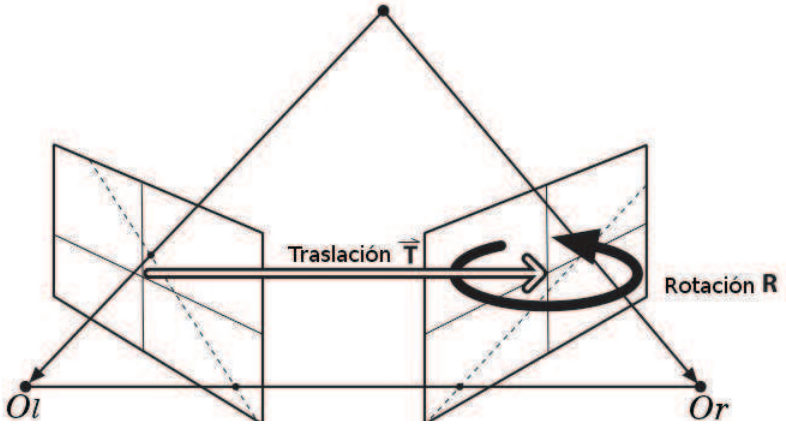


Figura 2.12: La geometría esencial de la formación de imágenes estéreo se representa con la matriz esencial E, contiene la traslación T y la rotación R de la segunda cámara relativa a la primera en coordenadas globales. Imagen modificada de [11].

2.4.3. Cálculo de la matriz Esencial

La relación existente entre las dos visualizaciones del punto tridimensional P, p_l y p_r, sirve como definición de la matriz esencial E. Estos puntos visualizados p_l y p_r, representan las ubicaciones físicas en las coordenadas de ambas cámaras sobre el plano proyectivo de cada una y se pueden relacionar utilizando geometría epipolar. Tales puntos no deben confundirse con P_l y P_r, que son las ubicaciones del punto P en los sistemas de coordenadas de cada cámara.

Es posible elegir cualquier conjunto de coordenadas, izquierda o derecha. En este trabajo se elige que el sistema de coordenadas esté centrado en O_l (el centro óptico de la cámara izquierda). En estas coordenadas, la ubicación del punto observado es P_l, el origen de la otra cámara se ubica en T, y R es la rotación entre cámaras.

El punto P visto en las coordenadas de la cámara derecha es P_r, entonces se tiene la relación

$$P_r = R(P_l - T). \tag{2.12}$$

Aquí, la introducción del plano epipolar resulta útil. Dicho plano se puede representar de diversas maneras, una de ellas es con la ecuación 2.13, la cual indica que todos los puntos x en un plano con vector normal n y pasando por el punto a tiene la siguiente restricción

$$(x - a).n = 0. \tag{2.13}$$

Dado que el plano epipolar contiene los vectores P_l y T, teniendo un vector perpendicular a ambos (por ejemplo, $T \times P_l$), se puede usar como n en la ecuación del plano

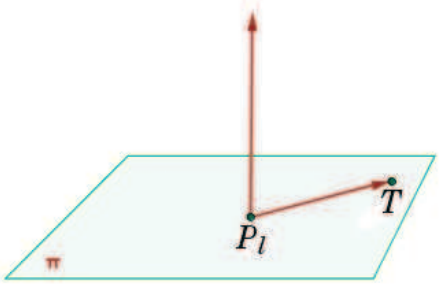


Figura 2.13: Plano definido por vector normal y punto

2.13 (Figura 2.13). Entonces una ecuación para todos los posibles puntos P_l a través del punto T y conteniendo ambos vectores puede ser

$$(P_l - T)^T (T \times P_l) = 0. \tag{2.14}$$

El objetivo final es relacionar q_l y q_r (coordenadas de píxel), primero relacionando P_l y P_r. Se obtiene P_r según la ecuación 2.12, la cual por conveniencia se reescribe $(P_l - T) = R^{-1} P_r$. Haciendo esta sustitución y ya que $R^T = R^{-1}$ dado que R es ortogonal resulta

$$(R^T P_r)^T (T \times P_l) = 0. \tag{2.15}$$

Siempre se puede reescribir un producto cruz como una multiplicación de matrices. Definiendo la matriz S como

$$T \times P_l = S P_l \Longrightarrow S = \begin{vmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{vmatrix}. \tag{2.16}$$

Esto lleva al primer resultado. Haciendo esta sustitución para el producto cruz resulta

$$(P_r)^T R S P_l = 0. \tag{2.17}$$

Este producto RS es lo que define la matriz esencial E, lo que lleva a la ecuación compacta

$$(P_r)^T E P_l = 0. \tag{2.18}$$

Con esto se llega a la relación que existe entre las coordenadas del punto en cada uno de los dos sistemas. Pero lo que realmente se requiere es una relación entre los

puntos vistos desde los dispositivos, entonces, se sustituye usando las ecuaciones de proyección del modelo pinhole (ver ecuación 2.2) $p_l = f_l P_l / Z_l$ y $p_r = f_r P_r / Z_r$ y luego dividiendo todo por $Z_l Z_r / f_l f_r$ para obtener el resultado final

$$p_r^T E p_l = 0. \tag{2.19}$$

Parecería a simple vista que dado uno de los términos p, el otro queda completamente especificado, pero resulta que E es una matriz de rango deficiente[10], en consecuencia esto termina siendo una ecuación para una línea.

2.4.4. Cálculo de la matriz Fundamental

La matriz E contiene toda la información acerca de la relación geométrica de las dos cámaras una respecto de la otra, pero ninguna información de las cámaras en sí. En la práctica lo que interesa son las coordenadas de píxel. Entonces, para encontrar una relación entre píxeles de una imagen y su línea epipolar correspondiente en la otra, se debe introducir información de las intrínsecas de ambas cámaras, sustituyendo p por q. Por ello se utiliza la matriz intrínseca de la cámara que los relaciona, es decir $q = Mp$ (siendo M la matriz intrínseca de la cámara) o, equivalentemente, $p = M^{-1} q$ [11].

Aquí la ecuación para E resulta

$$q_r^T (M_r^{-1})^T E M_l^{-1} q_l = 0. \tag{2.20}$$

Para simplificar dicha expresión, se define la matriz fundamental F como

$$F = (M_r^{-1})^T E M_l^{-1} \tag{2.21}$$

por lo tanto

$$q_r^T F q_l = 0. \tag{2.22}$$

Para resumir, la matriz fundamental F es similar a la matriz esencial E, excepto que F opera en coordenadas de píxel de la imagen mientras que E opera en coordenadas

[10] Para una matriz cuadrada de n x n como E, de rango deficiente significa que hay menos de n valores propios distintos de cero. Como resultado, un sistema de ecuaciones lineal especificado por una matriz de rango deficiente no tiene una única solución.

físicas[11]. Así como E, la matriz fundamental F es de rango 2.

[11]Notar en la ecuación 2.21 que si se tienen imágenes rectificadas y se normalizan los puntos dividiendo por las longitudes focales, la matriz intrínseca M se vuelve la matriz identidad y $F = E$.

Capítulo 3

Calibración de cámaras

En esta sección se describe la transformación proyectiva entre dos planos y como se utiliza para la calibración de la cámara mediante un objeto plano conocido. Luego se expone como relacionar en el espacio las cámaras que componen el dispositivo estereoscópico y la manera de alcanzar el modelo estéreo ideal o modelo paralelo-frontal.

3.1. Transformaciones mediante homografías

Una homografía H es una transformación proyectiva que determina una correspondencia entre dos figuras geométricas planas. El mapeo de un punto $\tilde{Q}$ desde una superficie plana al sensor CCD de la cámara es un ejemplo de la misma. Dicho mapeo se puede expresar con una multiplicación de matrices según [11, 21]

$$\tilde{q} = sH\tilde{Q}, \tag{3.1}$$

donde el parámetro s es un factor de escala arbitrario. Tal matriz consta de dos partes: la transformación de coordenadas entre los sistemas del objeto y la cámara y la proyección que introduce la matriz intrínseca . Siendo W la matriz donde se combina la rotación R y la traslación t, y M la matriz intrínseca, la homografía resulta [11, 22]

$$\tilde{q} = sMW\tilde{Q}. \tag{3.2}$$

Dado que se está visualizando un objeto plano, no interesa la coordenada $\tilde{Q}$ definida en todo el espacio, sino la coordenada $\tilde{Q}'$ definida sólo sobre el plano. Para ello, se define el objeto plano de manera que $Z = 0$. Por ello, si se divide la matriz de rotación en 3 vectores columnas ($R = [r_1 r_2 r_3]$), una de las columnas ya no se requiere según

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}.$$

En consecuencia, la matriz de homografia H que mapea un puntos de un objeto plano en el sensor se describe completamente mediante

$$H = sM \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{3.3}$$

y

$$\tilde{q} = sH\tilde{Q}', \tag{3.4}$$

siendo H una matriz de 3 x 3.

Con múltiples imágenes del mismo objeto plano se resuelven las homografías para calcular las traslaciones y rotaciones así como las intrínsecas de la cámara. Dado que el mapeo de un cuadrado a cualquier cuadrilátero se describe mediante cuatro puntos (x, y), cada homografía brinda ocho ecuaciones al costo de seis incógnitas extrínsecas nuevas. Con suficientes imágenes se pueden calcular cualquier número de incógnitas. La homografía H mapea las posiciones de los puntos del plano origen al plano destino. Sólo se requiere la relación entre puntos origen y destino, por ello se calcula H sin necesidad de conocer las intrínsecas de la cámara [11].

3.2. Calibración mediante objeto conocido

La calibración de cámaras es obligatoria cuando se extrae información métrica a partir de imágenes bidimensionales. Dicha calibración se puede clasificar en: calibración fotogramétrica y autocalibración.

- Calibración fotogramétrica: se observa un objeto de calibración cuya geometría tridimensional se conoce. Se puede subdividir según la cantidad de dimensiones del objeto:
    - Mediante objeto 3D: se trata de un objeto con dos o tres planos ortogonales entre sí. Dicho objeto requiere de mecanismos de fabricación precisos y un sistema de distribución que permita el envío sin deterioro, lo que ocasiona que el costo se incremente.

- Mediante un plano: se realiza mediante la observación de un objeto plano desde múltiples orientaciones. Resulta sencillo contar con un patrón de calibración de este tipo (por ejemplo un tablero de ajedrez), constituyendo un método más eficiente para la calibración que cuando se utiliza un objeto 3D.
- Mediante una línea: el objeto de calibración en este caso es un conjunto de puntos colineales.

- Autocalibración: no se utiliza objeto de calibración, sino que se mueve la cámara observando una escena estática. El movimiento puede ser de rotación y/o traslación (conocidos). Se requieren mecanismos de desplazamiento extremadamente precisos.

En este trabajo se utiliza calibración fotogramétrica mediante objeto plano (un tablero de ajedrez). Conociendo la descripción matemática de las intrínsecas, distorsión radial y tangencial, se observa dicho objeto de calibración conocido desde múltiples orientaciones para calcularlas. Capturando los puntos característicos (Figura 3.1) se calcula la orientación de la cámara con cada imagen así como los parámetros intrínsecos [22–24].

Este tipo de calibración ha tenido múltiples implementaciones en la última década. En general requieren que en múltiples vistas de tablero el usuario identifique la ubicación de tales puntos característicos. En este trabajo se utiliza un método para la extracción automática de tales puntos [23].

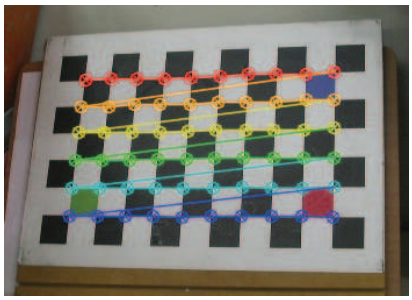


Figura 3.1: Puntos característicos.

Captura de puntos característicos

1. Captura de imagen: Se obtiene una imagen del tablero en escala de gris.
2. Umbralización adaptativa: La umbralización es apropiada para la separación de los casilleros blancos de los negros. El umbral adaptativo puede tomarse como el brillo medio de la imagen. Esta manera de umbralizar permite utilizar imágenes iluminadas de manera no uniforme.
3. Erosión: Separa los casilleros en las esquinas aislando los cuadriláteros negros.

4. Generación de cuadriláteros: Los contornos de estos cuadriláteros se encuentran aplicando un algoritmo de seguimiento de contornos binarios. El mismo busca contornos cerrados, a los cuales se les ajusta un polígono, donde cada vértice es el elemento del vector que los representa [25].

5. Enlace de cuadriláteros:

   a) Por cada esquina de cuadrilátero, se calcula la distancia a todas las esquinas de los demás cuadriláteros. Luego se almacena la menor distancia junto a la esquina correspondiente.

   b) Se verifica si tal distancia es inferior al borde más pequeño de ambos cuadriláteros. Con esto se busca que no se enlacen cuadriláteros muy alejados.

   c) Si se pasan los tests anteriores, ambas esquinas se enlazan y el punto característico se extrae del punto medio entre ambas.

6. Paso adicional: De los múltiples niveles de erosión, el algoritmo selecciona el patrón en el cual se haya encontrado la mayor cantidad de puntos característicos. Si tal cantidad coincide con la cantidad esperada que se brinda como argumento, se considera un patrón válido, de no hacerlo, la imagen es inservible para la calibración.

Refinamiento de sub-píxel

El punto característico encontrado q es cercano a la ubicación real a nivel de subpíxel. Se examinan los vectores $q - p$, siendo p un punto en la vecindad de q. Si p se encuentra en una región uniforme, su gradiente es 0. Por otro lado si el vector $q - p$ está a lo largo de un borde, el gradiente en p es ortogonal a tal vector. En ambos casos, el producto punto entre el gradiente en el punto cercano p y el vector asociado $q - p$ es cero según

$$< \nabla I(p), q - p >= 0. \tag{3.5}$$

Con múltiples valores aleatorios para p, se genera un sistema de ecuaciones iguales a 0 con el que se aproxima q de manera más precisa. Una vez que se encuentra un nuevo valor para q, se lo utiliza como nuevo punto de comienzo para el algoritmo.

Matemáticas de calibración

Sin considerar la distorsión en un principio, se resuelven los demás parámetros utilizando la homografía. Como se vió en la sección 3.1, se puede fijar H igual a la matrix intrínseca M por una combinación de las dos primeras columnas de la matriz de rotación r_1 y r_2, y el vector de traslación t, luego de incluir el factor de escala s (Ecuación 3.2) [24].

Esto produce

$$H = [h_1 h_2 h_3] = sM[r_1 r_2 t]. \tag{3.6}$$

De este sistema de ecuaciones se extraen las relaciones

$$\begin{aligned} &h_1 = sMr_1 \text{ ó } r_1 = \lambda M^{-1} h_1, \\ &h_2 = sMr_2 \text{ ó } r_2 = \lambda M^{-1} h_2 \end{aligned}$$

y

$$h_3 = sMt \text{ ó } t = \lambda M^{-1} h_3,$$

donde

$$\lambda = 1/s. \tag{3.7}$$

Extrayendo la escala s, los vectores de rotación r_1 y r_2 son ortonormales. Dados dos vectores a y b se tiene que $(ab)^T = b^T a^T$. Utilizando el producto punto $r_1^T r_2 = 0$ y sustituyendo r_1 y r_2 con las relaciones previas, se obtienen dos restricciones según

$$h_1^T (M^{-1})^T M^{-1} h_2 = 0, \tag{3.8}$$

y

$$h_1^T (M^{-1})^T M^{-1} h_1 = h_2^T (M^{-1})^T M^{-1} h_2. \tag{3.9}$$

Simplificando mediante $B = (M^{-1})^T M^{-1}$, esta matriz B tiene la forma

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

Usando dicha matriz B ambas restricciones tienen la forma $h_i^T B h_j$ en ellas. Como B es simétrica, dicha forma se puede reescribir como un producto punto por un vector unidimensional b de seis componentes, para que las restricciones se reescriban como

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

Usando la expresión v_{ij}^T, las dos restricciones 3.8 y 3.9 se reescriben como

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

Entonces, recopilando K imágenes de tableros de ajedrez juntos, se forman K pares de estas ecuaciones como

$$Vb = 0.$$

Se resuelve este sistema de ecuaciones para encontrar b, lo cual define la matriz B desde la que se extraen las intrínsecas de cámara. Las extrínsecas se extraen de la ecuación 3.6. Como no se consideraron las distorsiones los puntos visualizados están en el lugar incorrecto. Las intrínsecas calculadas junto a los parámetros de distorsión puestos en cero se utilizan como una aproximación inicial para resolver un sistema de ecuaciones mayor. Denominando (x_p, y_p) a la ubicación en coordenadas de píxel del punto de una cámara sin distorsiones y (x_d, y_d) a su ubicación distorsionada resulta

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} f_x X^W / Z^W + c_x \\ f_y Y^W / Z^W + c_y \end{bmatrix}.$$

Tales resultados de calibración sin distorsión se utilizan en

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

Con esta ecuación evaluada en múltiples puntos se forma un sistema de ecuaciones para calcular los parámetros de distorsión, luego de lo cual los parámetros intrínsecos y extrínsecos se reestiman.

3.3. Calibración estéreo

Hasta aquí se analizaron las cámaras del dispositivo por separado. Ahora se debe realizar la calibración estéreo para relacionarlas calculando la relación geométrica entre ambas. Se calcula la matriz de rotación R y el vector de traslación T entre cámaras, lo

cual trae el sistema de coordenadas de la cámara derecha al de la izquierda. Utilizando las extrínsecas de las calibraciones individuales, se lleva cualquier punto tridimensional P en coordenadas del objeto, a las coordenadas de cualquiera de las cámaras, según [11, 26]

$$P_l = R_l P + T_l \text{ y } P_r = R_r P + T_r. \tag{3.10}$$

Ambas vistas del punto P se relacionan mediante $P_l = R^T(P_r - T)$[1], siendo R la matriz de rotación y T el vector de traslación. Relacionando las ecuaciones 3.10 se obtienen las ecuaciones

$$R = R_r(R_l)^T \tag{3.11}$$

y

$$T = T_r - RT_l. \tag{3.12}$$

Con múltiples vistas de tableros de ajedrez se obtienen igual número de rotaciones y traslaciones que se insertan dentro de las ecuaciones 3.11 y 3.12. Dado que puede existir ruido y/o errores de redondeo, los resultados de cada par de tableros pueden diferir ligeramente para R y T. Para contrarrestar este problema se toma la mediana de R y T como una aproximación inicial y luego se ejecuta un algoritmo iterativo de Levenberg-Marquardt para encontrar la solución final. Dicho algoritmo es un proceso iterativo de convergencia a través de la técnica de los mínimos cuadrados ponderados [27].

3.4. Rectificación estéreo

El cálculo de la disparidad estéreo mediante triángulos semejantes es posible con la configuración paralelo frontal. Por ello se realiza la rectificación estéreo con el fin de reproyectar ambos planos de imagen para volverlos coplanares, con filas perfectamente alineadas.

Los métodos de rectificación se pueden clasificar según si utilizan la matriz fundamental F o no. Un ejemplo de rectificación mediante el uso de matriz fundamental es el algoritmo de Hartley, el cual depende de la correspondencia de puntos en común entre pares de imágenes, se utiliza cuando no se cuenta con patrones de calibración como el tablero de ajedrez. En este proyecto se utiliza el algoritmo de Bouguet, el cual

[1] P_l y P_r indican las ubicaciones del punto tridimensional P en el sistema de coordenadas de la cámara izquierda y derecha respectivamente. R_l y T_l indican los vectores de traslación y rotación desde la cámara al punto tridimensional para la cámara izquierda (así como R_r y T_r representan lo mismo para la cámara derecha). R y T son la traslación y rotación que trae el sistema de coordenadas de la cámara derecha al de la izquierda.

depende de la rotación R y la traslación T. Esa información extra simplifica la tarea de rectificación, y permite resultados más precisos [11, 28].

3.4.1. Rectificación estéreo calibrada: Algoritmo de Bouguet

La rotación R se divide a la mitad entre ambas cámaras, resultando en las matrices r_l y r_r. Con tales rotaciones los planos son coplanares y sus ejes ópticos quedan paralelos en dirección del vector suma de donde habían estado apuntado. Para la alineación de filas se calcula la matriz R_{rect}, la cual produce una rotación alrededor del centro óptico. Las columnas de dicha matriz se obtienen según

$$e_1 = \frac{T}{||T||}, \tag{3.13}$$

$$e_2 = \frac{[-T_y T_x 0]^T}{\sqrt{T_x^2 + T_y^2}}, \tag{3.14}$$

y

$$e_3 = e_1 \times e_2. \tag{3.15}$$

La alineación de filas de las cámaras se obtiene utilizando $R_l = R_{rect}\, r_l$ y $R_r = R_{rect}\, r_r$ para las cámaras izquierda y derecha respectivamente.

3.4.2. Mapa de rectificación

Luego de obtenerse la calibración y rectificación estéreo, se calculan los mapas para transformar las imágenes de entrada y obtener el sistema estéreo requerido. Para cada píxel destino (u, v) los mapas de rectificación se obtienen aplicando la siguiente secuencia.

1. $x \leftarrow (u - c_x^{'})/f_x^{'}$
2. $y \leftarrow (v - c_y^{'})/f_y^{'}$
3. $[XYW]^T \leftarrow R^{-1} * [xy1]^T$
4. $x^{'} \leftarrow X/W$

5. $y^{'} \leftarrow Y/W$
6. $x^{''} \leftarrow x^{'}(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_1xy + p_2(r^2 + 2x^2)$
7. $y^{''} \leftarrow y^{'}(1 + k_1r^2 + k_2r^4 + k_3r^6) + p_1(r^2 + 2y^2) + 2p_2xy$
8. $map_x(u, v) \leftarrow x^{''} f_x + c_x$
9. $map_y(u, v) \leftarrow y^{''} f_y + c_y$

Capítulo 4

Módulo de calibración

En esta sección se presentan los resultados del módulo de calibración. Para iniciar se hace una breve descripción del procedimiento ejecutado, para luego exponer los parámetros intrínsecos y de distorsión obtenidos mediante la calibración individual de cada una de las cámaras. Posteriormente se muestran los resultados de la calibración estéreo e imágenes que sirven de ejemplo para la rectificación. Por último se expone el mecanismo utilizado para medir la precisión de la calibración.

El instrumento utilizado para realizar el módulo de calibración fue la cámara a utilizar en este proyecto, es decir la webcam marca Minoru®.

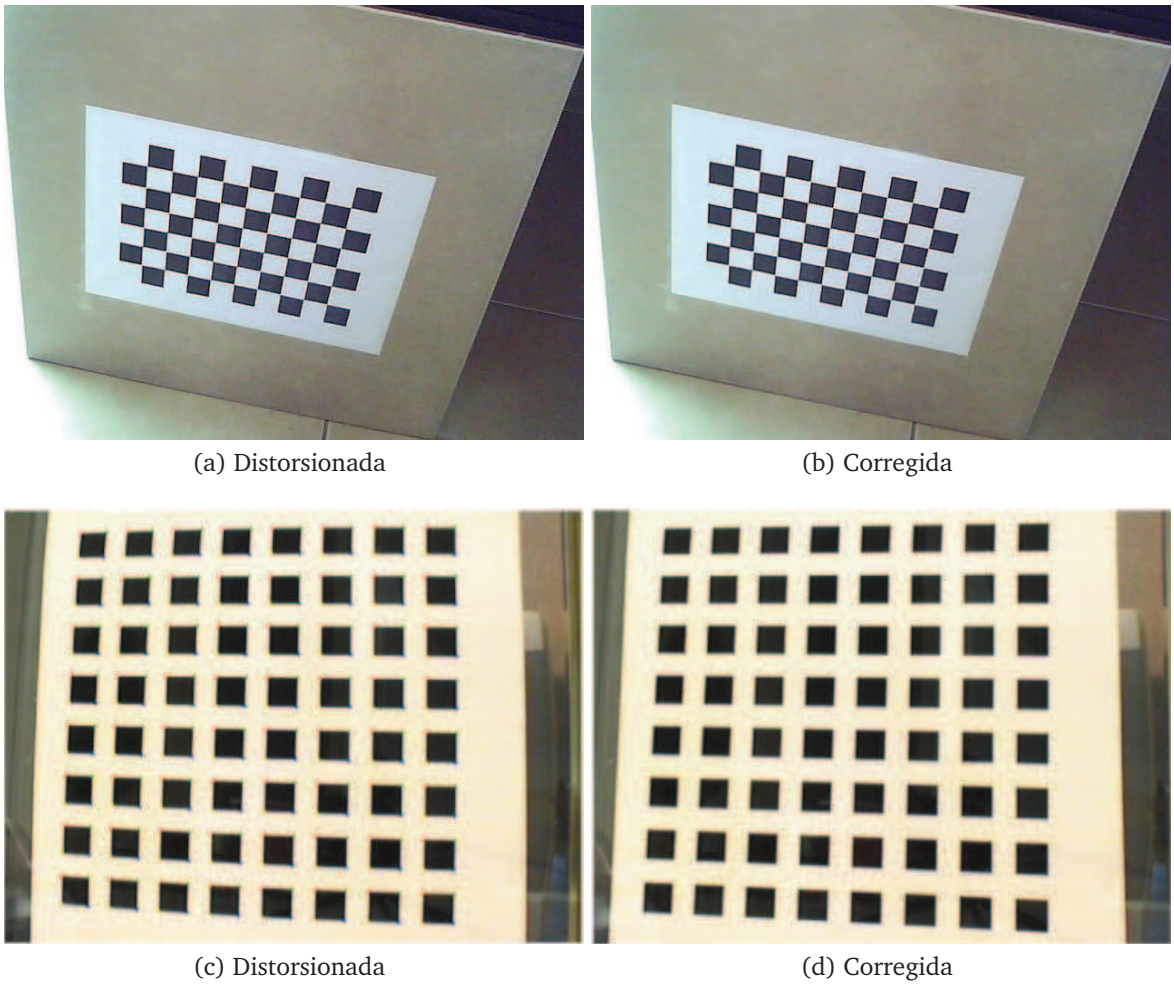
Imágenes de calibración

Se realizó la captura de imágenes de un tablero de ajedrez desde múltiples orientaciones mediante el dispositivo estereoscópico. Los conjuntos de imágenes izquierdo y derecho sirven para calcular los parámetros de calibración del lente izquierdo y derecho respectivamente. Con la captura de los puntos característicos, se calcula la matriz de homografía H que los relaciona. Los valores que componen la homografía sirven de entrada al sistema de ecuaciones que resuelve las matrices intrínsecas y parámetros de distorsión de ambos lentes.

Resultados de calibración

Denominando M_{izq} y M_{der} a las matrices intrínsecas izquierda y derecha, y D_{izq} y D_{der} a los vectores que contienen los parámetros de distorsión. Para éstos los resultados son

$$M_{izq} = \begin{vmatrix} 459{,}47 & 0 & 167{,}06 \\ 0 & 459{,}69 & 90{,}91 \\ 0 & 0 & 1{,}0 \end{vmatrix}, \tag{4.1}$$



(a) Distorsionada (b) Corregida

(c) Distorsionada (d) Corregida

Figura 4.1: Comparación entre imágenes originales y sin distorsión. Arriba, par de imágenes propio del proyecto. Abajo, par de imágenes tomadas de [24].

$$M_{der} = \begin{vmatrix} 451{,}99 & 0 & 152{,}84 \\ 0 & 451{,}64 & 101{,}31 \\ 0 & 0 & 1{,}0 \end{vmatrix}, \tag{4.2}$$

$$D_{izq} = \begin{vmatrix} -0{,}211 & 1{,}67 & 0 & 0 & -8{,}74 \end{vmatrix} \tag{4.3}$$

y

$$D_{der} = \begin{vmatrix} 0{,}0058 & -1{,}47 & 0 & 0 & 7{,}06 \end{vmatrix}. \tag{4.4}$$

Comparación entre imagen distorsionada y su corrección

Aplicando el modelo de corrección según los parámetros calculados se resuelven las distorsiones como se observa en la Figura 4.1. Además de imágenes propias de este proyecto, se exponen dos imágenes donde la corrección de la distorsión se hace más notoria.

Calibración estéreo

Mediante la utilización de las extrínsecas calculadas previamente se resuelve la relación espacial de los lentes, cuyos valores se especifican mediante

$$R = \begin{vmatrix} 0{,}999 & -0{,}0006 & 0{,}026 \\ 0{,}0008 & 0{,}999 & -0{,}0092 \\ -0{,}026 & 0{,}009 & 0{,}999629 \end{vmatrix} \tag{4.5}$$

y

$$T = \begin{vmatrix} -6{,}005 & -0{,}018 & -0{,}04 \end{vmatrix}. \tag{4.6}$$

Se observa que la matriz de rotación R obtenida es cercana a la matriz identidad, esto se debe a que las cámaras se encuentran dispuestas de manera prácticamente coplanar en el dispositivo estereoscópico, con el objetivo de aproximarse lo más posible al modelo ideal para la triangulación. El primer elemento del vector de traslación es la distancia a lo largo de la línea base y se aproxima al valor de 6 cm correspondiente a la webcam utilizada.

Efecto de la rectificación

Con los valores obtenidos mediante el algoritmo de Bouguet para la rectificación, las imágenes originales se remapean al modelo paralelo frontal con las líneas epipolares coincidentes a lo largo de filas. Luego de la rectificación los puntos aparecen en la misma fila (Figura 4.2).

Medida de precisión de la calibración

Las imágenes que se utilizaron para la calibración también se pueden utilizar para verificarla. Al resolver la homografía que relacionaba el tablero de ajedrez con el plano de imagen, se obtuvieron por cada vista los valores de rotación R, traslación T, intrínsecas de cámara y parámetros de distorsión. Tales parámetros se pueden utilizar para reproyectar los puntos del objeto y obtener el error cuadrático medio entre los valores reproyectados y los puntos característicos refinados a nivel subpíxel. Se considera que dicho valor debe ser menor a 1 píxel para configuraciones de alta precisión [29]. En este proyecto el valor obtenido es 0.36, lo cual significa que los puntos reproyectados están en promedio a 0.36 unidad de píxel de su posición real.

La matriz fundamental relaciona las imágenes en coordenadas de píxel, por lo cual también se utiliza para verificar la precisión usando la restricción epipolar $x'^{T} F x = 0$, lo que equivale a la expresión

$$x'^{T} l' = 0. \tag{4.7}$$

El producto Fx permite obtener la línea epipolar l' asociada al punto x. Representando la línea $ax + by + c = 0$ mediante el vector $[a, b, c]$, con los vectores normalizados

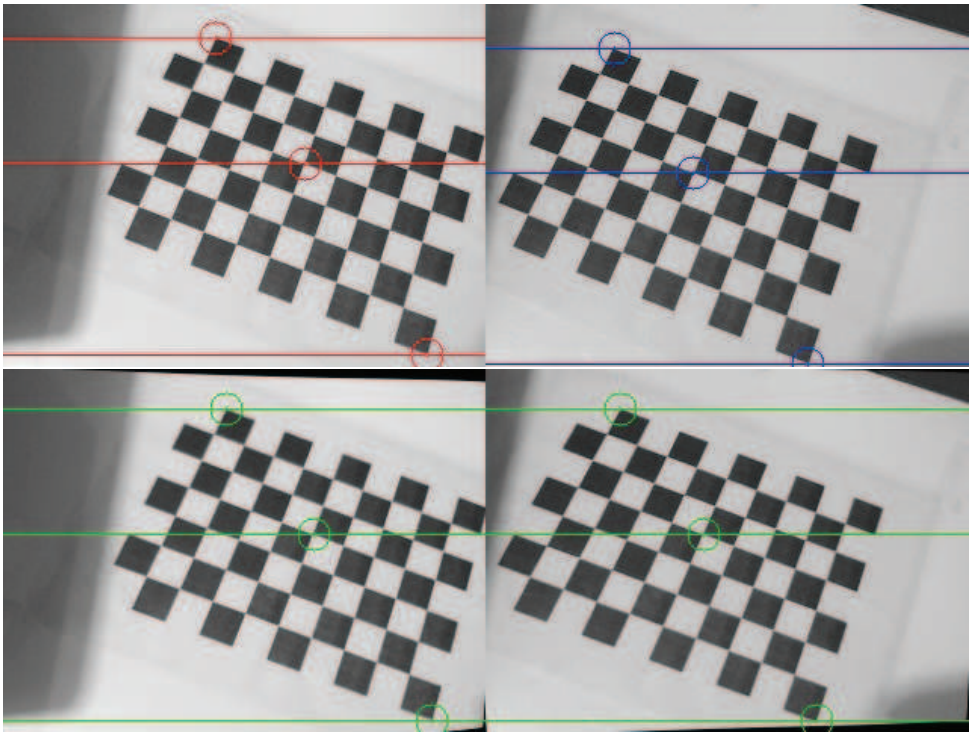


Figura 4.2: Par original (Arriba). Par rectificado (Abajo).

de la forma $a^2 + b^2 = 1$, se calcula el valor promedio para la ecuación 4.7 evaluada en todos los puntos refinados a nivel sub-píxel y se comprueba que tiende a 0. El valor obtenido mediante éste cálculo fue 0.42.

Capítulo 5

Módulo de Navegación

En esta sección se realiza una comparativa de métodos que se utilizan para obtener mapas de disparidad, una descripción del algoritmo utilizado para la generación de los mismos y la manera útil de interpretarlos para el esquive de obstáculos. Posteriormente se procede a la descripción del ensamblaje del Lego® Mindstorms® NXT y de las clases específicas a utilizar de la API de LeJoS. Para finalizar se exponen las nociones básicas de la Arquitectura de Control, la cual sirve para brindarle un marco a la programación de Sistemas Robóticos Inteligentes, en particular para este proyecto, la Arquitectura de Subsunción.

5.1. Comparación de algoritmos de correspondencia

Existe correspondencia estéreo cuando el mismo punto del espacio se identifica en las dos vistas del dispositivo estereoscópico. Dicha correspondencia se puede determinar según determinadas características, como ser intensidad, color o características estructurales como bordes o gradiente de la imagen. Aunque la geometría epipolar reduce el espacio de búsqueda a una dimensión, encontrar correspondencias aún puede ser complicado debido a varias razones, por ejemplo:

- Ruido: valores que difieren debido a errores en la digitalización (muestreo), calibración imperfecta, ruido provocado por el dispositivo de entrada, etc.
- Oclusiones: puntos que sólo son visualizados desde una de las cámaras.
- Definiciones poco claras en las imágenes: regiones de luminosidad constante, lo cual puede conducir a emparejamientos ambiguos.

Los algoritmos de correspondencia se agrupan según la salida que producen: dispersa (sobre un conjunto reducido) o densa (disparidad calculada en todos los píxeles). Los algoritmos de disparidad dispersa se basan en puntos identificables o distinguibles en una imagen, porque realizan emparejamiento entre segmentos o bordes entre dos imágenes. Este proyecto requiere que la disparidad se calcule en todos los puntos posibles, sin discriminar en base a características, por ello se analizarán en profundidad solamente los algoritmos que producen salidas densas.

5.1.1. Algoritmos de disparidad dispersa

Resultan útiles cuando se requiere cálculo de profundidad veloz y al mismo tiempo poco nivel de detalle en la imagen completa. Se enfocan en características salientes en las imágenes dejando oclusiones y áreas pobremente texturadas fuera de los cálculos, por lo que se consiguen altas velocidades de procesamiento, con densidad limitada.

5.1.2. Algoritmos de disparidad densa

Los métodos que producen disparidad densa han ganado popularidad a medida que el poder computacional ha crecido. Dichos algoritmos se clasifican en locales (basados en áreas) y globales. Los métodos locales tienen a su favor la velocidad de cálculo mientras que los globales la precisión en el mapa obtenido. Dichos métodos locales se denominan también métodos basados en ventanas o áreas, porque la disparidad en un punto dado depende de los valores de intensidad de una ventana de soporte finito. Los métodos globales (basados en energía) son precisos pero costosos computacionalmente. Su objetivo es minimizar una función de costo global, combinan datos y términos de suavización teniendo en cuenta la imagen completa [30, 31].

Métodos globales

Producen resultados muy acertados. Su objetivo es encontrar la disparidad óptima $d = d(x, y)$ la cual minimiza una función de costo global E, la que combina datos y términos de suavización:

$$E(d) = E_{data}(d) + \lambda . E_{smooth}(d), \tag{5.1}$$

donde E_{data} considera los valores de los píxeles (x, y) a lo largo de toda la imagen, E_{smooth} provee al algoritmo asunciones de suavización y λ es un factor de peso. La desventaja de estos métodos es que tienen demasiada demanda computacional, haciéndolos inapropiados para utilizar en este proyecto, ya que se tiene que utilizar un algoritmo capaz de responder en tiempo real [30].

Métodos locales

Comparan píxeles en una vecindad en lugar de píxeles individuales. Esto es porque se basan en la intensidad de los mismos, lo cual puede tender a inexactitudes debido a ruido en la imagen, cambios en la iluminación o desenfoque por el movimiento. Son rápidos a la vez que producen mapas de disparidad aceptables. Varían según la función de costo utilizada para evaluar la similaridad en la ventana. Son los algorimos que se encuentran mayormente utilizados [31, 32].

Las comparaciones se hacen a lo largo de las mismas líneas de píxeles gracias al modelo brindado por la geometría epipolar. El mecanismo de dichos métodos consiste en tomar alguna ventana de una imagen y desplazarla sobre la otra, a la vez que aplican alguna función de costo [3, 32].

Funciones de costo:

- **Sum of Absolute Differences (SAD):** Calcula similaridad en una ventana W según

$$SAD(x,y) = \sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)|. \tag{5.2}$$

  Dicho cálculo representa la suma de diferencias en valor absoluto entre píxeles de la ventana izquierda y la derecha. La mejor correspondencia será donde se obtenga el menor valor, cuyas coordenadas se utilizarán para definir la disparidad. Agrandando la ventana se obtienen mapas de disparidad más precisos pero se incrementa el costo computacional. Según el tipo de aplicación, se debe equilibrar entre velocidad de procesamiento y calidad de dicho mapa [33].

- **Sum of Squared Differences (SSD):** utiliza la función de costo

$$SSD(x,y) = \sum_{(i,j)\in W} (I_1(i,j) - I_2(x+i, y+j))^2. \tag{5.3}$$

  Tienen mayor complejidad computacional que SAD debido a la potencia cuadrática presente en la fórmula, ya que el cálculo de cuadrados es una multiplicación mientras que el valor absoluto es una comparación [34].

- **Normalized Cross Correlation (NCC):** las correspondencias se obtienen dividiendo la sumatoria normalizada del producto de las intensidades en la ventana por la desviación estándar de dichas intensidades según

$$NCC(x,y,d) = \frac{\sum_{x,y\in W} I_1(x,y).I_2(x,y-d)}{\sqrt{\sum_{x,y\in W} I_1^2(x,y).\sum_{x,y\in W} I_2^2(x,y-d)}}. \tag{5.4}$$

Comparativa

Se toma como referencia un trabajo donde se realizó una comparación de los métodos implementados en Matlab (Tabla 5.1), utilizando un Intel Core i5 con gráficos embebidos. Los algoritmos se aplicaron a pares de imágenes estéreo de 384 x 288 píxeles. De acuerdo a los resultados la implementación mediante SAD es la elección conveniente para sistemas de tiempo real, dado que tienen el menor costo computacional a medida que crece el tamaño de la ventana. En este proyecto se utilizan ventanas de 41 píxeles de lado [34].

Tabla 5.1: Tiempo de cálculo en segundos [34].

	Tamaño de la ventana		
Función de costo	3	11	25
SAD	0.138	0.142	0.160
SSD	0.136	0.145	0.174
NCC	0.215	0.279	0.339

5.2. Generación del mapa de disparidad

El tamaño del mapa de disparidad resultante será menor al de las imágenes originales debido al rango de disparidad y tamaño de la ventana SAD que se utilice.

Generar el mapa de disparidad consta de tres etapas:

1. Prefiltrado: las imágenes se normalizan para reducir diferencias en brillo y realzar texturas. Para ello se desplaza una ventana cuyo píxel central I_c se reemplaza por min[max($I_c - \tilde{I}$,$-I_{cap}$),I_{cap}], siendo $\tilde{I}$ el brillo promedio en la ventana e I_{cap} un valor positivo que por defecto es 30.

2. Búsqueda de correspondencias: se buscan características con textura suficiente[1] a lo largo de líneas epipolares horizontales mediante ventana SAD. Dicho requerimiento acerca de la textura se debe a que áreas demasiado homogéneas están sujetas a correspondencias ambiguas.

3. Posfiltrado: se verifica la diferencia en porcentaje existente entre la mejor coincidencia (menor valor calculado en el desplazamiento) y la siguiente en orden ascendente. Si la diferencia no supera el porcentaje esperado no se considera la disparidad calculada para ese píxel. El margen va de 0 a 100 donde 0 especifica que no se requiere esta restricción.

[1] Se define un mínimo para la suma de diferencias absolutas en la ventana por debajo del cual, no se realiza la búsqueda.

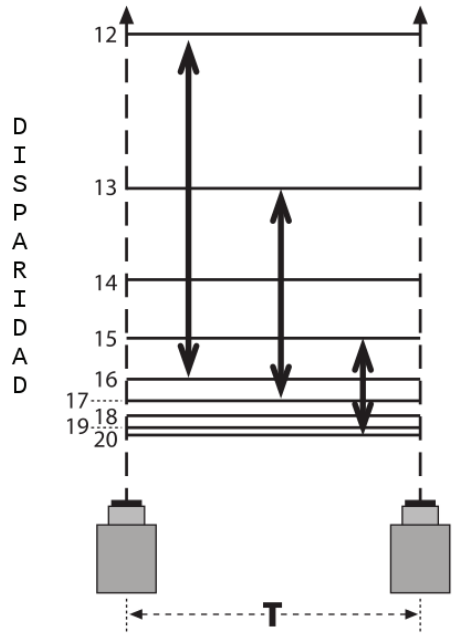


Figura 5.1: Horóptero obtenido según diferentes rangos de disparidad. Imagen modificada de [11].

La coordenada de comienzo de búsqueda (disparidad mínima) y el número de disparidades a buscar definen el horóptero[2] cubierto por el dispositivo estereoscópico, fuera del cual la profundidad no es calculable. La Figura 5.1 muestra los horópteros cubiertos por el dispositivo con distintos rangos de disparidad a buscar. Decrementando la distancia T entre cámaras o la longitud focal, o, incrementando el rango de búsqueda o el ancho del pixel, se maximiza dicho horóptero (las vistas se superponen en mayor medida). Reduciendo el número de disparidades a buscar se recorta el tiempo computacional limitando la longitud de la búsqueda [11].

Casi todas las escenas estéreo obedecen la restricción de orden. Si los puntos A, B y C aparecen de izquierda a derecha en la imagen izquierda, y B no aparece en la imagen derecha porque está ocluído, A y C seguirán viéndose de izquierda a derecha. Este algoritmo asume dicha restricción [11, 35].

Se espera que el mapa de disparidad denso sea suave y a la vez detallado. Cuando un algoritmo genera un mapa de disparidad suave se tiende a perder detalle, en contraposición mapas muy detallados suelen ser ruidosos. Este algoritmo genera ruido en los bordes porque la ventana toma el primer plano por un lado y el segundo plano por el otro, generando una región local de disparidades grandes y pequeñas denominadas moteo (speckle). Para prevenir estos emparejamientos de borde, se fija un detector de moteo a través de una ventana de moteo. Dentro de esta ventana el emparejamiento es permitido siempre que el rango entre disparidad mínima y máxima sea el permiti-

[2]Volumen 3D que se cubre por el algoritmo de búsqueda estéreo.

do [11,36].

Interpretación del mapa de disparidad

Una vez que se cuenta con los mapas de disparidad, se requiere un algoritmo para detectar y esquivar posibles obstáculos al frente del robot. De acuerdo a dicha información, el módulo de navegación define hacia donde rotar el Lego® Mindstorms® NXT.

Para ello, el mapa de disparidad se divide en tres ventanas: una central y dos laterales. Por cada mapa de disparidad obtenido se procede según:

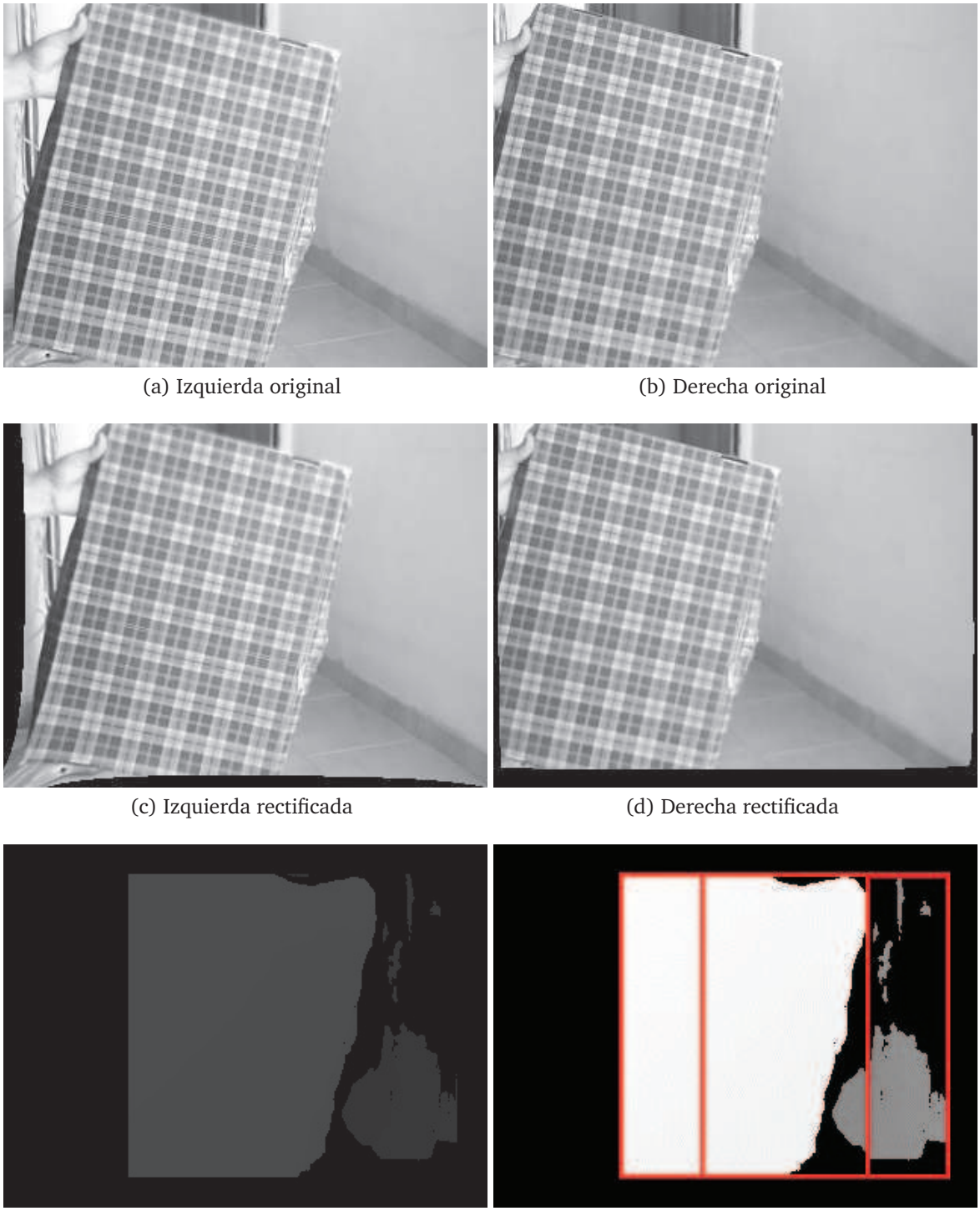
- De las ventanas, se calculan los píxeles p cuyo valor de disparidad $D(p)$ es superior a un umbral definido T, el cual representa puntos característicos cercanos al dispositivo estereoscópico.
- Se examina tal cantidad. Si el porcentaje que representa con respecto al total de píxeles en dicha ventana no supera un porcentaje predefinido r, se supone que no existen obstáculos detectados en esa dirección. Si se detecta colisión[3] en la ventana central, se rota hacia la derecha o la izquierda de acuerdo al menor porcentaje obtenido de las ventanas laterales [37].
- Si no se detecta obstáculo en la ventana central, se examinan las ventanas laterales. La ocurrencia de obstáculo en las mismas generará una rotación en la dirección opuesta. El ángulo de rotación es una fracción de la rotación que ocurre en el caso de colisión en la ventana central.

Se observa en la Figura 5.2, la secuencia a partir de la captura de cada par de frames. En primer lugar se realiza la rectificación estéreo, luego de la cual se puede obtener la disparidad correspondiente. Luego dicha imagen se divide en las tres ventanas mencionadas para su interpretación.

5.3. Tracción diferencial

Este proyecto se realiza con el Lego® Mindstorms® NXT ensamblado en el formato de robot diferencial, esto es con dos ruedas a cada lado del cuerpo del robot controladas por sendos motores independientes, más una tercer rueda para brindarle estabilidad (Figura 5.3). Para el avance en línea recta, ambas ruedas rotan en la misma dirección a igual velocidad. Si las ruedas rotan en dirección opuesta a igual velocidad, el robot gira alrededor del centro de eje. De otra manera, según la velocidad de rotación y dirección, el centro de rotación caerá en cualquier punto sobre la línea definida por los dos puntos de contacto de las cubiertas.

[3]El término colisión no representa un choque físico real, sino la detección de puntos pertenecientes a un obstáculo a poca distancia del robot, lo cual obliga al esquive del mismo.



(a) Izquierda original

(b) Derecha original

(c) Izquierda rectificada

(d) Derecha rectificada

(e) Disparidad

(f) Disparidad normalizada

Figura 5.2: Generación del mapa de disparidad.

Figura 5.3: Lego® Mindstorms® NXT 2.0 ensamblado en formato diferencial.

Con esta configuración se puede utilizar odometría, esto es estimar su posición en función de las rotaciones de las ruedas del robot. La información provenienente de odometría es el método más utilizado para calcular el posicionamiento del robot. Esto se debe a que provee precisión aceptable con poco costo computacional. Por ello, tal método es el candidato para este proyecto, donde se requiere obtener respuestas en tiempo real de un software instalado en la plataforma Raspberry Pi. Sin embargo, se debe considerar que la idea detrás de la odometría es la sumatoria de movimientos a través del tiempo, lo que inevitablemente conducirá a errores acumulativos. Tales errores pueden ser causados por múltiples factores, tales como [38]:

- Diámetro de ruedas desigual.
- Ruedas desalineadas.
- Tránsito sobre terreno desparejo o resbaloso.
- Fuerzas externas (interacción con objetos externos).

Control del Lego® Mindstorms® NXT mediante el firmware LeJOS NXJ

Este proyecto utiliza el LeJOS NXJ como reemplazo del firmware original del Lego, el cual brinda mayor versatilidad en la programación que el lenguaje original denominado NXT-G. Es una pequeña Java Virtual Machine que incluye[4]:

- Un lenguaje orientado a objetos (Java).

[4] `http://lejos.sourceforge.net/nxj.php`

- Programación multihilo.
- Arreglos multidimensionales.
- Recursión.
- Sincronización.
- Manejo de excepciones.
- Tipos de datos Java incluyendo float, long y string.
- Una API sobre robótica bien documentada.

Mediante el uso de la biblioteca de clases Java (classes.jar) que implementan el API (Application Programming Interface) de leJOS NXJ, es posible el control del Lego® Mindstorms® NXT sin necesidad de implementar clases adicionales. En este proyecto se utilizan dos en particular: DifferentialPilot para controlar los movimientos del robot y OdometryPoseProvider para registrar la posición actual del mismo a partir de los movimientos que ha realizado.

Clase DifferentialPilot

Las clases LeJOS NXJ trabajan a diferentes niveles de abstracción. En el nivel inferior, la clase NXTRegulatedMotor controla el motor que gira la rueda. Por encima de ésta se encuentra la clase DifferentialPilot. La misma usa NXTRegulatedMotor para realizar movimientos elementales: rotar en el lugar, transitar en línea recta, o en forma de arco. Para lograr esto controla la velocidad y dirección de rotación de los motores.

La clase DifferentialPilot necesita saber a que puertos se conectan los motores y si rotar los motores hacia adelante hace que el robot se mueva hacia adelante o hacia atrás, el diámetro de las ruedas y el ancho del eje. Con el diámetro de la rueda se calcula la distancia que transitó y con el ancho del eje cuanto se rotó. Ambos parámetros deben estar en las misma unidad de medida. Con el ajuste de estos parámetros, se considera que los errores en la distancia recorrida y ángulo rotado son menores o iguales al 2 %[5].

Clase OdometryPoseProvider

Esta clase estima la ubicación actual y la dirección a la cual apunta el robot (heading). Utiliza coordenadas cartesianas, con ángulos en grados, donde el ángulo 0 representa la dirección positiva del eje x y 90 a dirección positiva del eje y. Cuando se instancia se registra como un listener[6] con su MoveProvider (la instancia de DifferentialPilot), el cual le informa de los movimientos que realiza. Con tal información es capaz de realizar navegación por estima (dead reckoning), esto es estimar la ubicación actual

[5] `http://lejos.sourceforge.net/nxt/nxj/tutorial/WheeledVehicles/WheeledVehicles.htm`
[6] Objeto que escucha eventos.

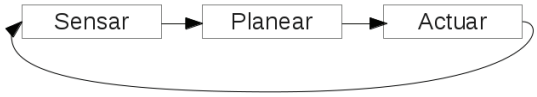
utilizando trigonometría basada en el rumbo y la velocidad de navegación a lo largo de un período. La dirección y coordenadas (x, y) se almacenan en un objeto denominado Pose. Tal objeto permite obtener el ángulo de rotación relativa (relative bearing) a un punto que se indique como destino, lo cual resulta útil para orientar al Lego® Mindstorms® NXT a un punto definido como destino.

5.4. Arquitectura de control

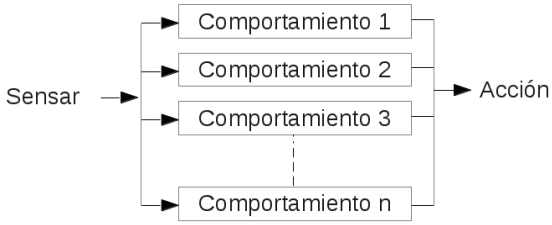
En robótica, no resulta sencillo encontrar métodos sistemáticos de análisis debido a la naturaleza interdisciplinaria de la misma. Representa un desafío combinar racionalmente módulos que incluyan percepción, planeamiento y toma de decisiones para generar un sistema inteligente (IRS: Intelligent Robot System). Esta carencia de métodos sistemáticos con enfoques de reutilización, conduce muchas veces a que los desarrolladores de sistemas robóticos, se encuentren reinventando la rueda cada vez que comienzan un proyecto. Esto es debido a que la diversidad de aplicaciones robóticas dificulta establecer una estructura unificada para el análisis.

No obstante, se han podido identificar tres paradigmas generales a la hora de crear un IRS según la manera en que se relacionan las acciones básicas de un robot (sensar, planear y actuar): el reactivo, el deliberativo y el híbrido (reactivo/deliberativo) (Figura 5.4) [39].

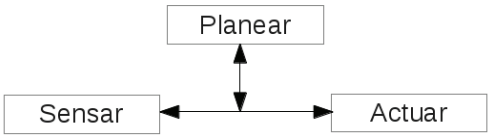
- Paradigma deliberativo o jerárquico: la arquitectura más antigua para los IRS, fue la que prevaleció hasta fines de los años '90'. Consiste en la estructura jerárquica **S**→**P**→**A**. Supone tal orden para las acciones (sensar, planear, actuar), en un procesamiento hacia abajo (top-down). Mediante el sensado se genera un modelo del mundo, el cual es utilizado por el robot para planificar explícitamente la siguiente acción. La desventaja de este paradigma es que cualquier cambio en la misión del robot requiere remodelar, planear desde el principio y rediseñar los programas que controlan los actuadores.
- Paradigma reactivo: propone una conexión directa entre el sensado y las acciones sin requerir la modelización del mundo, propiciando un modelo apropiado para aplicaciones en tiempo real. El fundamento detrás del mismo es que la inteligencia real del robot queda manifiesta en la interacción y adaptabilidad del robot con un entorno real, y no por el conocimiento y tampoco por el razonamiento, y que el mejor modelo de todos, es el mundo en sí. Está basado en el comportamiento observado en algunos organismos vivos. Las acciones posibles del robot se descomponen en comportamientos, los cuales actúan en paralelo. Tales comportamientos se traducen en comandos para los actuadores. Ejemplo de este paradigma es la Arquitectura de Subsunción, la cual se esplicará posteriormente.



(a) Paradigma deliberativo



(b) Paradigma reactivo



(c) Paradigma híbrido

Figura 5.4: Paradigmas en Robótica.

- Paradigma híbrido: combina los modelos anteriores. Permite incorporar la modelización del mundo a la vez que responden en tiempo real a los cambios en el ambiente. En dicho paradigma se puede reconfigurar la capa reactiva, para definir el subconjungo de comportamientos a utilizar Un modelo de este paradigma se denomina “Planificación, Percepción - Acción”. En el mismo el robot realiza un planeamiento sobre como descomponer una tarea en subtareas y elige los comportamientos adecuados para cumplirlas. A partir de ese momento se ejecutan los comportamientos según el paradigma reactivo. La información de sensado también se pone a disposición del módulo de planificación.

Arquitectura de Subsunción

La Arquitectura de Subsunción es una arquitectura para robots reactivos. En oposición a la inteligencia artificial tradicional, en lugar de guiar los comportamientos en base a representaciones simbólicas del mundo, la Arquitectura de Subsunción utiliza información sensorial para guiar las acciones en tiempo real, lo que la hace adecuada para este proyecto.

Consta de una jerarquía modular de comportamientos, los cuales representan una manera de actuar o de ejecutar una tarea. Tales comportamientos entran en competencia entre sí para definir cuál es el que estará activo en un determinado momento

controlando al agente. Los niveles superiores son capaces de subsumir (predominar) sobre los niveles inferiores. Requiere la existencia de un árbitro para definir cuál es el comportamiento activo.

Dichos comportamientos se organizan en una estructura de capas que representa la jerarquía de los mismos, siendo de mayor importancia los comportamientos de las capas superiores. Las capas inferiores representan comportamientos más simples, como la evasión de obstáculos. Cuando sea factible que dos o más comportamientos estén activos al mismo tiempo, el árbitro optará por el comportamiento de mayor jerarquía.

Todas las capas o comportamientos reciben información del sensor, trabajan en paralelo y generan salidas. Las salidas pueden ser comandos a los motores o señales que supriman otras capas. Para comenzar la ejecución de un comportamiento es obligatorio que el anterior haya finalizado, por ello cada uno debe tener una condición de salida, que devuelva el control al árbtitro para que decida con cuál comportamiento continuar.

El desarrollo se realiza utilizando la implementación existente en el API de leJOS NXJ para el modelo de subsunción. La misma consta de una clase Arbitrator y una interface Behavior, la cual modela los comportamientos. Tales comporamientos tienen los métodos:

- takeControl: Para definir si este comportamiento debería estar en control del robot.
- action: Código que se ejecuta una vez que este comportamiento toma el control.
- suppress: Código para cancelar inmediatamente al método action.

Se implementa una clase adicional denominada Minoru en la cual se inserta toda la información relativa al sensado, desde la calibración del dispositivo estereoscópico, generación e interpretación del mapa de disparidad. Mediante dicha información, se puede saber si existe colisión, información de entrada para los comportamientos Esquivar, Flanquear y Avanzar. En primer lugar, el robot se orienta al destino que es parámetro del sistema, posteriormente los comportamientos entran en competencia por el control.

- Esquivar: Toma el control si existe colisión, información disponible a partir de la clase Minoru. Devuelve el control una vez concretada la rotación determinada por la interpretación del mapa de disparidad.
- Flanquear: Actúa si hubo colisión, y hace que el robot transite la distancia necesaria para sortear el obstáculo. La misma está calculada según la distancia máxima a la que se detecta colisión más una tolerancia que es parámetro del sistema. Una vez finalizado dicho recorrido, el robot se reorienta al destino.

- Avanzar: Toma el control si el camino está despejado. Controla la distancia al punto destino, si la misma se encuentra por debajo de un umbral determinado, detiene el robot.

Capítulo 6

Experimentos y resultados

En esta sección se presentan los resultados de los experimentos de odometría y de trayectoria con obstáculos dispuestos de diferentes maneras, con el objetivo principal de demostrar que el robot se comporta según lo esperado, aproximándose hacia el punto destino realizando esquive de obstáculos. Al finalizar se realiza una discución acerca de los resultados obtenidos.

6.1. Pruebas de odometría

Para estimar la precisión de la odometría se hizo transitar al Lego® Mindstorms® NXT de dos maneras, con el camino libre y con un obstáculo a la mitad de un recorrido en una línea recta de 1,80 m de distancia. Como obstáculo se utilizó una caja de 29 cm de ancho, 24 cm de alto y 14 cm de profundidad. Según la información provista por la odometría el Lego® Mindstorms® NXT debía detenerse a 1,80 m una vez concretado dicho recorrido, punto que se utilizó para medir la distancia entre el punto al que debía llegar y el punto al cual llegó. Se realizaron tres pruebas de cada tipo de recorrido obteniéndose los resultados que se detallan en las Tablas 6.1 y 6.2.

Tabla 6.1: Error sin obstáculo.

Recorrido	Error en cm
1	3,1
2	3,2
3	4,5
Error promedio	3,6
Porcentaje de error	2,0 %

Tabla 6.2: Error con obstáculo.

Recorrido	Error en cm
1	5,1
2	5,2
3	6,3
Error promedio	5,53
Porcentaje de error	3,07 %

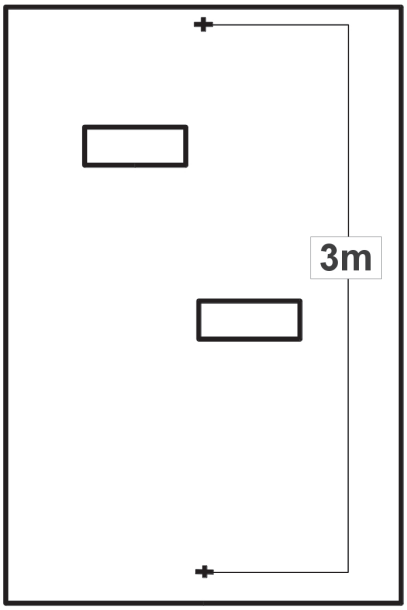
6.2. Pruebas de trayectoria

De acuerdo a las mediciones, el tiempo promedio desde que se capturan las imágenes hasta que se genera el mapa de disparidad con su posterior interpretación es de 3,1 segundos en el Raspberry Pi.

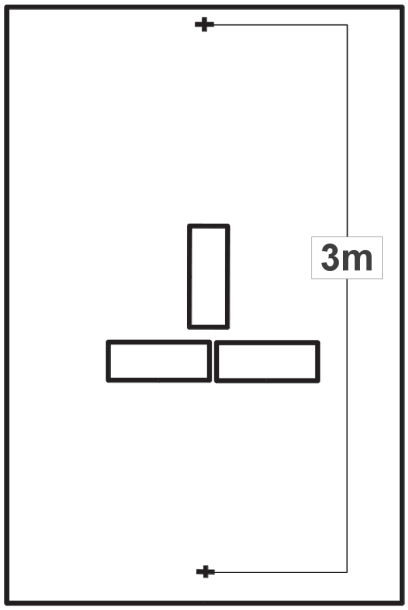
Según los valores obtenidos en la etapa de calibración, la distancia focal promedio es de 455,7 píxeles. Tal medida se puede utilizar para calcular la profundidad Z a partir de la disparidad utilizando la ecuación 2.8. Se sabe que la distancia de línea base del dispositivo estereoscópico es $T = 6$ cm, y que el algoritmo de correspondencia fija la coordenada de comienzo de búsqueda en 0 (mismas coordenadas en ambas visualizaciones) y el número de disparidades a buscar en 64. Estos parámetros proporcionan un mapa de disparidad que barre de los 42,7 cm de profundidad al "infinito".

Tales resultados sirven para especificar la velocidad máxima del robot y el umbral para el mapa de disparidad. Definiendo el umbral en 47 píxeles, se estarán visualizando puntos que van de los 42,7 cm a los 58,17 cm de profundidad (15,47 cm de diferencia). La velocidad del robot debe ser tal que sea posible sensar al menos una vez dentro de esa diferencia, para que no queden zonas sin medir. Como se vio, el tiempo promedio de sensado es de 3,1 segundos, por lo tanto la velocidad máxima del robot será de 15,47 cm/3,1 s, es decir 4,99 cm/s. Se comprobó que el mismo software en una Notebook Dell Core 2 Duo con 2 GB de RAM, demora en promedio 110 ms para llegar a la acción a partir de las imágenes sensadas.

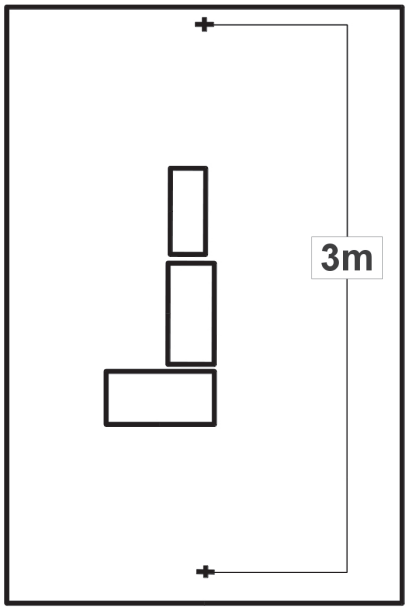
Se observa en las Figuras 6.2, 6.3, 6.4 y 6.5 capturas de tres recorridos (disponibles en http://www.youtube.com/), donde el robot debe llegar a un punto ubicado 3 m por delante del mismo. En cada ejemplo se colocaron cajas a manera de obstáculos en diferentes posiciones para demostrar la reacción del robot a los mismos de acuerdo a los modelos de la figura 6.1.



(a) Primer ejemplo



(b) Segundo ejemplo



(c) Tercer ejemplo

Figura 6.1: Diseños de trayectorias.

En todos los casos se comenzó el recorrido con el camino libre de obstáculos, por lo cual el comportamiento Avanzar tomó el control del robot.

En las Tablas 6.3, 6.4, 6.5 y 6.6, se exponen el comportamiento que controlaba el robot, la interpretación del mapa de disparidad y la acción a realizar en cada uno de las capturas.

Tabla 6.3: Primer ejemplo, primer tramo

Captura	Comportamiento	Interpretación del mapa	Acción
a	Avanzar	Sin colisión	Avanzar
b	“	“	“
c	Esquivar	Colisión en área central	Detenerse - Rotar a la izquierda
d	Flanquear	Sin colisión	Avanzar
e	Avanzar	“	Rotar a punto destino - Avanzar
f	“	“	Avanzar
g	“	“	“
h	Esquivar	Colisión en área central	Detenerse - Rotar a la derecha

Tabla 6.4: Primer ejemplo, segundo tramo

Captura	Comportamiento	Interpretación del mapa	Acción
a	Flanquear	Sin colisión	Avanzar
b	Avanzar	“	Rotar a punto destino - Avanzar
c	“	“	Avanzar
d	“	“	“
e	“	“	“
f	“	“	Llegó a destino - Detenerse

Tabla 6.5: Segundo ejemplo

Captura	Comportamiento	Interpretación del mapa	Acción
a	Avanzar	Sin colisión	Avanzar
b	“	“	“
c	Esquivar	Colisión en área central	Detenerse - Rotar a la izquierda
d	“	Colisión en área lat. der.	Rotar a izquierda - Avanzar
e	Flanquear	Sin colisión	Avanzar
f	Avanzar	“	Rotar a punto destino - Avanzar
g	“	“	Avanzar
h	“	“	Llegó a destino - Detenerse

Tabla 6.6: Tercer ejemplo

Captura	Comportamiento	Interpretación del mapa	Acción
a	Avanzar	Sin colisión	Avanzar
b	Esquivar	Colisión en área central	Detenerse - Rotar a la derecha
c	Flanquear	Sin colisión	Avanzar
d	Avanzar	Sin colisión	“
e	“	“	“
f	“	“	“
g	“	“	“
h	“	“	Llegó a destino - Detenerse

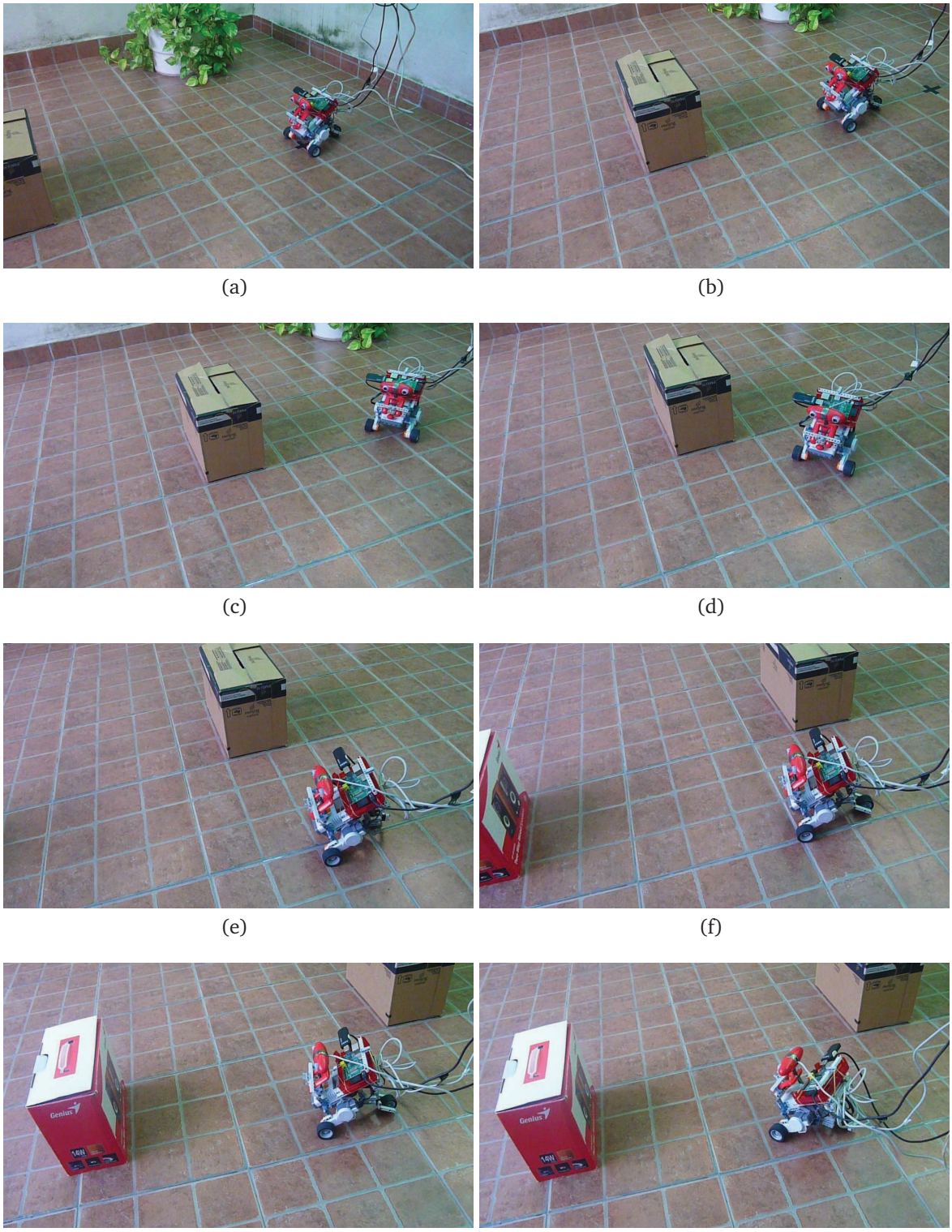
6.3. Discusión de resultados

En las pruebas de odometría se observó un porcentaje de error superior en el recorrido que se realizó con obstáculo. Esto se debe a que el robot debió realizar mayor cantidad de maniobras para llegar a destino. Tal detalle concuerda con lo expuesto sobre odometría, donde se mencionó el problema de propagación de errores producto de la sumatoria de movimientos que requiere dicha técnica.

Utilizando las seis pruebas de odometría para el cálculo se obtuvo un error promedio final de 2,53 %, el cual significó una distancia aproximada de 4,56 cm para el trayecto de 1,80 m.

En las pruebas de trayectoria el robot se comportó de acuerdo a lo esperado, con resultados satisfactorios. Fue necesario colocar los obstáculos a partir de una distancia superior a la distancia mínima desde la cual el dispositivo estereoscópico sensa según los parámetros establecidos en el algoritmo de búsqueda de correspondencias, esto es desde los 42,7 cm.

Teniendo en cuenta que para trayectos más largos el error de odometría se propagaría, se buscó minimizarlo. Esto se logró disminuyendo el umbral de distancia al punto destino para las pruebas de trayectoria, quedando fijo en 2 cm. A pesar de ello, se observa en las tres trayectorias que el robot no finaliza concretamente sobre el punto destino (centro de eje sobre la cruz negra en el piso), terminando finalmente con un error promedio de 1,5 %, significando una distancia aproximada de 4,6 cm para el trayecto de 3,00 m.



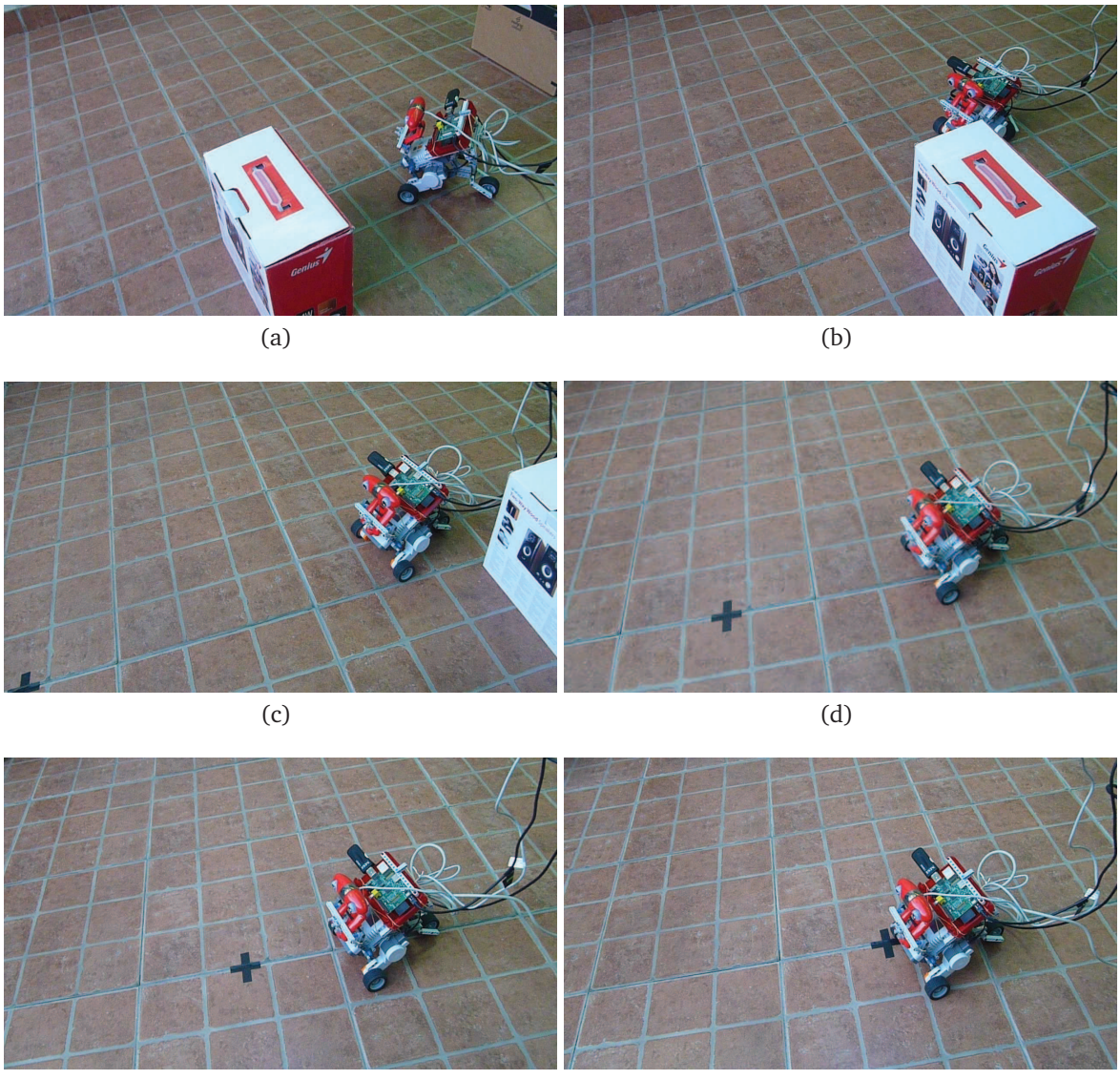
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figura 6.2: Primer ejemplo, primer tramo.

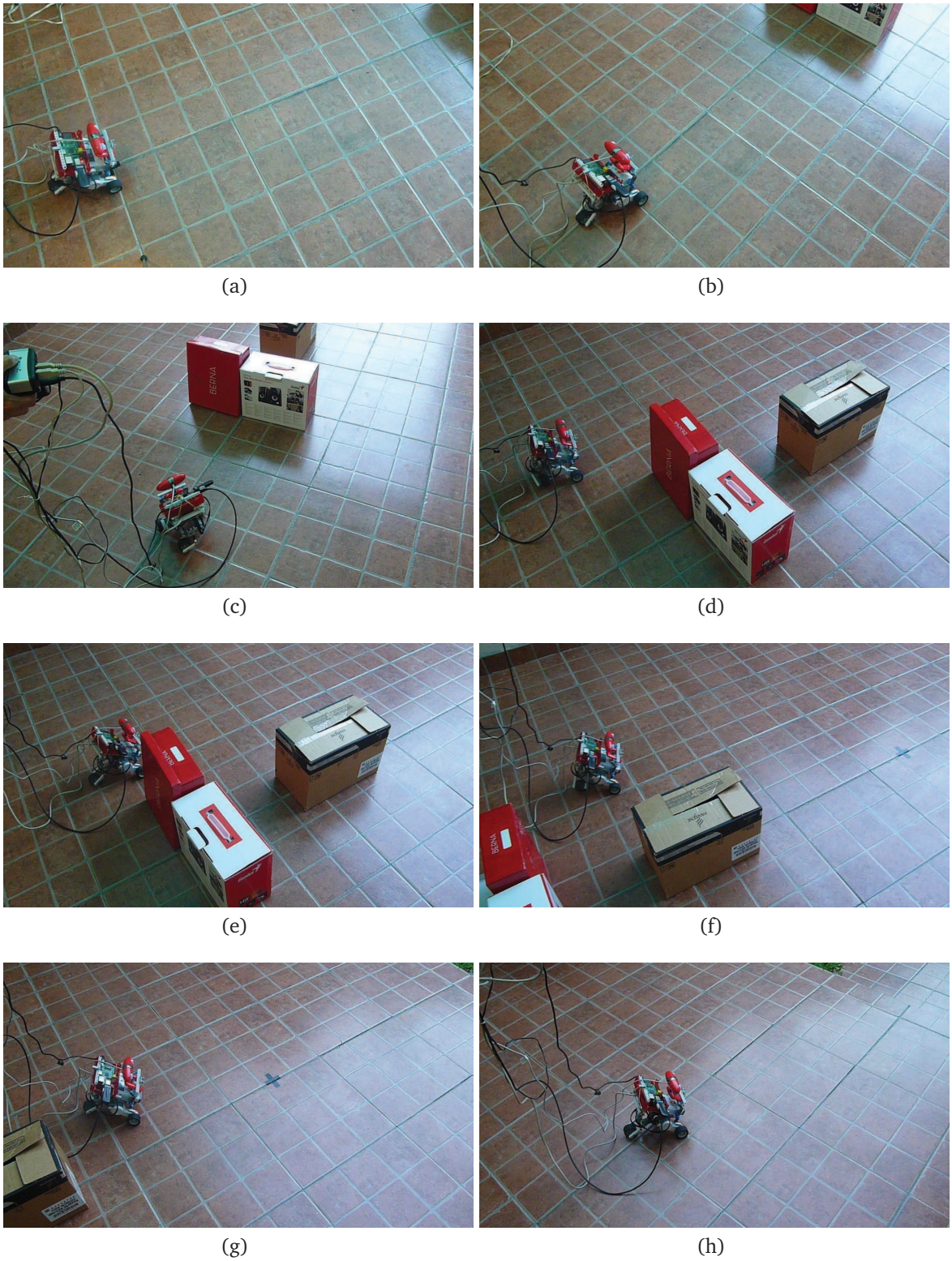


(a) (b)

(c) (d)

(e) (f)

Figura 6.3: Primer ejemplo, segundo tramo. Enlace al video: http://www.youtube.com/watch?v=pDb9A7_Op8Q.



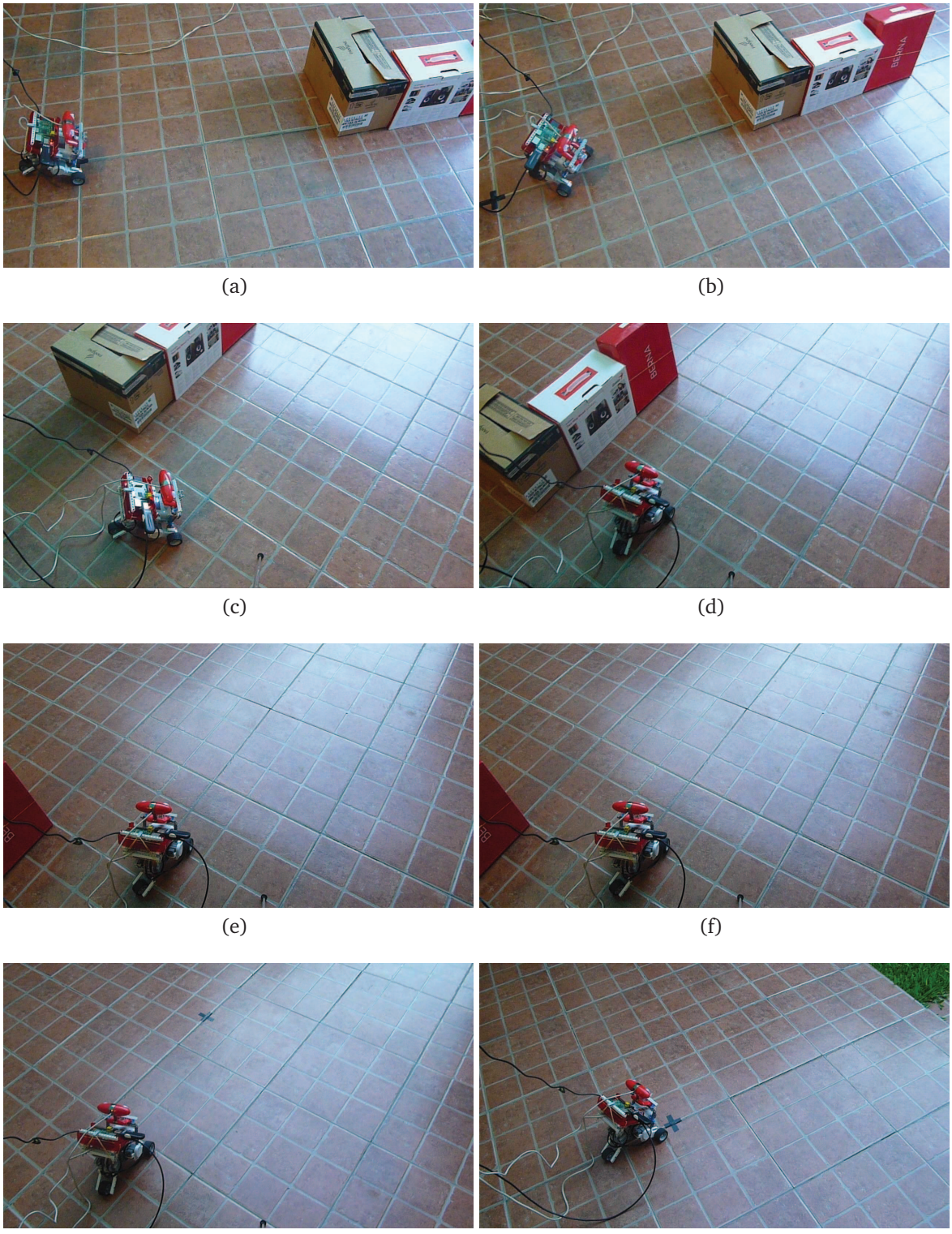
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figura 6.4: Segundo ejemplo. Enlace al video: http://www.youtube.com/watch?v=7ZvnDwADMVA.



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figura 6.5: Tercer ejemplo. Enlace al video: http://www.youtube.com/watch?v=lSnfdcqETmg.

Capítulo 7

Conclusiones y desarrollos futuros

7.1. Conclusiones

En este proyecto se implementó un sistema de navegación para Lego® Mindstorms® NXT para ejecutarse en el ordenador de placa reducida Raspberry Pi, con lo cual se alcanzó el objetivo principal del mismo.

Se cumplieron las etapas necesarias para alcanzar lo propuesto, desde la calibración del dispositivo estereoscópico para la remoción de distorsiones, la generación e interpretación de los mapas de disparidad para el esquive de obstáculos y el control del Lego® Mindstorms® NXT para llegar a un punto determinado utilizando información de odometría.

Dado que el software se ejecuta en una mini computadora, se tienen las restricciones lógicas dadas por su capacidad de procesamiento limitada. Por ello es necesario que el avance del Lego® Mindstorms® NXT sea a baja velocidad. Tal requerimiento permite que el sistema sea capaz de realizar el procesamiento total de las imágenes, desde la captura hasta la interpretación del mapa de disparidad, y con ello definir una orden para el robot a intervalos de tiempos apropiados para que el mismo transite sin colisionar al destino.

La utilización de la mini computadora Raspberry Pi en conjunto con la plataforma de software elegida representa un condicionante para el desempeño del robot. Esto se puede resolver de tres maneras: mediante hardware más potente, una solución empaquetada o mediante el uso de rutinas de software hechas a medida en el Raspberry Pi.

Utilizar hardware más potente (como la notebook mencionada en el proyecto), permitiría que los cálculos necesarios para la toma de decisiones se realicen en el orden

de los milisegundos, posibilitando que el robot se mueva a velocidades superiores. Si se desea realizar una solución comercial empaquetada (como el robot limpiador Roomba o similar) utilizando los algoritmos desarrollados en este proyecto, se deben tener en cuenta los costos de producción y facilidad de ensamblaje. Es en este caso donde la mini computadora Raspberry Pi lleva la ventaja, dado que su costo, tamaño y peso son significativamente menores [40].

La mejor opción a futuro sería el desarrollo de software a medida. Para ello sería necesario migrar la mayoría de código Java a C o C++, utilizando rutinas propias de conexión para la cámara web y el robot, y desarrollar algoritmos óptimos de cálculo para poder comparar el desempeño. Se podría por ejemplo, intentar sacar provecho de la GPU buscando la manera de realizar algunos de los cálculos en dicho procesador.

Comparando este proyecto con trabajos actuales de objetivos similares realizados en el país, se concluye que se ha logrado dar un paso adelante hacia la autonomía en robótica. Esto se debe a que se logró desarrollar un sistema capaz de ejecutarse en tiempo real en una computadora de capacidad reducida, cuyo tamaño y peso permiten acercarse significativamente a la implementación de una solución empaquetada. Además, se ha podido brindar un comportamiento de mayor complejidad que sólo el de esquivar obstáculos, tal como es el caso de lo logrado en uno de los trabajos observados [41, 42].

La precisión con la cual el robot llega al punto destino es dependiente de los errores de odometría, y está fuertemente condicionada por la correcta orientación del mismo al inicio del recorrido. Minimizando el umbral de distancia al destino es posible obtener mejores resultados. Se fijó dicho umbral en 2 cm con lo cual se alcanzó un error de 1,5 % en el recorrido de 3 m.

7.2. Desarrollos futuros

Se propone como progreso a este proyecto la búsqueda de la disminución de los costos computacionales implícitos en el procesamiento de las imágenes, a los efectos de permitir que la captura de éstas y en consecuencia las órdenes al robot, se produzcan con mayor frecuencia.

Ya que resultan inevitables los errores de posicionamiento debido a que se utiliza odometría, este sistema se puede mejorar utilizando la información de otros sensores disponibles para el Lego® Mindstorms® NXT, particularmente el acelerómetro (aceleraciones lineales) y el giróscopo (aceleraciones angulares). Con la fusión de los datos provenientes de las distintas fuentes, un filtro de Kalman sería capaz de obtener los parámetros óptimos para estimar el posicionamiento.

El sistema actualmente no optimiza el recorrido al punto final. La estrategia frente a un obstáculo es definir hacia dónde rotar en base al mapa de disparidad, sin tener en

cuenta la ubicación relativa del destino. Una mejora podría consistir en que al momento de definir el esquive se tenga en cuenta el destino para minimizar el recorrido.

Bibliografía

[1] A. Stelzer, H. Hirschmüller, and M. Görner. Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain. *The International Journal of Robotics Research*, 31(4):381–402, 2012.

[2] J. Ruiz de Garibay P. *Robótica: Estado del Arte*. PhD thesis, Tesis doctoral en Sistemas de la Información, Universidad de Deusto, 2006.

[3] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Press Syndicate of the University of Cambridge, 2004.

[4] L. Manucci. Reconstrucción 3D a partir de estereografías. Proyecto final de carrera, UNL, Facultad de Ingeniería y Ciencias Hídricas, Santa Fe, Argentina, 2010.

[5] D. Orozco and A. Felipe. *Sensores, métodos y técnicas para la navegación de robots móviles autónomos*. Universidad Pontificia Bolivariana, 2001.

[6] S. B. Goldberg, M. W. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 5, pages 5–2025. IEEE, 2002.

[7] M. Chen, Z. Cai, and Y. Wang. A method for mobile robot obstacle avoidance based on stereo vision. In *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, pages 94–98. IEEE, 2012.

[8] Z. Yong-guo, C. Wei, and L. Guang-liang. The navigation of mobile robot based on stereo vision. In *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference on*, pages 670–673. IEEE, 2012.

[9] S. Birchfeld. *An Introduction to Projective Geometry*. Stanford University, 1998.

[10] O. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. The MIT Press, 1993.

[11] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly, 2008.

[12] Seven S Beauchemin and Ruzena Bajcsy. Modelling and removing radial and tangential distortions in spherical lenses. In *Multi-Image Analysis*, pages 1–21. Springer, 2001.

[13] L. Ma, Y. Chen, and K. Moore. Rational radial distortion models of camera lenses with analytical solution for distortion correction. *International Journal of Information Acquisition*, 1(02):135–147, 2004.

[14] K. S. Choi, E. Y. Lam, and K. K. Y. Wong. Automatic source camera identification using the intrinsic lens radial distortion. *Optics express*, 14:11551–11565, 2006.

[15] J. G. Fryer and D. C. Brown. Photogrammetric engineering and remote sensing. *IAPR TC 7 Workshop on Analytical Methods in Remote Sensing for Geographic Information Systems*, 52(1):51–58, 1986.

[16] Image distortion. http://josedolz.jimdo.com/computer-vision-tutorials/image-distortion/. Fecha de acceso: 2013-09-12.

[17] Spatial calibration in depth. http://zone.ni.com/reference/en-XX/help/372916P-01/nivisionconcepts/spatial_calibration_indepth/. Fecha de acceso: 2013-09-12.

[18] Y. Zhao, W. Cheng, L. Jia, and S. MA. *The Obstacle Avoidance and Navigation based on Stereo Vision for Mobile Robot*. International Conference on Optoelectronics and Image Processing, 2010.

[19] L. Tang, C. Wu, and Z. Chen. Image dense matching based on region growth with adaptive window. *Pattern recognition letters*, 23(10):1169–1178, 2002.

[20] G. L. Foresti, C. Micheloni, and T. Ellis. Active video-based surveillance system: the low-level image and video processing techniques needed for implementation. *Signal Processing Magazine, IEEE*, 22(2):25–37, 2005.

[21] S. Negahdaripour, R. Prados, and R. Garcia. Planar homography: accuracy analysis and applications. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 1, pages I–1089. IEEE, 2005.

[22] G. Medioni and S. B. Kang. *Emerging topics in computer vision*. Prentice Hall PTR, 2004.

[23] M. Rufli, D. Scaramuzza, and R. Siegwart. Automatic detection of checkerboards on blurred and distorted images. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3121–3126. IEEE, 2008.

[24] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[25] S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[26] A. Bódis-Szomoru, T. Dabóczi, and Z. Fazekas. Calibration and sensitivity analysis of a stereo vision-based driver assistance system. *Chapter in Stereo Vision, In-Tech, ISBN 978-953-7619-22*, 2008.

[27] M. I. A. Lourakis. *A Brief Description of the Levenberg-Marquardt Algorithm Implemented*. Institute of Computer Science, Foundation for Research and Technology, 2005.

[28] A. Davidson. *Killer Game Programming in Java*. O'Reilly, 2009.

[29] A. Habib, A. Jarvis, G. Detchev, D. Stensaas, D. Moe, and J. Christopherson. Standards and specifications for the calibration and stability of amateur digital camaras for close-range mapping applications. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37, 2008.

[30] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos. Review of stereo vision algorithms: from software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[31] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[32] J. Braux-Zin. Toward real-time dense 3d reconstruction using stereo vision. Master's thesis, School of Computer Science and Communication, 2011.

[33] A. Kuhl. Comparison of stereo matching algorithms for mobile robots. Technical report, The University of Western Australia, Faculty of Engineering, Computing and Mathematics, Australia, 2005.

[34] S. Patil, J. Simon Nadar, J. Gada, S. Motghare, and S. S. Nair. Comparison of various stereo vision cost aggregation methods. *International Journal of Engineering and Innovative Technology*, 2, 2013.

[35] S. Intille and A. Bobick. *Disparity-space images and large occlusion stereo*. Springer, 1994.

[36] C. L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(7):675–684, 2000.

[37] L. Nalpantidis, I. Kostavelis, and A. Gasteratos. Stereovision-based algorithm for obstacle avoidance. In *Intelligent Robotics and Applications*, pages 195–204. Springer, 2009.

[38] J. Borenstein. Experimental results from internal odometry error correction with the omnimate mobile robot. *Robotics and Automation, IEEE Transactions on*, 14(6):963–969, 1998.

[39] W. Xie, J. Ma, M. Yang, and Q. Zhang. Research on classification of intelligent robotic architecture. *Journal of Computers*, 7(2):450–457, 2012.

[40] Robot roomba vacuum cleaning robot. http://www.irobot.com/us/learn/home/roomba.aspx. Fecha de acceso: 2014-04-09.

[41] Pire T. Evasión de obstáculos en tiempo real usando visión estéreo. Technical report, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, UBA, 2012.

[42] C. D. Ortega and F. E. Moyano. Técnicas de implementación de visión estereoscópica en robótica. *EST - Jornadas Argentinas de Informática e Investigación Operativa*, pages 439–451, 2013.

Apéndice - Anteproyecto

A.1. Introducción

Desde sus inicios los desarrollos de la visión por computadora han estado inspirados en el estudio del sistema visual humano. Representan una alternativa para la percepción del entorno para máquinas y permiten realizar mediciones sin contacto del mismo[1]. Este tipo de desarrollos se está volviendo más popular en aplicaciones robóticas, donde se utilizan para el control basado en visión de robots móviles. Dicha visión puede dividirse en dos categorías de diseño: monocular y estéreo, cada uno con sus ventajas y desventajas, siendo la ventaja principal del diseño estéreo la posibilidad de generar información tridimensional.

Actualmente, es en estas aplicaciones robóticas móviles donde se está poniendo mayor énfasis, en gran medida debido a la necesidad de adaptación robótica a un entorno flexible, como por ejemplo en el caso del transporte de materiales entre cualesquiera de los puntos de una instalación fabril o en un lugar que resulte inhóspito para las personas, como ocurre luego de un desastre nuclear [1, 2].

La visión estéreo se consigue con la observación de la misma escena desde dos perspectivas diferentes, lo cual provee un medio para determinar la forma y su posición tridimensional. Esto es posible mediante la búsqueda de correspondencias de puntos en ambas vistas (es decir, capturar el mismo punto en las dos visualizaciones), lo que permite calcular diferencias de coordenadas del punto en cada una de ellas. Dicha diferencia se denomina disparidad y al conjunto total de diferencias mapa de disparidad, con el cual se pueden inferir distancias relativas. Para que esta búsqueda de correspondencias gane en precisión y eficiencia, se debe: por un lado obtener lentes libres de distorsión y por el otro hacer uso de la geometría asociada al sistema estéreo, a los fines de encontrar las relaciones espaciales entre ambos lentes. Estas relaciones permiten la rectificación de imágenes, lo cual posibilita que las búsquedas sean unidimensionales [3].

Para el movimiento del robot los mecanismos constan en general de un sistema de

[1] www.tecnicaenlaboratorios.com/Nikon/Info_medicion_sin_contacto.htm

locomoción que puede basarse en tecnologías que se vienen utilizando desde hace siglos para el transporte, pudiendo ser ruedas, cadenas o hélices. Otras tecnologías buscan imitar a la naturaleza en la resolución de algún problema (biomímesis), como por ejemplo un sistema de locomoción que utilice piernas o que repte como una serpiente [2].

El sistema de navegación y esquive de obstáculos consiste en un algoritmo que se basa en la interpretación tridimensional de la escena visualizada. Debe contar con un módulo de navegación que sea capaz de tomar tal interpretación y responder en tiempo real, para cumplir con las restricciones temporales que permitan la interacción adecuada con el entorno.

A.2. Justificación

Existen diversas tecnologías para percibir el entorno, cada una de ellas con sus ventajas y desventajas. Se pueden usar sensores ópticos tales como fotorresistencias, fotodiodos o fototransistores. Los fototransistores pueden usarse como sensores infrarrojos de tipo On-Off, es decir pueden indicar solamente si el obstáculo está o no presente, para obtener información sobre distancia se debe recurrir a mecanismos más complejos [5].

Otro tipo de sensores muy utilizados son los de ultrasonido, los cuales hacen uso de las propiedades del sonido y del tiempo transcurrido entre emisión y recepción de la señal. Cuando se utiliza el mismo sensor para emisión y recepción tiene la desventaja de que aparece una “zona muerta”, esto es la distancia mínima a la que no se puede detectar obstáculos, la cual se determina por el tiempo que necesita el sensor para pasar del estado de emisión al de recepción. Otra desventaja que surge de la utilización de estos sensores es el denominado “eco falso”, provocado por el rebote de la señal en múltiples superficies antes de llegar al sensor, lo que implica que la estimación de la distancia al obstáculo sea mayor a la real [2].

Mediante el uso de un sensor de imágenes se podrían evitar las desventajas mencionadas. Además se contaría con la posibilidad de obtener otro tipo de información, no disponible con otros sensores, como por ejemplo color, reconocimiento de caracteres, objetos, etc. Por ello, este trabajo se realiza con la intención de brindar una solución al problema de navegación autónoma, basado en un sensor de imágenes [5].

Otra ventaja que presenta el uso de múltiples sensores de este tipo, es que permite obtener información tridimensional del entorno, información valiosa cuando la complejidad de la escena aumenta [6].

También debe notarse que desarrollar un sistema basado en visión artificial representa una evolución para la robótica, ya que dicho sistema está inspirado en el sentido más utilizado por la mayoría de los animales, la visión. En este trabajo se utilizará un sensor de tipo estereoscópico. Este tipo de sensor está diseñado para la imitación de

la forma en que la naturaleza resuelve un problema de diseño, ya que se está creando un sistema basado en visión condicionada por la disposición frontal de los ojos, como tenemos los humanos.

La utilización de sistemas de visión estereoscópicos en robótica móvil mejora en gran medida el desempeño gracias al uso de información tridimensional, particularmente útil en situaciones que involucran el reconocimiento de escenas no familiares. Considerando un sistema de navegación que utiliza visión estéreo, deben resolverse tres problemas: el primero es cómo obtener la información tridimensional de la escena con lo que ello implica (contrarrestar la distorsión inherente en toda cámara y conocer la disposición física del dispositivo de visión), el segundo es tener un mecanismo que pueda interpretar la información provista por tal sensor y el tercero es la estimación del movimiento del robot mediante un módulo de navegación. Dicho módulo de navegación debe lograr resolver el camino a seguir en tiempo real, para permitir que dicha navegación se realice sin riesgos para el entorno del robot y para sí mismo. Tal restricción temporal implica que el sistema sea predecible en cuanto al tiempo que demora en brindar una respuesta. El algoritmo de visión estéreo puede implementarse mediante hardware específico o en una computadora [1, 7, 8].

Este trabajo tiene como finalidad dotar a un robot de autonomía. Con ella, lo que se busca es reducir la intervención humana al mínimo, lo que representa un paso más para acercar la robótica a nuestra vida diaria. Hasta ahora los avances conseguidos contribuyeron a los primeros intentos de utilizar robots móviles en aplicaciones de la vida real, como asistentes personales para personas con discapacidades o ancianas por ejemplo, mascotas robóticas, entre otros. Un sistema de este tipo se puede aplicar en diversas situaciones, por ejemplo para un dispositivo explorador donde se requiera navegación autónoma sin riesgos para el mismo, con la ventaja agregada de que utiliza solamente un sensor para lograr el cometido, lo que deviene en menores costos para su implementación.

Cuando las innovaciones son como en este caso inspiradas en la naturaleza se acuñan bajo los términos “bioimitación” o “biomímesis”[2]. El argumento principal para apelar a estas soluciones es que en millones de años de evolución de la vida en la Tierra, la naturaleza seguramente ya encontró soluciones de diseño óptimas para enfrentar los desafíos a los que se enfrentan los humanos. Estas soluciones además conducen a diseños eficientes, prácticos y sostenibles desde un punto de vista ingenieril, y tienen además el potencial para descubrir nuevas tecnologías, materiales y procesos.

Se establece como interés central de este trabajo realizar una aplicación en el marco de la biomímesis. Dicha aplicación es posible a través del uso de un sistema de visión estéreo calibrado y rectificado como herramienta que genera información útil para un algoritmo que posibilita evitar obstáculos.

[2]http://www.athanor.es/reportajes/bioimitacion

A.3. Objetivos

General

Desarrollar un sistema de navegación basado en visión estéreo.

Específicos

- Desarrollar el módulo para calibración automática de cámaras utilizando un patrón conocido.
- Desarrollar el algoritmo para procesar imágenes ya calibradas, con las que se pueda generar los mapas de disparidades.
- Diseñar y desarrollar el módulo para navegación basado en procesamiento estereoscópico de imágenes.
- Implementar el módulo desarrollado en una mini computadora y dotar a un robot de navegación autónoma en tiempo real.
- Evaluar y analizar el desempeño del sistema desarrollado.

A.4. Alcances

Se desarrollará un software capaz de realizar la calibración individual de cada uno de los lentes de un dispositivo estereoscópico, a los efectos de eliminar las distorsiones que provoquen. La información de esta calibración se utilizará para realizar la calibración estéreo y rectificación de las imágenes que se capturen. El tipo de calibración será fotogramétrica mediante la utilización de un objeto plano. Como objeto de calibración se utilizará un tablero de ajedrez.

Por cada par de imágenes tomado desde el dispositivo ya calibrado y rectificado, se realizará una búsqueda de correspondencias de puntos para poder calcular diferencias de coordenadas y generar así un mapa de disparidad. Cada mapa de disparidad será interpretado a los efectos de detectar y esquivar obstáculos.

El Lego® Mindstorms® deberá aproximarse a un punto relativo al comienzo del recorrido. El desplazamiento se llevará a cabo en una escena estática con terreno plano y condiciones de iluminación controladas.

La computadora Raspberry Pi donde se instala el sistema debe estar conectada a la energía eléctrica.

A.5. Metodología

Investigación bibliográfica

En esta etapa, se realizará una recopilación y estudio exhaustivo de la bibliografía pertinente, que sirva al entendimiento del modelo matemático que describe la relación entre un punto 3D y su proyección en el plano de imagen.

Posteriormente, se investigará acerca de la geometría asociada a la visión estereoscópica denominada geometría epipolar, la cual provee el modelo matemático para describir la visión estéreo.

Calibración de cámaras

Se investigará la teoría asociada a la calibración de cámaras, para la eliminación de toda distorsión provocada indefectiblemente por el uso de lentes.

Cuando esta tarea esté concluída, se implementará el algoritmo que haga posible la calibración individual de cada uno de los lentes, así también como la calibración estéreo, necesaria para definir relaciones espaciales entre los mismos. Esta calibración estéreo hará posible la rectificación de imágenes basada en la geometría epipolar.

Algoritmo de correspondencias y módulo de navegación

Adquirir nociones sobre los algoritmos existentes para la búsqueda de correspondencias, e implementar alguno de ellos para generar el mapa de disparidad. Dicho mapa, servirá de entrada al algoritmo para la navegación, el cual será el responsable de emitir órdenes al Lego® Mindstorms®. Para el envío de órdenes se implementará un módulo de comunicación basado en el firmware LeJOS.

El software desarrollado se implementará en una mini computadora de bajo costo denominada Raspberry Pi, la cual estará conectada tanto al Lego® Mindstorms® como al dispositivo estereoscópico.

Documentación

Se producirán reuniones periódicas con el director de tesis para evaluar la evolución y retroalimentar el avance. Se creará la documentación que sirva como marco para el entendimiento de este trabajo. Dicha documentación será el Informe Final y se irá realizando de manera simultánea con el desarrollo del proyecto.

A.6. Plan de tareas

El cálculo se realiza con una base de 2 horas diarias de trabajo de lunes a sábado, excepto durante la realización de la última actividad de cada etapa, donde se adicionará 1 hora más a los efectos de cumplir con la redacción de los informes pertenecientes a los puntos de control.

La escritura del informe final, será simultánea a la tercer etapa. Para ello se agregará una hora diaria desde el comienzo de la misma (Figura A.1).

1 - Investigación bibliográfica (84 hs).

1.1 Modelos de cámara y transformaciones proyectivas (12 hs).

1.2 Distorsiones (12 hs).

1.3 Visión Estéreo (24 hs).

1.4 Estudio sobre geometría Epipolar (24 hs).

1.5 Escritura del punto de control (12 hs).

Punto de control: Informe sobre bibliografía recopilada.

2 - Calibración de cámaras (108 hs)

2.1 Transformación mediante homografía (12 hs).

2.2 Calibración mediante objeto conocido (12 hs).

2.3 Estudio sobre calibración estéreo (12 hs).

2.4 Estudio sobre rectificación estéreo (12 hs).

2.5 Revisión de los lenguajes C++ y Java (12 hs).

2.6 Estudio de la librería OpenCV y la interface JavaCV (12 hs).

2.7 Implementación del módulo de calibración (24 hs).

2.8 Escritura del punto de control (12 hs).

Punto de control: Informe sobre lo investigado y programación del módulo de calibración.

3 - Algoritmo de correspondencias y módulo de navegación (120 hs).

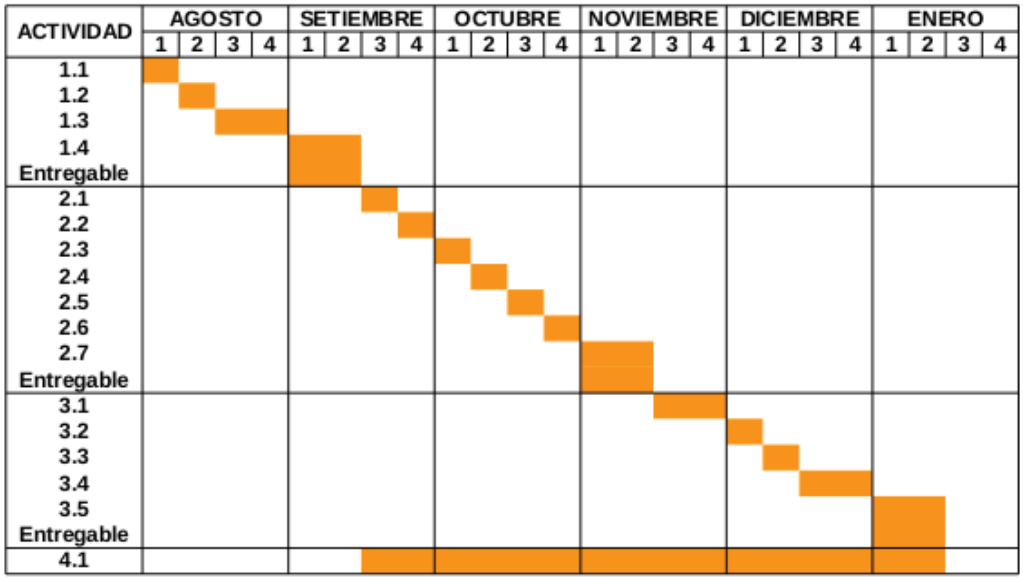


Figura A.1: Planificación.

3.1 Conocimiento sobre algoritmos de correspondencia, implementación (24 hs).

3.2 Instalación de sistema operativo y librerías en el Raspberry Pi (12 hs).

3.3 Estudio del firmware LeJOS (12 hs).

3.4 Implementación del módulo de navegación basado en el mapa de disparidad (24 hs).

3.5 Ajustes del módulo de navegación basado en los resultados (24 hs).

3.6 Escritura del punto de control (12 hs).

Punto de control: Informe sobre rectificación, algoritmo de correspondencia seleccionado y descripción del módulo de navegación. Dicho punto de control se entregará como parte del informe final.

4 - Documentación

4.1 Escritura del Informe final (96 hs).

Punto de control: Informe Final.

A.7. Riesgos y estrategias de mitigación de los mismos

1. La capacidad de procesamiento del Raspberry Pi no es suficiente para satisfacer los requerimientos de ejecución del software en tiempo real.
   Impacto: Medio-Alto.
   Mitigación: implementación del software en PC o NoteBook, desde donde se puedan enviar órdenes al Lego® Mindstorms® NXT vía Bluetooth.

2. La webcam estéreo provista se rompe o no se puede utilizar por algún otro motivo.
   Impacto: Bajo.
   Mitigación: utilización de alguna de las otras dos webcams estéreo adquiridas por la facultad.

3. Imposibilidad de utilizar el robot Lego® Mindstorms® NXT por rotura u otro motivo.
   Impacto: Medio-Alto.
   Mitigación: utilización de algún software para simulación del recorrido del robot.

4. Las últimas versiones de librerías a utilizar, no proporcionan el resultado esperado.
   Impacto: Bajo-Medio.
   Mitigación: utilización de versiones de librería más antiguas.

A.8. Recursos necesarios y disponibles para su realización

Al comienzo del proyecto se cuentan con todos los recursos, no es necesaria la adquisición de ninguno.

- Material bibliográfico.
- PC o notebook con el siguiente software:
  - Sistema operativo Linux Debian Wheezy.
  - Eclipse como IDE para el desarrollo.
  - Librería para el procesamiento de imágenes OpenCV.
  - Entorno de desarrollo para C++ y Java en su versión 8.
  - Librería JavaCV para embeber código de OpenCV desde JAVA.
  - Librería leJOS.
- Acceso a internet.
- Webcam estéreo.

- Lego® Mindstorms® NXT.
- Mini ordenador Raspberry Pi.

A.9. Presupuesto para su ejecución

Costos indirectos:

PC:

Micro Intel Core I3-3220 -3.30ghz	$932,50
Mother skt 1155 Gigabyte GA-H61M-S1	$411,70
Memoria Ddr3 4Gb 1333Mhz Pc3 10600 Kingston	$293,30
HD 1Tb Disco Rigido WD Caviar Blue	$590,60
Gabinete Kit Sentey 450W-Office (CS1-1399)	$488,70
Monitor LG E2240S	$1530

Otros:

Internet 1MB	$165
Impresora Samsumg ML1640	$380
Software	$0
Resma A4	$30
Toner	$520
Artículos de librería varios	$50

Total de costos indirectos: $5391,8

Costos directos:

Minoru 3D Webcam	$210,14
Lego® Mindstorms®	$6900
Raspberry Pi	$550

Costos directos por actividad:

TAREA	ITEM	HORAS	$
1.1	RRHH	12	840
1.2	RRHH	12	840
1.3	RRHH	24	1680
1.4	RRHH	24	1680
Entregable	RRHH	12	840
2.1	RRHH	12	840
2.2	RRHH	12	840
2.3	RRHH	12	840
2.4	RRHH	12	840
2.5	RRHH	12	840
2.6	RRHH	12	840
2.7	RRHH	24	1680
Entregable	RRHH	12	840
3.1	RRHH	24	1680
3.2	RRHH	12	840
3.3	RRHH	12	840
3.4	RRHH	24	1680
3.5	RRHH	24	1680
Entregable	RRHH	12	840
4.1	RRHH	108	7560

Total de costos directos: $35380,14

Costo total del proyecto: $40771,94