UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DETECCIÓN ROBUSTA DE OBJETOS EN ROBOTS HUMANOIDES

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN
CIENCIAS DE LA INGENIERÍA,
MENCIÓN INGENIERÍA ELÉCTRICA

JOSÉ MIGUEL YÁÑEZ ARANCIBIA

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
RODRIGO PALMA AMESTOY
PABLO GUERRERO PÉREZ

SANTIAGO DE CHILE
MARZO 2013

RESUMEN DE LA TESIS
PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN INGENIERÍA ELÉCTRICA.
POR: JOSÉ MIGUEL YÁÑEZ ARANCIBIA.
FECHA: MARZO DE 2013.
PROF. GUÍA: DR. JAVIER RUIZ DEL SOLAR.

En este trabajo se presenta un sistema de detección de objetos para robots móviles basado en la información visual entregada por las cámaras del robot y el estado de los *encoders* de cada articulación. Debido a la baja capacidad de procesamiento que poseen los computadores de este tipo de robots, no es posible utilizar técnicas clásicas de visión computacional para la detección de objetos. Por ésto, esta tesis propone un sistema de percepción alternativo, de bajo consumo computacional, de alto desempeño y robusto. El sistema desarrollado no sólo detecta objetos sino que también les calcula: la pose respecto al sistema de referencia del robot y el error de la medición dada por una matriz de covarianza. También identifica el objeto detectado en caso de tratarse de objetos repetidos en el entorno. El ambiente de trabajo es el fútbol robótico, por lo que los objetos de interés a detectar corresponden a: los arcos, las líneas de la cancha y sus respectivas intersecciones. Dado que éstos objetos poseen una geometría regular, los sistemas de percepción se basan en el ajuste de modelos de líneas rectas que luego son considerados posibles bordes de los objetos. Los puntos que conforman las líneas de borde son determinados a partir de la información de color para el caso de arcos, e información de altos gradiente de intensidad para el caso de las líneas. Luego una serie de reglas dispuestas en cascada determinan si los modelos encontrados corresponden a objetos en la imagen. La detección de objetos también hace uso de la información del estado de los *encoders* de los motores de cada articulación para eliminar falsos positivos y para calcular la pose de los objetos respecto al sistema de referencia del robot. Éste cálculo se realiza mediante técnicas de geometría proyectiva y hace uso de la información sobre el estado de la cámara, el cual está dado por una matriz homogénea calculada a partir de la información de los *encoders*. Dado que el estado de la cámara está sometido a condiciones de ruido e imperfecciones de fabricación del robot, se ha diseñado un sistema de calibración que mejora notablemente la exactitud del cálculo de la pose de los objetos detectados. Las detecciones son acompañadas de una matriz de covarianza generada a partir de funciones determinadas mediante análisis de información estadística. De esta forma las detecciones pueden ser utilizadas como observaciones directas para estimadores de estados basados en filtros de Kalman extendido (por ejemplo, el proceso de auto-localización). Con esta información el comportamiento del filtro de Kalman es más natural y menos susceptible a oscilaciones dadas por detecciones con poses ruidosas (como por ejemplo, cuando el objeto se encuentra a grandes distancias). Dado que los objetos líneas, esquinas y t-líneas son ambiguos, es decir, no se pueden identificar utilizando sólo información visual, se ha diseñado un sistema de identificación basado en la auto-localización del robot. El sistema funciona con una tasa de detección alta (93.8 % para el caso de los arcos) y funciona a una velocidad de aproximadamente 10 cuadros por segundo, lo cual satisface lo esperado. El sistema funciona en un robot móvil humanoide que posee un procesador AMD Geode de 500MHz y con 256MB de memoria RAM.

Agradecimientos

Tesis parcialmente financiada por Proyecto Fondecyt 1061158.

Tabla de contenido

Índice de figuras

Capítulo 1

Introducción

El presente trabajo se desarrolla en el marco de las actividades realizadas por el equipo de fútbol robótico del Laboratorio de Robótica del Departamento de Ingeniería Eléctrica de la Universidad de Chile.

1.1. Motivación

Todo robot móvil necesita de información acerca de su entorno para desenvolverse adecuadamente y llevar a cabo alguna tarea asignada. Esta información puede provenir de diversas fuentes sensoriales como radares, lásers, cámaras, etc. y puede ser utilizada directamente por el robot para: (i) interactuar con los objetos presentes en el entorno de acuerdo a su tarea, (ii) identificar la escena y así determinar el tipo de entorno y (iii) recoger información sobre objetos estáticos para poder auto-localizarse espacialmente en el entorno. Es por esto que la información del entorno juega un rol sumamente importante para un robot móvil.

Una de las fuentes de información del entorno más completa y utilizada por robots móviles, y en especial robots humanoides, es la información visual proveniente de imágenes capturadas por sus cámaras. Por lo general, y en lo que concierne a esta tesis, la información visual del entorno consiste en: (i) la detección de objetos de interés, (ii) el cálculo de la posición y orientación del objeto detectado y (iii) todos los procesos adicionales que permiten realizar este procedimiento de manera robusta.

La velocidad y habilidad física de un futbolista humano resultan características determinantes en su rendimiento al momento de jugar y hacer goles. Esta misma idea se aplica a un robot que juega fútbol. Para asegurar que el robot sea veloz y tenga un buen desempeño en general, es indispensable contar un sistema de control lo suficientemente robusto y veloz. A su vez, el sistema de control necesita un sistema de visión que le entregue información acerca del entorno de manera robusta, rápida (a más de 10 imágenes por segundo) y con bajas tasas de error.

Los sistemas de visión robótica, y en especial los sistemas de visión de robots bípedos, poseen una gran limitante al momento de implementar cualquier tipo de algoritmo de detección visual, y es que éstos poseen computadores embebidos con baja capacidad de

procesamiento. Éste es el principal desafío al momento de diseñar un sistema de visión robótica.

1.2. Visión robótica

En general, los robots móviles humanoides poseen una capacidad de procesamiento bastante limitada en comparación a la de los computadores tradicionales. Esto se debe a las consideraciones de diseño que se tienen al fabricar este tipo de robots, las cuales principalmente corresponden a: disponibilidad de espacio (tamaño reducido) y bajo consumo energético. Es usual que los robots móviles humanoides cuenten con un computador embebido de tamaño pequeño y con especificaciones de hardware limitadas.

La detección de objetos puede llevarse a cabo mediante técnicas clásicas de visión computacional, entre éstas se encuentran: (i) descriptores SIFT [12], (ii) clasificadores en cascada [24], etc. las cuales, si bien poseen un buen desempeño, requieren de grandes capacidades de procesamiento por lo que no es factible su uso en robots móviles humanoides.

Por lo anterior surge la necesidad de implementar sistemas de detección visual específicos para cada objeto, de bajo costo computacional y de alto desempeño. Éstos sistemas, al igual que cualquier otro sistema de visión, deben ser capaces de lidiar con los típicos problemas visuales tales como cambios de escala, oclusiones parciales, cambios de luminosidad, etc. En términos de desempeño deben lograr velocidades de procesamiento de aproximadamente 10 imágenes por segundo como mínimo para asegurar reactividad al robot.

En la literatura existen básicamente dos tipos de visión robótica que cumplen estos requerimientos de velocidad: (i) basada en color y (ii) basada en gradientes de intensidad. Por lo general, cuando la visión está basada en color, ésta requiere de una etapa de clasificación de los píxeles en colores preestablecidos mediante algún entrenamiento. Cuando la visión está basada en gradientes se buscan puntos de alta diferencia de intensidad y se procesan de tal forma para evaluar si se ajustan al modelo de algún objeto de interés en particular. Esto último no requiere etapa de calibración y por lo tanto es invariante a cambios de iluminación. Cabe mencionar que no existen impedimentos técnicos para que un sistema de visión de robótica pueda estar compuesto por un sub-sistema basado en color y otro basado en gradientes de alta intensidad.

Podemos decir que la detección de objetos es un problema altamente complejo principalmente por: i) la variabilidad en el proceso de adquisición de la imagen (oclusiones, escalas y otras transformaciones, variabilidad en la iluminación capturada por la cámara, entre otras), ii) variabilidad en el fondo (ruido en diferentes niveles, variabilidad en la iluminación emitida en el entorno, entre otros) y iii) restricciones de tiempo de procesamiento, debido especialmente a los límites de los computadores embebidos en los sistemas robóticos actuales. Adicionalmente, en muchos escenarios donde se desenvuelve un robot móvil, resultaría conveniente el uso de los *objetos indistinguibles*[a] que puedan estar presentes, específicamente para mantener una mejor retroalimentación visual de los sistemas de auto-localización del robot, debido a que este tipo de objetos puede aparecer en forma más

[a]En visión robótica corresponden a objetos del ambiente cuyas características son idénticas entre ellos y que pueden ser detectados por el sistema de percepción de un robot móvil

frecuente que los objetos únicos. Esto agrega una complejidad adicional, que es la necesidad de *identificación de objetos indistinguibles* para calcular correspondencias entre los objetos detectados y los objetos de un mapa y así tener una tasa adecuada de de observaciones que permitan mantener una auto-localización con niveles de error adecuados.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar e implementar sistemas de detección visual de objetos de alto desempeño y robustos para robots humanoides, con aplicación en fútbol robótico.

1.3.2. Objetivos específicos

- Desarrollar e implementar módulos de detección de objetos de interés basados en las características propias de cada objeto. Específicamente, éstos objetos corresponden a arcos, líneas y sus respectivas intersecciones (e.g. esquinas).
- Calcular de la manera más exacta posible la posición y ángulo de cada objeto respecto al sistema de coordenadas del robot.
- Acompañar a cada detección de una medida estimada de incertidumbre, la que ayude directamente a mejorar el desempeño del sistema de estimación de estados y de la auto-localización.
- Identificar correctamente objetos ambiguos.

1.4. Hipótesis

Es factible desarrollar sistemas de percepción visual de objetos de alto desempeño para un robot móvil humanoide. Éstos sistemas deben tener una velocidad de funcionamiento tal que garantice un alto grado de reactividad por parte del robot, y considerando a la vez, que éste posee una limitada capacidad de procesamiento.

1.5. Estructura del documento

El documento se estructura de la siguiente manera: El capítulo 3, Plataforma de Trabajo, describe el fútbol robótico y el robot que se utiliza para realizar pruebas. También se describe la plataforma de control del robot utilizada UChileLib y el principio básico de funcionamiento del sistema de visión. En el capítulo 2, Trabajo Relacionado, se analizan las principales características que debe poseer un sistema de visión robótica en el ambiente del fútbol robótico. Se analizan otros trabajos de la literatura que intentan resolver el mismo problema o similares. El capítulo 4, Sistema de detección robusta de objetos, consiste en la descripción detallada del trabajo llevado a cabo. El capítulo 5, Resultados, expone los

resultados de lo descrito en el capítulo anterior además de las conclusiones y bibliografía utilizada en esta tesis.

Capítulo 2

Plataforma de trabajo

En éste capítulo se describen de manera general el fútbol robótico, el robot humanoide que se utiliza, la librería de control del robot y la herramienta de desarrollo. También se describe el trabajo ya existente que entregará información de bajo nivel a los sistemas desarrollados en esta tesis.

2.1. Fútbol robótico: escenario SPL RoboCup

Incluso para los seres humanos, el jugar fútbol es una actividad compleja que requiere diversas habilidades de parte de los jugadores. Éstos deben: ser capaces de estimar las posiciones de los objetos dentro de la cancha, tomar decisiones acertadas según el estado del juego, estimar la trayectoria de la pelota y ser capaces de correr o caminar en la dirección adecuada. Esto hace del fútbol una actividad muy interesante y desafiante si se quiere llevar a cabo por sistemas robóticos, particularmente por robots humanoides. Bajo este marco, el fútbol robótico ha sido un ambiente fructífero para el desarrollo de nuevos algoritmos y sistemas los cuales son extensibles a otros problemas de robótica de mayor o menor complejidad. Todo lo anterior hace de RoboCup, un escenario de investigación muy interesante para el desarrollo de la robótica móvil en todas sus áreas.

RoboCup es una organización internacional dedicada a fomentar el desarrollo de la robótica y la inteligencia artificial [16]. Para esto, RoboCup organiza anualmente una competencia internacional de robótica que incluye diversas categorías, que se diferencian tanto en el tipo de robots a utilizar como en las actividades a desarrollar. Sin embargo, la mayoría de las categorías tienen como objetivo el jugar fútbol con un equipo de robots y varias de ellas utilizan robots humanoides de diferentes tamaños y características. Una de estas categorías corresponde a la *Standart Platform Leage* (SPL).

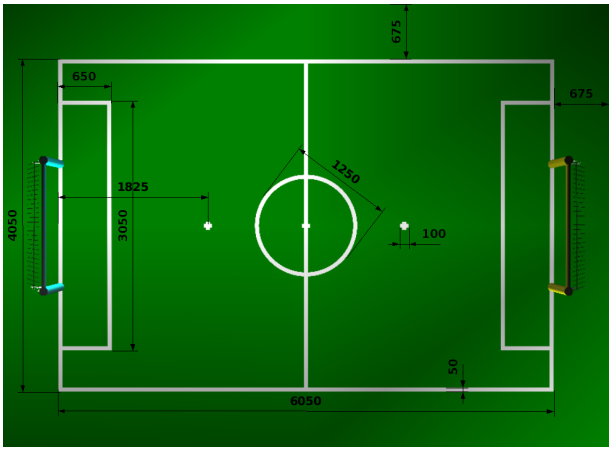


Figura 2.1: Esquema del campo de juego en la SPL de RoboCup.

Las reglas de la SPL de RoboCup son, en términos generales, las mismas que existen en el fútbol humano. Es decir, se juega con los pies, en un terreno limitado y el objetivo es anotar en el arco del equipo adversario. No obstante, existen ciertas restricciones adicionales para hacer el juego más apropiado a las capacidades de los robots, una de las más importante es el uso de objetos de interés con colores, geometría y dimensiones conocidas tales como: los arcos (ver figura 4.6), las líneas y combinaciones de éstas últimas como las esquinas (ver figura 4.11). El campo de juego en el que se desarrolla la acción es como el que se presenta en la figura 2.1.

2.2. Robot Nao

El robot Nao es un robot humanoide fabricado por la empresa francesa *Aldebaran Robotics* [2]. *Aldebaran Robotics* cuenta con una versión especialmente diseñada para el desarrollo de investigación en robótica, siendo esta plataforma la utilizada en la SPL de RoboCup y que se puede observar en la figura 2.2.

Figura 2.2: Robot Nao de *Aldebaran Robotics*.

El robot Nao tiene una altura de 58 cm aproximadamente y pesa cerca de unos 4.3 Kgs. Posee un total 23 grados de libertad (articulaciones): dos grados de libertad en la cabeza, cinco en cada brazo, uno en la pelvis, cinco en cada pierna y uno en cada mano. Además, está equipado con un procesador AMD Geode con una frecuencia de 500MHz y 256MB de memoria tipo SDRAM, además de 2GB memoria tipo flash. Cabe destacar la limitada capacidad de procesamiento de este robot, comparado con las capacidades actuales de un computador de escritorio. Sin embargo, para computadores embebidos, estas capacidades son las habituales. Esto es uno de los factores más importantes a considerar a la hora de diseñar e implementar estrategias de control y metodologías para resolver los problemas perceptuales en robótica, especialmente cuando estos están relacionados con el uso de información visual.

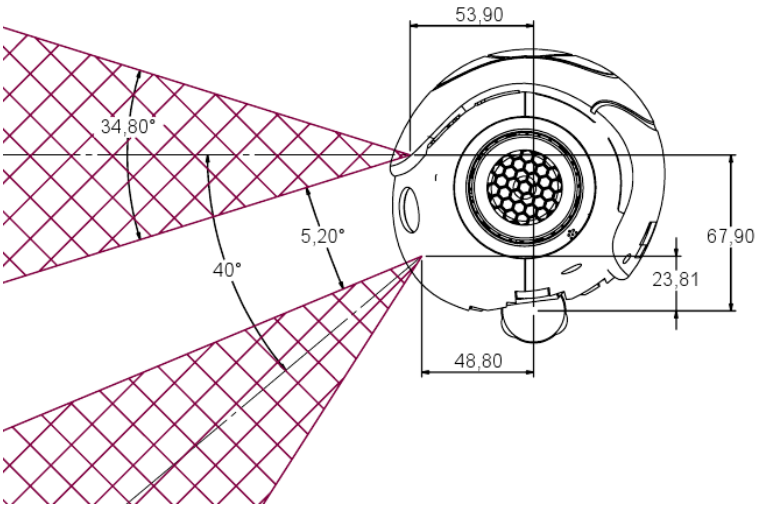


Figura 2.3: Distribución de las cámaras de video en la cabeza del robot Nao.

Este robot cuenta con sensores de tacto, sonares, giróscopos, acelerómetros y, en particular, dos cámaras de video CMOS, con resolución de 640x480, situadas en la cabeza del robot tal como se muestra en la figura 2.3. Las cámaras no pueden ser utilizadas simultáneamente, lo que impide realizar procesos de visión estéreo. El objetivo de tener dos cámara es tener un mayor campo visual.

El robot Nao es la principal plataforma de pruebas utilizada en la validación de esta tesis.

2.3. Librería de control UChileLib

El sistema propuesto en esta tesis, esta desarrollado como parte de una librería de control para robots móviles denominada UChileLib [20], cuyo desarrollo ha sido llevado a cabo por el Laboratorio de Robótica del Departamento de Ingeniería Eléctrica de la Universidad de Chile. La librería UChileLib es una librería generica de robótica, que puede ser adaptada para el control de múltiples plataformas, tanto reales como simuladas. Para ello, la arquitectura de software utilizada divide la implementación en dos grandes grupos:

- Módulos plataforma independientes: los módulos plataforma independientes implementan todos los métodos y librerías necesarias para el control del sistema robótico, sin que estos dependan de características específicas del *hardware* a controlar. Esto abarca desde la implementación de librerías de bajo nivel, por ejemplo para operaciones con matrices, manejo de arreglos dinámicos u operaciones con sistemas de coordenadas, hasta los módulos de alto nivel que implementan métodos complejos como filtros de Kalman o de partículas, el sistema de comunicación de datos entre módulos o el sistema de visión robótica. Para el caso de las librerías de más bajo nivel,

es evidente que es sencillo independizar el problema de la plataforma. Sin embargo, en el desarrollo de la librería UChileLib, esta independencia se ha llevado también a los módulos de alto nivel. En el módulo de visión, por ejemplo, se asume la existencia de una imagen de tamaño paramétrizado, capturada por una cámara con características parametrizadas también. De este modo, en todos los algoritmos se utilizan los parámetros de tamaño y características de forma genérica, pero dependiendo de la plataforma, estos tendrán valores particulares. Es en este punto donde se involucran los módulos plataforma dependientes.

- Módulos plataforma dependientes: estos corresponden a una serie de módulos que deben ser programados en cada robot donde se desee ejecutar la librería de control UChileLib. Estos módulos son los encargados de realizar toda la comunicación con el *hardware* de la plataforma y determinan el valor de los parámetros que en los módulos plataforma independientes se asumieron como datos. De este modo, un robot que quiera usar los métodos de visión robótica de la librería, necesitara implementar la lectura de la imagen y el llenado de los parámetros como tamaño, distancia focal y todos los necesarios por el algoritmo de visión que se desee ejecutar. Así mismo, en caso que se requiera, debe existir un módulo de lectura de los *encoders* y acelerómetros del robot, que llene dichos campos de información en el sistema independiente de la plataforma.

Gracias a esta estrategia de programación, en la actualidad existen cuatro plataformas diferentes operando con UChileLib, donde mas de un 90 % del código es común. Estas plataformas son: i) robot humanoide Hajimie Robot 18 [22], ii) robot humanoide Tanker UCH H1, desarrollado por el grupo de investigación del laboratorio de robótica [21], iii) robot humanoide comercial Nao [2] y un simulador de alto nivel desarrollado por el grupo de investigación del laboratorio de robótica, como parte de la librería UChileLib y denominado UChile HL-SIM [7]. El módulo de visión integrado propuesto en esta tesis es completamente independiente de la plataforma, pero se ha utilizado tanto en el robot Nao, como en el simulador UChile HL-SIM, para evaluar y validar su desempeño. En el resto de las plataformas se encuentran operativos todos los módulos tratados en esta tesis, pero no han sido realizadas evaluaciones cuantitativas de desempeño.

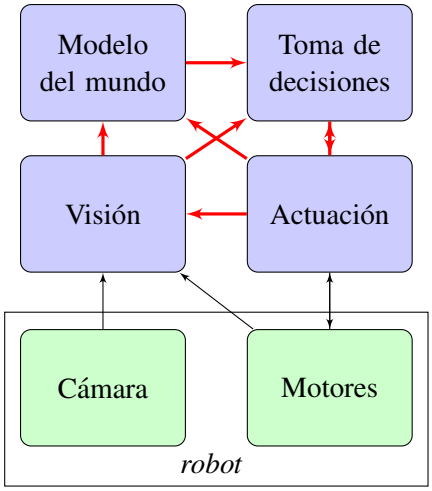


Figura 2.4: Arquitectura general UChileLib.

Desde el punto de vista abstracto y enfocado en los módulos de control principales, la arquitectura general de la librería UChileLib, mostrada en la figura 2.4, se puede dividir en 6 módulos. Estos son los siguientes:

- Cámara y *encoders*: este módulo entrega la información adquirida por los sensores del robot. En este trabajo son de interés la cámara y los *encoders*. La cámara corresponde al sensor principal del sistema y entrega una imagen en color, la cual será procesada para detectar los objetos relevantes. Los *encoders*, por su parte, indican la posición de cada uno de los motores del robot.

- Motores: este módulo recibe las órdenes para cada uno de los motores del robot, para lograr que éste se desplace de la forma deseada.

- Visión: es el principal módulo de percepción del sistema. Básicamente recibe la información de la cámara y de los *encoders*, y ejecuta un grupo de perceptores que intentan detectar los objetos de interés presentes en cada *frame*.

- Modelo del mundo: se encarga de mantener una estimación, tanto de la pose de los objetos de interés como de la pose del robot. Lo anterior permite generar un mapa global con toda esta información. Este módulo se divide en dos sub-módulos relevantes:

    - Estimación de poses: este módulo es el encargado de mantener una estimación del estado de cada uno de los objetos de interés en el espacio relativo al robot. El grupo de estimaciones de todos los objetos en este espacio se denomina mapa relativo. El módulo de estimación de poses recibe las detecciones generadas por el sistema de visión, así como las odometrías estimadas por el sistema de actuación. Para el proceso de filtrado y estimación, se utiliza un filtro extendido de Kalman, donde las detecciones visuales se usan como las observaciones del

filtro y las odometrías se utilizan como las señales de control del sistema para ejecutar la etapa predictiva del filtro.

- Auto-localización: es el módulo encargado de realizar el proceso de auto-localización del robot, el que estima la pose del robot en el sistema de referencia global. El módulo de auto-localización recibe como observaciones los estados de las estimaciones de los objetos de referencia. Dado que éstos son fijos, implementa un filtro de Kalman que en su etapa correctiva utiliza esta información para calcular la pose absoluta del robot en la cancha. La etapa predictiva utiliza la información odométrica entregada por el módulo de actuación del sistema.

- Actuación: es el módulo encargado de sincronizar y ejecutar las órdenes de movimientos en el *hardware* del robot. Por otra parte, estima la odometría ejecutada para que ésta sea utilizada por los módulos que la requieran.

- Toma de decisiones: este módulo toma todas las decisiones de acciones que debe realizar el robot. Recibe como entrada la información visual, la información de estimación de poses, la estimación de la auto-localización, la odometría y toda la información sensorial que pueda resultar útil para la determinación del comportamiento del robot. Dicha información se comunica al módulo de actuación para que este ejecute las acciones determinadas.

Adicionalmente, existe una capa en la librería que implementa un sistema de comunicaciones genérico, que permite definir la comunicación de cualquier tipo de dato entre los diferentes módulos del sistema. Esté permite la configuración de la comunicación en cualquier dirección y cada dato puede ser enviado al número de módulos receptores que sea necesario.

2.4. Herramienta de Desarrollo: Vision Tool

La librería UChileLib posee una serie de herramientas adicionales con interfaz gráfica que ejecutan algunos de los módulos descritos anteriormente con la finalidad de probarlos, verificar su funcionamiento, buscar errores, etc. Dichas pruebas serían prácticamente imposible de hacerse directamente en el robot. Una de ellas y la más utilizada en esta tesis es VisionTool, la cual puede procesar imágenes desde secuencias de videos o en línea desde el robot ejecutando el módulo de visión. Con ella, es posible visualizar sobre la imagen las detecciones o etapas de las detecciones de cada perceptor. También, en ella se puede realizar el procedimiento de calibración de colores (2.5.1) y la calibración de las matrices de corrección del estado de la cámara (4.1.3).

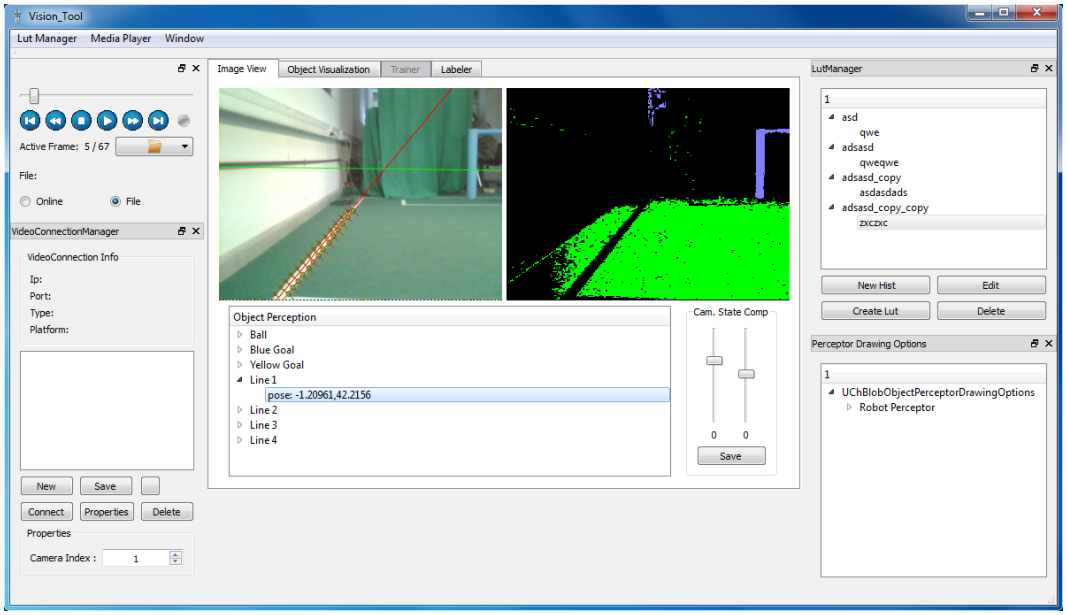


Figura 2.5: Captura de la interfaz de la herramienta de visión VisionTool.

2.5. Segmentación de colores

El diseño general del sistema de visión que se presenta en esta tesis se puede apreciar en la figura 2.6. Para la detección de las líneas, esquinas y t-esquinas se utilizará la información de altos gradientes de intensidad de la imagen, ya que no poseen un color identificable y principalmente porque su característica más importante es que su información de intensidad es altamente contrastable con la intensidad del fondo de la cancha. Dado que el arco es un objeto con volumen, e identificable uno de otro por su color, se utilizará la información de color proveniente de la imagen. Ésta información de color es pre-procesada por el módulo regiones color"de la figura 2.6 y es entregada al perceptor como regiones de pixeles vecinos de un mismo color . Éste módulo pertenece a un trabajo previo por lo que no forma parte del desarrollo de ésta tesis, motivo por el cual es descrito a continuación.

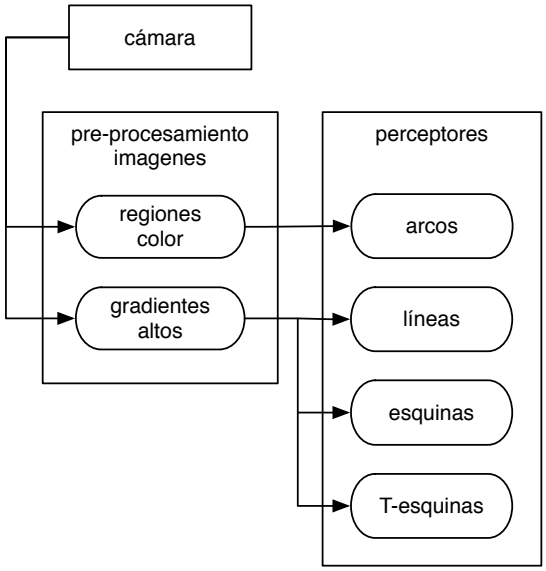


Figura 2.6: Diagrama de bloques del sistema de percepción básica

En la figura 2.6 se muestra un diagrama de bloques general del sistema de visión que se propone en esta tesis.

En este trabajo se desarrolla gran parte del diseño general del sistema de visión mostrado en el diagrama de bloques de la figura 2.6. Se exceptúa la generación de regiones de color por lo que se describe a continuación.

2.5.1. Generación de candidatos usando color

En términos generales, en determinados problemas de robótica los objetos de interés pueden tener la característica de poseer colores definidos y uniformes. Este es el caso del escenario del fútbol robótico, donde cada objeto de interés tiene un color conocido de antemano. Independiente de lo anterior, la importancia que se dará al color en la percepción de cada objeto es una decisión arbitraria. En nuestro caso, se ha decidido que para la pelota, cuyo perceptor no será abordado en este trabajo, y para los arcos, el color constituirá un característica fundamental en la generación de los candidatos a objetos. Para las líneas, esquinas y t-esquinas, en cambio, el color no será utilizado como un determinante absoluto del candidato.

De este modo, el módulo de generación de candidatos usando información de color tiene por objetivo detectar y caracterizar un conjunto de regiones formadas de pixeles conexos y del mismo color, para cada uno de los colores de interés. Para ello, se utiliza una tabla de color, que ha sido entrenada de forma *off-line*.

Segmentación de colores y la tabla de color

Dado un determinado espacio tri-cromático de color, de componentes Y, U y V, al que llamaremos espacio YUV, y dado un píxel perteneciente a dicho espacio de color, $p = (p_y, p_u, p_v)$, donde cada componente del píxel pertenece a un espacio $\mathbf{H}$, que corresponde al conjunto de números naturales [0 - 255] que es la resolución estándar del sensor de una cámara de video, entonces la tabla de color puede ser modelada como una función $T(p) : \mathbf{H}^3 \mapsto \mathbf{N}_c$, donde $\mathbf{N}_c$ corresponde al espacio de clases de color. El espacio de posibles clases de color incluye los colores de interés, además de la clase "ningún color", entregada en caso de que el píxel no corresponda a ninguno de los colores que se desean segmentar.

La generación de dicha tabla de color se realiza mediante un proceso *off-line*. Mediante el uso de la herramienta VisionTool, se capturan muestras etiquetadas de los colores de interés directamente sobre imágenes capturadas por la cámara del robot. Usando estas imágenes como guía, se seleccionan las fronteras de los objetos de interés mediante una región poligonal, asegurándonos que sólo queden pixeles del color deseado dentro de dicha región. Posteriormente, se indica manualmente a la herramienta, a qué clase de color pertenece la región seleccionada. El sistema almacena los datos con sus respectivas etiquetas que indican a qué clase de color pertenecen dichos datos. De esta forma, se genera un histograma de pixeles para cada clase de color.

Si definimos al elemento i del histograma asociado al color c, como $h_i^c = (p_y^{i,c}, p_u^{i,c}, p_v^{i,c})$, y dado una cantidad de n muestras asociadas al color c, cada clase de color puede ser modelada como una variable aleatoria normal C definida como:

$$C(c) \sim N(\overline{h}^c, \Sigma_c)$$

$$\overline{h}^c = \frac{1}{n} \cdot \sum_{i=1}^{n} h_i^c \tag{2.1}$$

$$\Sigma_c = \begin{pmatrix} \sigma_{yy} & \sigma_{yu} & \sigma_{yv} \\ \sigma_{uy} & \sigma_{uu} & \sigma_{uv} \\ \sigma_{vy} & \sigma_{vu} & \sigma_{vv} \end{pmatrix} \tag{2.2}$$

Donde cada componente σ está dada por:

$$\sigma_{ab} = \frac{1}{n} \cdot \sum_{i=1}^{n} [(p_a^{i,c} - \overline{p}_a^{i,c}) \cdot (p_b^{i,c} - \overline{p}_b^{i,c})] \tag{2.3}$$

Estas estadísticas definen una geometría elipsoidal en el espacio de color YUV. Para poder evaluar la región exacta de la clase de color, se debe definir un umbral de innovación. Este umbral representa la mínima probabilidad para aceptar un píxel determinado como perteneciente a la clase de color evaluada. De este modo, la frontera que define la elipse que inscribe a los pixeles de la clase de ese color, depende del umbral seleccionado. Este umbral puede ser ajustado mirando los resultados obtenidos hasta encontrar un óptimo visual o puede ser optimizado utilizando los mismos datos muestreados. Sin embargo, en esta tesis no es abordado este problema y se asume cierto resultado dado por el sistema implementado en la librería UChileLib.

La tabla de color es generada evaluando cada una de las estadísticas representantes de cada clase, en cada una de las celdas del espacio de color YUV. Dependiendo del umbral de innovación, el resultado de la evaluación dirá si el píxel pertenece o no a la clase en cuestión. Todo el proceso anterior es realizado de forma *off-line*.

La tabla de color es la única información de entrenamiento relevante durante la operación *on-line* del sistema. Ésta es leída como dato al iniciarse el proceso visual. Durante la operación, cada imagen es segmentada en color. Para esto, cada uno de los pixeles de la imagen es evaluado en la tabla de color. El proceso completo de evaluar todos los pixeles genera una segunda imagen que tendrá, en vez del valor original del píxel, el valor de la clase de color correspondiente. Los pixeles que no pertenecen a ninguna clase de color quedan clasificados como de clase neutra. Actualmente, en el escenario de SPL de RoboCup, existen cuatro colores de interés: blanco, celeste, amarillo, verde y naranjo, por lo que la imagen resultante corresponde a una imagen de tan solo 6 niveles posibles, correspondientes a las clases de color de interés más la clase nula.

Caracterización de regiones

La siguiente etapa involucrada en el proceso de generación de candidatos a objetos, consiste en la compresión de la imagen segmentada en color, para facilitar la identificación y caracterización de regiones conexas.

El proceso de compresión aprovecha el hecho de que la imagen segmentada en colores posee pocos valores posibles en sus pixeles (clases de color) y estos normalmente se encontraran en grupos conexos, debido a que las regiones suelen tener una uniformidad de color. A modo de ejemplo, el campo de juego siempre dará lugar a grandes regiones clasificadas como verdes, compuestas de pixeles conexos que tienen dicha clasificación. Por esta razón, se utiliza un algoritmo denominado *run lenght encoding*, el cual genera una representación de la imagen basada en segmentos conexos con los mismos valores en sus pixeles. Cada fila de la imagen segmentada es analizada para determinar las coordenadas y tamaño de segmentos de pixeles de una misma clase de color. De este modo, si en una fila de la imagen tenemos un segmento formado por 40 pixeles del mismo color, este segmento sera remplazado por las coordenadas del origen del segmento (fila n, columna m) y por un campo que indique el tamaño del segmento, en ves de los 40 valores idénticos indicando el color de cada píxel. En promedio, para las imágenes manejadas por el sistema, la reducción de tamaño en la representación, usando esta compresión, es en torno al 70 %.

La representación anterior no sólo presenta ventajas desde el punto de vista de la reducción de la información. La mayor contribución de esta representación radica en la eficiencia con que se puede lograr el siguiente paso del sistema de generación de candidatos, que es la formación de regiones conexas del mismo color. Sobre la representación basada en filas de pixeles conexos del mismo color, se realiza un procedimiento que, usando las coordenadas de las filas, determina cuáles filas de un mismo color están traslapadas entre sí, con lo cual quedan identificadas completamente las regiones conexas de cada color. Esto resulta considerablemente más eficiente que realizar el mismo proceso sobre los píxeles de la imagen segmentada original.

A medida que se generan las regiones conexas, el sistema les calcula ciertas características de forma incremental. De este modo, cada vez que se determina que un segmento pertenece a una región dada, esa región re-calcula estas características integrando el nuevo segmento. Las características de interés son: tamaño de la región, centro de masa, coordenadas del rectángulo que inscribe la región y ocho puntos del borde de la región, determinados como los puntos de borde en 8 diferentes sentidos: hacia arriba, hacia abajo, hacia ambos lados y hacia las 4 direcciones en 45°de las anteriores (4 diagonales).

De este modo, para cada color de interés se obtiene un arreglo de regiones caracterizadas. Este arreglo es ordenado según el tamaño de la región. Como en el escenario de la SPL de RoboCup, cada objeto de interés esta compuesto de un único color y todos los objetos tienen colores diferentes entre sí, las regiones de cada color pasan a ser regiones candidatas al objeto respectivo.

Capítulo 3

Trabajo relacionado

Contenido

Dentro del fútbol robótico exiten varias líneas de investigación como visión robótica, locomoción, auto-localización, fusión sensorial multi-robot, comportamientos, máquinas de aprendizaje, etc. Esta tesis se centra en visión robótica por lo que a continuación se muestra una revisión del estado del arte en ésta área.

3.1. Detección de objetos en el fútbol robótico

En el evento anual de la RoboCup[a], paralelamente a las competencias, se desarrollan conferencias donde los grupos de investigación o equipos comparten sus trabajos y desarrollos más innovadores en cualquiera de las líneas de investigación involucradas en el fútbol robótico. Particularmente en el área de visión robótica, el desafío ha sido cada vez mayor debido a que las características del campo de juego se complejizan año tras año, algunos ejemplos de esto son la eliminación de barandas blancas en torno a la cancha que evitaban

[a]RoboCup es una organización internacional que agrupa diversos centros de investigación en robótica. Cada año se organiza el mundial de robótica RoboCup, donde la Standard Platform League corresponde a una de las categorías de dicha competencia y donde el objetivo es el desarrollo de sistemas de control para robots bípedos humanoides basados en visión robótica, para que estos jueguen fútbol en un escenario predefinido. Sitio web oficial: http://www.robocup.org

la detección de falsos positivos, se han ido eliminando gradualmente el número de faros[b] (actualmente el campo de juego no posee ninguno) y que se utilizaban como *landmarks* que ayudaban en tareas de auto-localización, los arcos han pasado de ser simples cajones rectangulares de color a ser arcos similares a los de fútbol humano, etc.

Debido a esta mayor complejidad del entorno, la tendencia actual es detectar las líneas de la cancha, un proceso que es más complejo pues no es basado en color como lo era la detección de todos los demás objetos. Además, las líneas son objetos repetidos en el campo de juego y no poseen ninguna característica que distinga unas de otras, es por esto que el proceso de percepción

Debido a esta mayor complejidad del entorno, la tendencia actual es detectar e identificar las líneas de la cancha, un proceso que intuitivamente ya es más complejo que detectar e identificar objetos únicos de colores distintintivos y fáciles de identificar como los antiguos faros.

La mayoría de los desarrollos de visión robótica en RoboCup son basadas en color la que consiste en una clasificación de los píxeles de la imagen en un color definido con anterioridad mediante algún tipo de entrenamiento. El problema de clasificación de colores puede ser visto como un problema de *clustering*. Una posible solución para este problema de clustering es el denominado k-means [13]. Este método resuelve el problema de encontrar valores representantes para un número determinado de clases, a partir de un gran número de muestras. En los sistemas de visión basados en color, este es un proceso que se realiza de forma *off-line*. Determinados los representantes, el sistema de segmentación los utiliza para definir a cual clase pertenece cada dato de entrada nuevo que se desea clasificar esto puede hacerse de forma *on-line*, o bien se puede generar una tabla de clasificación para realizar el proceso *on-line* indexando de forma directa los valores de la tabla. Esta última solución se utiliza para acelerar el proceso, pues todos los cálculos de pertenencia son pre-calculados al generar la tabla. Sin embargo, tiene el inconveniente de que la tabla suele utilizar una cantidad de memoria importante, lo que obliga a disminuir la resolución de la representación de las clases para disminuir el espacio utilizado en memoria, con la consecuente disminución de la calidad de la segmentación obtenida.

Dado que el método de entrenamiento k-means puede tener una solución extremadamente costosa a medida que se tienen más datos y a medida que se desean clasificar más clases, existen híbridos interesantes con sistemas de búsqueda de óptimos que intentan acelerar k-means. Este es el caso de Genetic k-means [11], que es un híbrido con un algoritmo genético. En este sistema, cada individuo del algoritmo genético que optimiza la función objetivo, corresponde a una matriz de pertenencia de cada dato a una clase. La función objetivo es calculada como la suma cuadrática de las distancias de un conjunto de datos a su media. De este modo, el algoritmo genético de búsqueda itera mutando y recombinando matrices de pertenencia de datos a las clases, calculando para cada matriz la distancia de esos datos a la media de cada clase determinada por la misma matriz.

Otra solución posible al problema de encontrar las clases de color, es el entrenamiento

[b]En robótica móvil un faro corresponde a un objeto cuyas características de color y forma se conocen de antemano, esto permite desarrollar sistemas de percepción visual para detectarlos, por lo general la disposición de estos objetos también es conocida

manual, donde los representantes son generados a través de la selección manual de valores desde un conjunto de imágenes de ejemplo [23]. Para el caso de un sistema de entrenamiento manual, existe la posibilidad de llenar inmediatamente la tabla de color, a medida que se realiza el entrenamiento, sin embargo, en este caso se hacen necesarias operaciones sobre la tabla de color del tipo dilatación y erosión [20,23], para rellenar y completar los espacios no clasificados. Debido a las condiciones actuales de la capacidad de procesamiento de los sistemas robóticos, independiente del método que se utilice para determinar las clases de los colores de interés, se hace absolutamente necesaria la operación en línea utilizando una tabla de color. Como se mencionó antes, esto consiste en llenar una tabla que asocia a cada coordenada del espacio de color, la clase de color correspondiente determinada en el entrenamiento [3, 10, 15, 17, 20].

Una alternativa para facilitar el entrenamiento manual y evitar la necesidad de operaciones de dilatación y erosión, es representar el muestreo realizado de una forma continua. En la literatura se han utilizado diversas representaciones como un espacio cuadrado o rectangular para representar cada clase de color [17], basándose en el promedio de las muestras para saber el centro de la clase e imponiendo normas heurísticas para determinar las fronteras de la misma. Otros, en cambio, han intentado generalizado las representaciones basándose en funciones de distribución de probabilidades que dan lugar a representaciones elípticas de las clases [15]. Una representación que ha probado ser bastante efectiva, es la representación elíptica con rotaciones. Esto se representa mediante un vector de medias y una matriz de covarianzas en el espacio de color utilizado. Si se asume una distribución gaussiana de probabilidad de que un píxel pertenezca a una clase de color determinada, bastan dichas estadísticas más un valor umbral (se denomina umbral de innovación) que defina el decaimiento máximo de la distribución gaussiana para que el píxel siga perteneciendo a la clase. Con esto, se puede llenar la tabla de color de forma bastante efectiva y bien generalizada [8]. Esta representación estadística de medias y covarianzas, más el umbral de innovación establecido, definen geométricamente, en el espacio tridimensional de colores, una elipse con su punto medio en las medias correspondientes, y rotada de acorde a la matriz de covarianzas entre las componentes del espacio de color.

Debido a las restricciones de procesamiento, la mayoría de los trabajos relacionados con fútbol robótico se basan en color, ya que las condiciones de iluminación varían muy poco a lo largo del día y entre diferentes campos de juego.

Luego de definir el método de clasificación existen dos principales formas de segmentar la imagen: (i) segmentar de la totalidad de la imagen, es decir todos los píxeles, ó (ii) segmentar parcialmenta la imagen.

La segmentación completa de la imagen como en [3, 10], hace más fácil la generación de regiones candidatas, de esta forma la extracción de características resulta simple ya que se cuenta con la máxima información que puede entregar la segmentación. Este proceso de segmentación total de la imagen cada vez es menos frecuente entre los trabajos de vision robótica ya que demora bastante clasificar todos los píxeles. En un intento de disminuir el tiempo de segmentación en [9] trabajan con la imagen estándar más pequeña (120x160).

Existen varias formas de realizar una segmentación parcial: (i) la segmentación bajo líneas de escaneo equidistantes entre sí [18], (ii) el uso de sonar visual [19] donde la densi-

dad de líneas de escaneo depende de la línea del horizonte y (iii) la segmentación por zonas de densidad de acuerdo a la distancia al horizonte. El beneficio de la segmentación parcial es la pequeña cantidad de píxeles a segmentar lo que implica una gran disminución del tiempo de procesamiento de esta etapa. El problema de la segmentación parcial es la extracción de características de las unidades candidatas, al tener menos información de la segmentación se requiere de procedimientos más complejos para armar y determinar la información de los candidatos.

Cabe destacar que la segmentación total es una solución más generalizada al momento de incluir otros objetos de interés del campo de juego, en su contraparte, la segmentación parcial puede ser una buena solución para los objetos actuales pero probablemente requerirá nuevas heurísticas si se requiere detectar otros objetos.

La detección de líneas del campo de juego se lleva a cabo de acuerdo al método de segmentación que se use. Por ejemplo en [4] utilizan líneas de escaneo para segmentar la imagen, los puntos medios de los pequeños segmentos blancos entre segmentos verdes son candidatos a formar parte de una línea de cancha. Luego mediante un algoritmo van uniendo los puntos formando cadenas de según: (i) si fue generado por una línea de escaneo diferente al del último punto aregado, (ii) su alineación y (iii) condiciones de distancia. En [18] las líneas de escaneo generan pequeños sectores blancos los que son sometidos a un algoritmo de *clustering* para ver si entre ellos forman una línea de cancha. En [9] utilizan la segmentación completa de la imagen, por lo que las líneas son analizadas como regiones enteras a las que le asignan una ecuación de la recta que mejor las representa.

Para la detección de otros objetos con volumen como por ejemplo arcos, en [18] los segmentos celestes o amarillos son sometidos a una transformación de Hough para encontrar los postes o travesaños de un arco candidato. En [9] al utilizar la segmentación completa extraen directamente las características de la regiones celestes y amarillas para determinar si corresponden a un arco.

Capítulo 4

Sistema de detección robusta de objetos

Contenido

A continuaciónn se describen los desarrollos del sistema de visión propuesto.

4.1. Estado de la cámara y sus aplicaciones

El estado de la cámara consiste en la posición y orientación que posee esta respecto al sistema de coordenadas del robot en el momento que se adquiere la imagen.

El sistema de visión es el encargado de procesar las imágenes capturadas y determinar si existen objetos de interés proyectados en ella. Dentro de los procesos de detección existe la posibilidad de utilizar información acerca de las relaciones físicas que posee el objeto real y su entorno, por ejemplo que la pelota se encuentra a la altura del suelo. Esto permite establecer regiones en la imagen donde es altamente posible encontrar la pelota y a la vez regiones donde es imposible. Para esto es necesario conocer el estado de la cámara en el instante que se adquirió la imagen. Por otra parte, una vez que un objeto es detectado, se calcula su pose respecto al sistema de coordenadas del robot de acuerdo a su posición en la imagen, su geometría real y el estado de la cámara. Estas dos circunstancias evidencian la importancia del conocimiento del estado de la cámara por parte del sistema de visión.

Previo a cualquier explicación, es necesario identificar dos sistemas de coordenadas: el sistema de coordenadas del robot y el sistema de coordenadas de la cámara. La figura 4.1

muestra el sistema de coordenadas del robot. Éste posee su origen en el centro del robot, entre ambos pies y a la altura de ellos (o del piso). El eje *x* del sistema apunta hacia adelante del robot, el eje *y* hacia el lado izquierdo y finalmente el eje *z* hacia arriba.

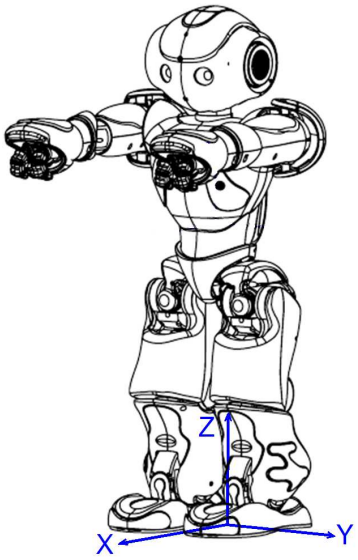


Figura 4.1: Sistema de coordenadas del robot.

La figura 4.2 muestra el sistema de coordenadas de ambas cámaras, el origen se encuentra en el centro del sensor de la cámara o foco. El eje *x* del sistema apunta hacia adelante de la cámara perpendicularmente al plano del sensor de la cámara, el eje *y* paralelamente al sensor de la cámara hacia el lado izquierdo y finalmente el eje *z* también paralelo al sensor de la cámara pero apuntando hacia arriba.

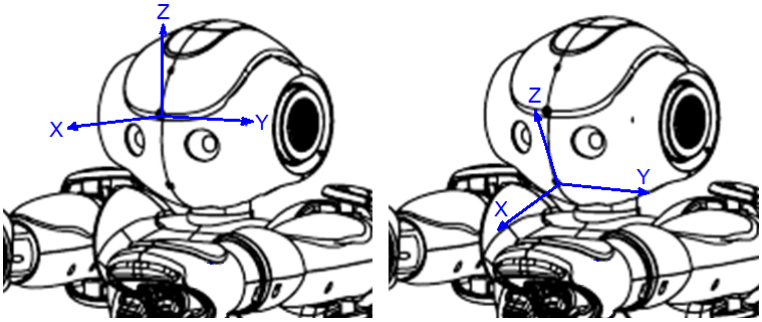


Figura 4.2: Sistemas de coordendas de ambas cámaras.

El estado de la cámara corresponde directamente a la transformación del sistema de coordenadas de la cámara al sistema de coordendas local del robot. En la literatura existen varias técnicas para describir una transformación espacial de coordenadas en tres dimen-

siones, entre ellas se encuentran: ángulos de euler, par de rotación, cuaterniones, matrices homogéneas, etc. [6]. En este trabajo se utilizan matrices homogéneas las que se componen post-multiplicando matrices de traslación y matrices de rotación según corresponda.

Una matriz homogénea M está compuesta por una matriz de rotación M_{rot} y un vector de traslación M_{tras}:

$$M = \left[\begin{array}{ccc|c} M_{rot} & & M_{tras} & \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.1}$$

La matriz de rotación M_{rot} es una matriz de dimensiones $3x3$ y el vector de traslación M_{tras}, es un vector de dimensiones $3x1$:

$$M_{rot} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.2}$$

$$M_{tras} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.3}$$

4.1.1. Estado de la cámara

Para determinar la matriz homogénea que define el estado de la cámara, denominada en adelante por M, es necesario dividir el robot en dos secciones: inferior y superior. La sección inferior comprende desde el sistema de referencia del robot (pies) hasta un sistema de referencia local ubicado en el punto medio del torso del robot, la transformación de esta sección inferior se denomina M_{inf}. La sección superior considera desde el sistema local del centro del torso del robot hasta el sistema local de la camara en uso, la transformación de esta sección inferior se denomina M_{sup}. Una vez calculadas ambas transformaciones es posible multiplicarlas para obtener M de acuerdo a la siguiente ecuación:

$$M = M_{inf} \cdot M_{sup} \tag{4.4}$$

En la sección inferior M_{inf} se deben determinar las matrices de transformación de cada pie, P_{der} para el pie derecho y P_{izq} para el pie izquierdo. En la sección superior M_{sup} se deben definir las matrices de transformación para cada cámara, C_{sup} para la cámara superior y C_{inf} para la cámara inferior. Para determinar cada una de ellas se requiere utilizar la cinemática directa del robot, la que depende de sus medidas físicas y del estado angular de las articulaciones involucradas en el movimiento. Para este caso particular las longitudes del robot son constantes, por lo que las cuatro matrices sólo quedan en función del estado angular de las articulaciones correspondientes. Gracias a una serie de sensores, denominados *encoders*, que posee el robot en cada una de sus articulaciones, se pueden evaluar las matrices homogéneas en cualquier instante. El detalle del cálculo de estas matrices está descrito en el anexo 1 al final de este documento.

P_{der} y P_{izq} poseen su sistema de referencia local en sus respectivos pies, por lo que es necesario moverlos al sistema de referencia del robot. Para esto se debe premultiplicar la

matriz P_{der} por una matriz P_{der}^{correc} que representa la traslación y rotación del sistema de referencia del pie respecto al del robot:

$$P'_{der} = P_{der}^{correc} \cdot P_{der} \tag{4.5}$$

Esta matriz P_{der}^{correc} se obtiene considerando solamente las traslaciones en los ejes x e y y la rotación en torno al eje z de P_{der} según como se muestra en la siguiente ecuación:

$$P_{der}^{correct} = \begin{bmatrix} P_{der_{11}} & P_{der_{12}} & 0 & P_{der_{14}} \\ P_{der_{21}} & P_{der_{22}} & 0 & P_{der_{24}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{4.6}$$

Análogamente P_{izq} también se debe que premultiplicar por una matriz $P_{izq}^{correct}$ deducida directamente de la misma P_{izq} y teniendo las mismas consideraciones que P_{der}^{correc}.

$$P'_{izq} = P_{izq}^{correc} \cdot P_{izq} \tag{4.7}$$

Donde:

$$P_{izq}^{correct}() = \begin{bmatrix} P_{izq_{11}} & P_{izq_{12}} & 0 & P_{izq_{14}} \\ P_{izq_{21}} & P_{izq_{22}} & 0 & P_{izq_{24}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{4.8}$$

Con esto se tienen las matrices de transformación de la cadera respecto a cada pie con un mismo sistema de referencia original, el sistema de referencia local del robot, y denominadas por P'_{der} ó P'_{izq}.

Para determinar M_{inf} se debe conocer el pie de apoyo del robot en el instante que se requiere evaluar el estado de la cámara, esto es debido al balaceo constante del robot entre un pie y el otro cuando camina. Para esto, se usa la información dada por cuatro sensores de presión localizados en cada pie del robot. En la práctica, resulta bastante certero asumir que los sensores del pie que suman un mayor valor de presión, corresponden a los sensores del pie de apoyo, por lo tanto M_{inf} toma directamente los valores de P'_{der} ó P'_{inf} de acuerdo a la siguiente ecuación:

$$M_{inf} = \begin{cases} P'_{der} & \text{si } F_{der} > F_{izq} \\ P'_{izq} & \text{si } F_{der} \leq F_{izq} \end{cases} \tag{4.9}$$

Donde F_{der} es la presión medida en el pie derecho y F_{izq} en el pie izquierdo. Finalmente M_{inf} queda definida sólo en función del estado angular de las articulaciones de las piernas y caderas y del pie en el que se encuentra apoyado el robot.

M_{sup} queda definida según el estado angular de los motores del cuello del robot y de la cámara con la que se adquirió la imagen actual, por lo que tomará directamente los valores de C_{sup} ó C_{inf} según la siguiente ecuación:

$$M_{sup} = \begin{cases} C_{sup} & \text{si la imagen se adquirió con la cámara superior.} \\ C_{inf} & \text{si la imagen se adquirió con la cámara inferior.} \end{cases} \tag{4.10}$$

Conociendo los valores de M_{inf} y M_{sup} podemos obtener finalmente el estado de la cámara M relativo al sistema de coordenadas del robot según la ecuación 4.4.

4.1.2. Horizonte visual

Conociendo algunas características físicas de los objetos que se espera detectar en un sistema de visión, es posible generar reglas basadas en las relaciones espaciales que poseen los objetos de interés con su entorno. Una de ellas es la altura de los objetos reales. Conociendo el estado de la cámara es posible delimitar zonas en la imagen donde ellos se pueden proyectar y dónde no de acuerdo a su geometría real. Para esto resulta práctico definir una altura de referencia en el sistema de coordenadas del robot y proyectarla en la imagen. Bajo estos requerimientos es posible definir y utilizar el horizonte visual. En la figura 4.3, se puede apreciar una representación del horizonte visual. En esta figura el plano P corresponde al plano de proyección que es paralelo al sensor de la cámara y que se encuentra ubicado a distancia focal desde el centro de proyección o foco. Además, se observa el plano H, paralelo a la superficie de la cancha y ubicado a la altura del foco de la cámara. Descritos los planos P y H podemos definir como horizonte visual al segmento $\overline{h_{izq}h_{der}}$ resultante de la intersección de ambos planos. En la figura también se puede apreciar los sistemas de coordenadas del robot $OXYZ$ y el sistema de referencia de la cámara $O'UVW$, este último se encuentra trasladado y rotado respecto al del robot según una matriz homogénea M.

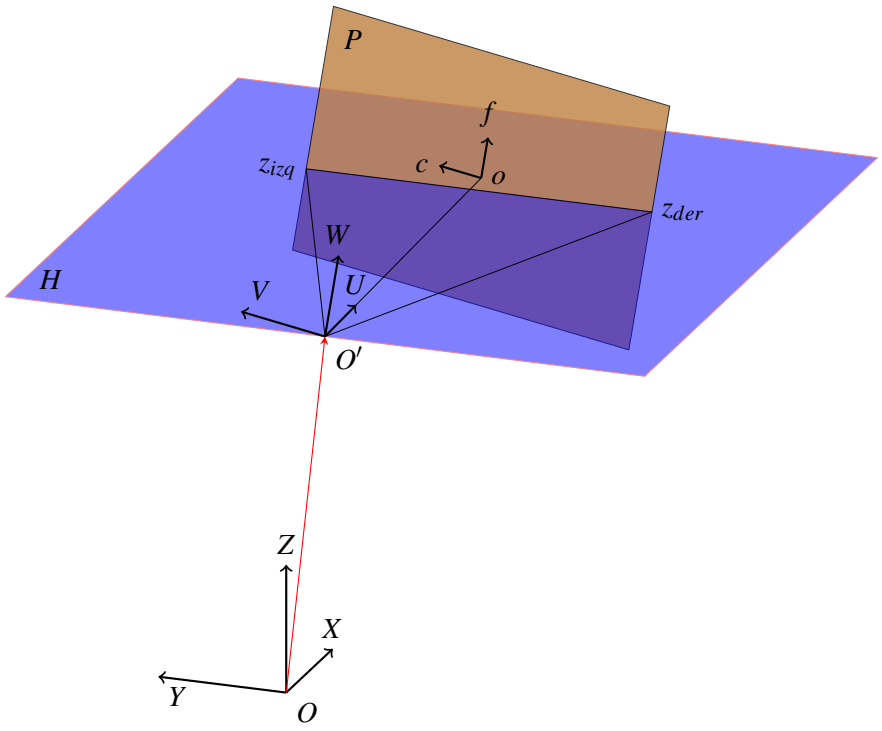


Figura 4.3: Intersección de los planos P y H que dan origen al horizonte visual.

Los puntos h_{izq} y h_{der} en la figura 4.3, representan los extremos del segmento de horizonte y se pueden definir sus posiciones respecto al sistema de referencia $O'UVW$ de la cámara como:

$$h_{izq} = \begin{bmatrix} distanciafocal \\ resolucionhorizontal/2 \\ h_{izq_z} \end{bmatrix} \tag{4.11}$$

$$h_{der} = \begin{bmatrix} distanciafocal \\ -resolucionhorizontal/2 \\ h_{der_z} \end{bmatrix} \tag{4.12}$$

Multiplicando ambos puntos por la sub-matriz de rotación M_{rot}, definida en la ecuación 4.1, obtenemos dos puntos que denominaremos h^r_{izq} y h^r_{der}:

$$h^r_{izq} = M_{rot} \cdot h_{izq} \tag{4.13}$$

$$h^r_{der} = M_{rot} \cdot h_{der} \tag{4.14}$$

Esta rotación es equivalente a orientar el sistema de referencia de la cámara $O'UVW$, con el sistema de referencia del robot $OXYZ$, esta alineación de sistemas de referencia permite encontrar los valores de h_{izq} y h_{der} que hacen que h^r_{izq} y h^r_{der} se encuentren definidos

en el plano H. Esto se logra igualando la componente z de h^r_{izq} y h^r_{der} a cero, como se muestra a continuación:

$$\begin{bmatrix} M_{rot_{11}} & M_{rot_{12}} & M_{rot_{13}} \\ M_{rot_{21}} & M_{rot_{22}} & M_{rot_{23}} \\ M_{rot_{31}} & M_{rot_{32}} & M_{rot_{33}} \end{bmatrix} \cdot \begin{bmatrix} distanciafocal \\ resolucionhorizontal/2 \\ h_{izq_z} \end{bmatrix} = \begin{bmatrix} h^r_{izq_x} \\ h^r_{izq_y} \\ 0 \end{bmatrix} \tag{4.15}$$

$$\begin{bmatrix} M_{rot_{11}} & M_{rot_{12}} & M_{rot_{13}} \\ M_{rot_{21}} & M_{rot_{22}} & M_{rot_{23}} \\ M_{rot_{31}} & M_{rot_{32}} & M_{rot_{33}} \end{bmatrix} \cdot \begin{bmatrix} distanciafocal \\ -resolucionhorizontal/2 \\ h_{der_z} \end{bmatrix} = \begin{bmatrix} h^r_{der_x} \\ h^r_{der_y} \\ 0 \end{bmatrix} \tag{4.16}$$

Los valores de h_{der_z} y h_{izq_z} pueden ser despejados de las ecuaciones anteriores, los que quedan como:

$$h_{izq_z} = -\frac{distanciafocal \cdot M_{rot_{31}} + resolucionhorizontal/2 \cdot M_{rot_{32}}}{M_{rot_{33}}} \tag{4.17}$$

$$h_{der_z} = -\frac{distanciafocal \cdot M_{rot_{31}} - resolucionhorizontal/2 \cdot M_{rot_{32}}}{M_{rot_{33}}} \tag{4.18}$$

De esta forma completamos las coordenadas de h_{izq} y h_{der} de acuerdo a 4.11 y 4.12 definidas en el sistema de la cámara $O'UVW$. Dado que son puntos en el plano de proyección de la imagen, podemos deducir directamente que el segmento de horizonte visual proyectado en la imagen con sistema de coordenadas ocf está definido por los puntos:

$$h^{imagen}_{izq} = \begin{bmatrix} resolucionhorizontal/2 \\ h_{izq_z} \end{bmatrix} \tag{4.19}$$

$$h^{imagen}_{der} = \begin{bmatrix} -resolucionhorizontal/2 \\ h_{der_z} \end{bmatrix} \tag{4.20}$$

El plano XY del sistema de referencia del robot y el plano XY del sistema de referencia global se asumen paralelos y a la misma altura, al igual que los demás sistemas de referencia de cada objeto dispuesto en el sistema de referencia global de la cancha. Bajo esta condición de paralelismo y altura, podemos hacer uso de la línea de horizonte proyectada en la imagen, para elaborar reglas contextuales físicas utilizadas en los procesos de detección de objetos.

4.1.3. Calibración del estado de la cámara

Como sabemos los modelos asociados a sistemas reales no son perfectos y la matriz homogénea M, que representa el estado de la cámara, no es la excepción. Hemos determinado que el estado de la cámara depende directamente del estado angular de las articulaciones el cual es medido por unos sensores de ángulo llamados *encoders*. Los *encoders* son sensores que, evidentemente, pueden tener errores imperceptibles si lo estudiamos aisladamente, pero pueden generar errores de posición grandes si consideramos las traslaciones de las partes

longitudinales del robot. Además, debemos considerar que el ensamblaje del robot tampoco es perfecto lo que puede ocacionar ”juegos”de las articulaciones que no son medidos por los *encoders*. Una de las evidencias más claras de este problema es la proyección de la línea del horizonte visual en la imagen, la que muchas veces se ve desalineada respecto a una referencia real de horizonte, en la figura 5.2 se aprecia la línea del horizonte (verde) desalineada respecto a una cañería de cobre que está paralela al plano del suelo. También se ve afectado el cálculo de la posición de un objeto, éstos son cada vez mayores mientras más alejado se encuentra el objeto.

Se espera que los errores angulares sean mayores en torno al eje y ya que la mayoría de las articulaciones del robot utilizado corresponden a rotaciones en torno a este eje. Por ésto se realiza un proceso de calibración donde se generan dos matrices de rotación en torno al eje y adicionales a las ya existentes, tal como si fueran dos pequeñas .ªrticulaciones virtualesçon el fin de compensar los errores angulares. Éstas compensaciones se aplican en dos puntos: la primera matriz de compensación se postmultiplica a la matriz de transformación de la cadera M_{inf} y la segunda a la matriz de transformación de la cámara M_{sup} definiendo M según la siguiente ecuación:

$$M = M_{inf} \cdot M_{inf}^{comp} \cdot M_{sup} \cdot M_{sup}^{comp} \tag{4.21}$$

Donde M_{inf}^{comp} y M_{sup}^{comp} corresponden a las matrices de compensación postmultiplicadas al nivel de la cadera y al nivel de la cámara respectivamente. Cada una de éstas matrices de transformación dependen de un pequeño ángulo que es calibrado utilizando la herramienta VisionTool que posee dos barras que representan el valor del ángulo de cada matriz de compensación, mientras se está conectado al robot se pueden mover manualmente hasta lograr alinear lo mejor posible la proyección de la línea del horizonte en la imagen con una referencia real de éste. Este par de pequeños ángulos, que definen a las matrices de compensación, varían de robot en robot por lo que cada uno debe ser calibrado independientemente.

4.1.4. Cálculo de la posición de un objeto

Una vez que un perceptor ha detectado un objeto, es necesario calcular su pose relativa al sistema de referencia del robot para posteriormente utilizarla como información para tomar desiciones, corregir la estimación de estado de objetos o para ayudar en la autolocalización del robot.

Es importante mencionar que es posible calcular la posición relativa al robot de un objeto gracias a dos condiciones: i) el sistema de coordenadas del robot está siempre a la misma altura que el sistema de coordenadas global y ii) la geometría y dimensiones reales del objeto de interés son conocidas. Bajo estas condiciones es posible determinar la posición de un objeto utilizando como datos: un punto de éste proyectado en la imagen, la altura que posee este punto en la realidad y el estado de la cámara M en ese instante.

En la figura 4.4, se muestra el sistema de coordenadas $OXYZ$, centrado en el pie de apoyo y el sistema $O'UVW$, centrado en el foco de la cámara. El sistema $O'UVW$ se encuentra trasladado y rotado respecto al sistema $OXYZ$. La región rectangular oscurecida representa el plano de proyección de la imagen la que posee el sistema de coordenadas ocf centrado

en la imagen. El punto B, corresponde a un punto de altura real conocida de un objeto de interés. El objetivo de este proceso es calcular la posición del punto B respecto al sistema de coordenadas $OXYZ$.

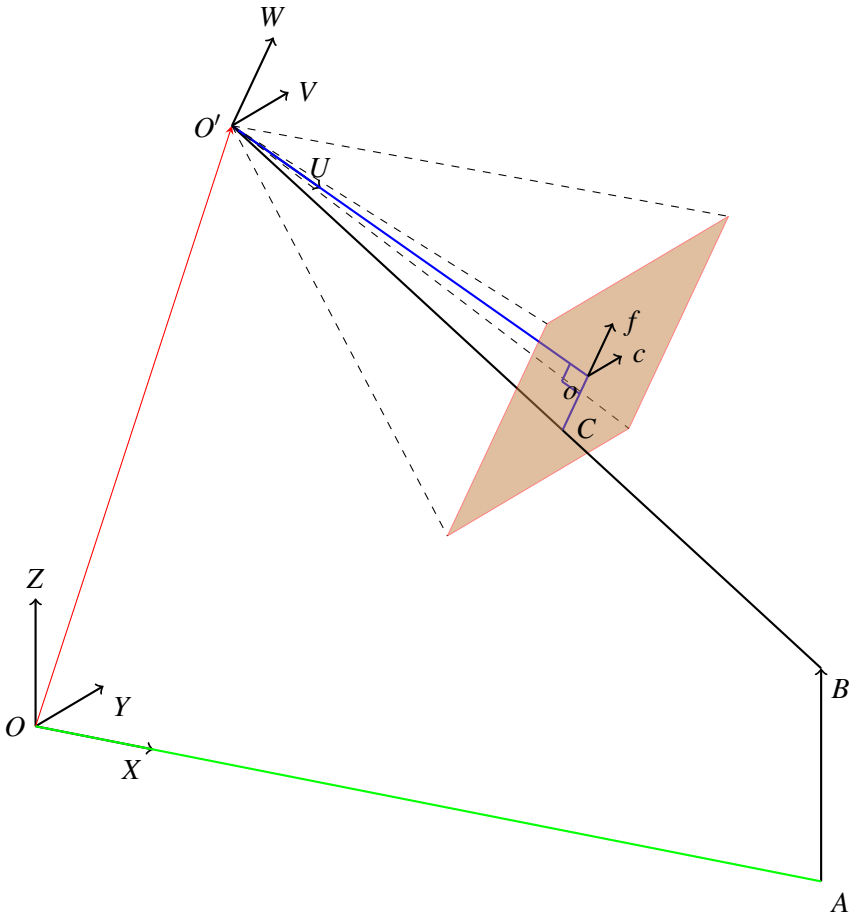


Figura 4.4: Esquema de los sistemas de coordenadas del robot y de la cámara utilizado para el cálculo de la pose de los objetos.

El punto B se proyecta en la imagen en el punto C cuyas coordenadas en la imagen se expresan como:

$$C_{ocf} = [I_{col}, I_{fil}] \tag{4.22}$$

Dado que el punto C se encuentra en el plano de proyección de la imagen es posible utilizar la distancia focal para expresar la posición del punto C en el sistema de referencia de la cámara $O'UVW$:

$$C_{O'UVW} = \begin{pmatrix} C_u \\ C_v \\ C_w \end{pmatrix} = \begin{pmatrix} DF \\ I_{col} \\ I_{fil} \end{pmatrix} \tag{4.23}$$

Si multiplicamos la sub-matriz de rotación M_{rot} por el vector $C_{O'UVW}$, se obtiene el punto C_r. Esta rotación es equivalente a alinear los ejes coordenados del sistema de referencia $O'UVW$ con los del sistema de referencia $OXYZ$, dando lugar a un sistema rotado $O'U'V'W'$ desde el cual el punto C_s se verá como $C_r = M_{rot} \cdot Cs$. Esta situación se ilustra en la figura 4.5.

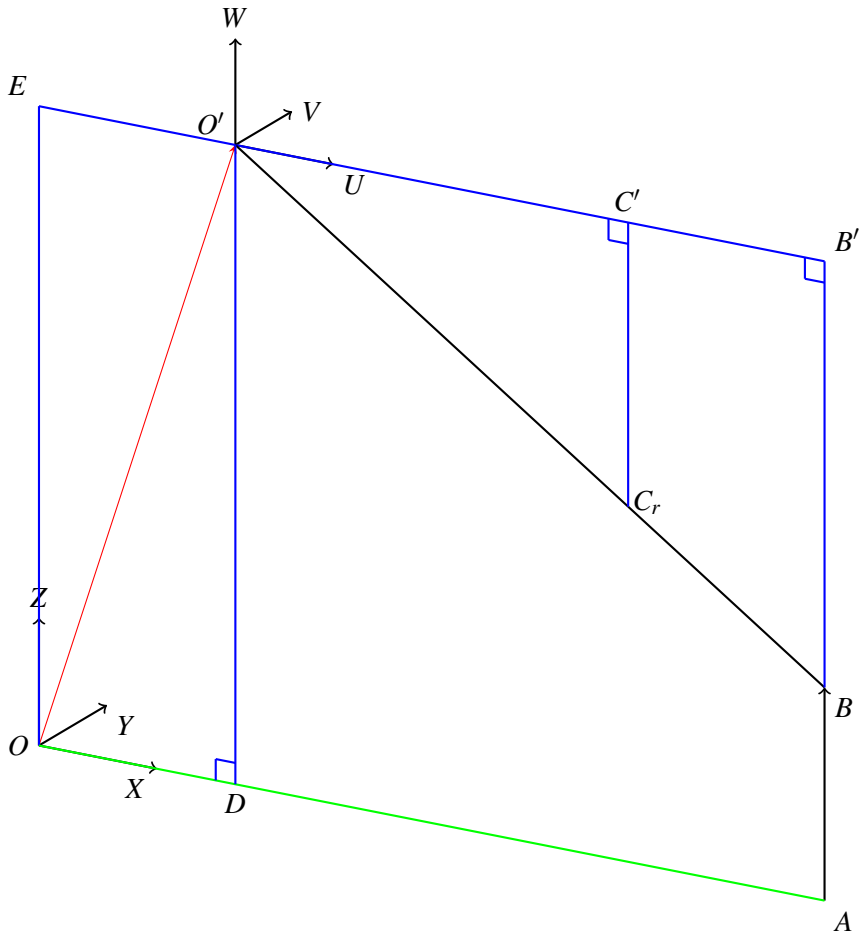


Figura 4.5: Sistema $O'U'V'W'$ originado de la rotación de $O'UVW$ en M_rot.

Para simplificar esta explicación y las figuras utilizaremos sólo las proyecciones sobre el plano XY del sistema de referencia $OXYZ$ con las que es posible calcular la componente x de la posición del punto B en el sistema de referencia del robot. Haciendo las mismas proyecciones en el plano YZ, es posible calcular la componente y de la posición del punto B de forma análoga.

Como se observa en la figura 4.5, al realizar el proceso de rotación y proyectando sobre el plano XZ, se forman dos triángulos rectángulos semejantes de vértices $O'C_rC'$ y $O'BB'$, para los cuales se cumple la siguiente relación entre sus segmentos de acuerdo al teorema de Thales:

$$\frac{\overline{O'C'}}{\overline{C_rC'}} = \frac{\overline{O'B'}}{\overline{BB'}} \tag{4.24}$$

$$\overline{O'B'} = \frac{\overline{O'C'}}{\overline{C_rC'}} \cdot \overline{BB'} \tag{4.25}$$

Como se ve en la figura 4.5, el segmento $\overline{O'B'}$ puede ser proyectado en el eje X del sistema, dando lugar al segmento $\overline{DA}$. Por otra parte, $\overline{O'C'}$ corresponde a la componente u' del punto C_r y el segmento $\overline{C_rC'}$ corresponde a la componente w' del punto C_r, las cuales son conocidas gracias a la rotación aplicada. Por otra parte, la posición de O' en el sistema relativo al robot $OXYZ$, denotado por $O' = (O'_x, O'_y, O'_z)$, se puede calcular usando la matriz homogénea de la siguiente forma:

$$\begin{pmatrix} O'_x \\ O'_y \\ O'_z \\ 1 \end{pmatrix} = M \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.26}$$

De este modo, la altura a la que se encuentra la cámara, en el sistema de coordenadas relativo al robot, corresponde a la componente O'_z, con lo cual se puede calcular el segmento $\overline{BB'}$ como la diferencia algebraica entre la altura de la cámara y la altura $\overline{AB}$ del del punto B del objeto, la que es conocida. Si denotamos el punto C_r visto desde el sistema $O'U'V'W'$ como:

$$C_r = \begin{pmatrix} C_r^{u'} \\ C_r^{v'} \\ C_r^{w'} \end{pmatrix} \tag{4.27}$$

Entonces podemos expresar el segmento $O'B'$ como:

$$\overline{O'B'} = \frac{C_r^{u'}}{C_r^{w'}} \cdot (O'_z - \overline{AB}) \tag{4.28}$$

Finalmente, la coordenada x del punto B, queda determinada por la suma de los segmentos $\overline{OD}$ y $\overline{DA}$. Como ya se mencionó, $\overline{DA}$ es igual al vector $\overline{O'B'}$ ya calculado. Por otra parte, $\overline{OD}$ corresponde a la componente x de la posición del punto O', dada por O'_x. Como ya se mencionó, la componente y se estima de forma análoga. La componente z, por su parte, corresponde directamente a la altura a la que se encuentra el punto. Así, la posición del punto B, queda determinada por la siguiente ecuación:

$$\begin{pmatrix} B_x \\ By \\ B_z \end{pmatrix} = \begin{pmatrix} O'_x + \frac{C_r^{u'}}{C_r^{w'}} \cdot (O'_z - \overline{AB}) \\ O'_y + \frac{C_r^{v'}}{C_r^{w'}} \cdot (O'_z - \overline{AB}) \\ \overline{AB} \end{pmatrix} \tag{4.29}$$

Dado que podemos conocer la posición de cualquier punto de un objeto de acuerdo a su altura, la orientación del objeto puede ser determinada calculando la posición de dos puntos equidistantes al sistema de referencia local del objeto, luego la diferencia de estas dos posiciones genera un vector cuyo ángulo representa la orientación del objeto respecto al sistema de referencia del robot. Conociendo la posición de cierto objeto i y su orientación queda definida la pose de éste respecto al sistema de coordenadas del robot denominada por z_i^t en el instante de tiempo t.

4.2. Percepción

4.2.1. Perceptor de arcos

El perceptor de arco es el encargado de realizar la detección de un arco basándose en la información de color y en las características geométricas observadas en cada imagen de entrada. En la configuración actual de la cancha existen dos arcos, uno de ellos es celeste y el otro amarillo, por lo que en el sistema de visión existen dos perceptores de arco casi idénticos, cuya única diferencia consiste en el color del conjunto de regiones que reciben como entrada, uno de ellos recibe y procesa los conjuntos celestes y el otro los conjuntos amarillos. En la figura 4.6, se muestra un modelo realista del arco amarillo, que sirve para tener una idea general del objeto que se desea percibir. En esta figura, se establece una convención para definir un sistema de coordenadas relativo al arco.

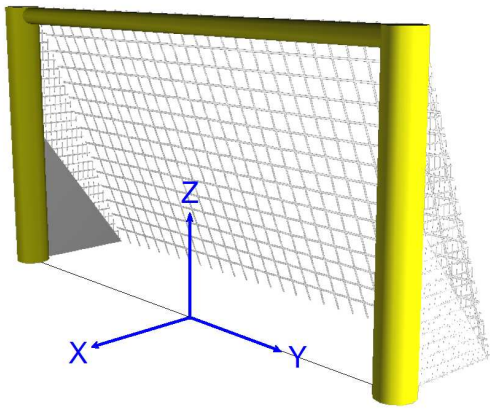


Figura 4.6: Modelo del arco real y sistema de referencia asociado.

Cada perceptor de arco recibe como entrada el conjunto de regiones candidatas del color

correspondiente. En cada perceptor de arco, las regiones candidatas se someten a una serie de análisis y evaluaciones en diferentes etapas, las cuales van eliminando progresivamente regiones que no cumplen satisfactoriamente ciertos requerimientos. Si alguna región logra superar todas las etapas, el perceptor entrega esa región como la detección de un arco. En caso de existir más de una región candidata que pase todas las evaluaciones, se entrega como detección la región de mayor tamaño. A continuación se describen las principales etapas del perceptor y sus correspondientes evaluaciones.

Estimación de pre-esquinas

Como se señaló en la etapa de generación de candidatos de la sección 2.5.1, a cada región candidata se le calculan ocho puntos clave en su frontera: superior, inferior, izquierdo, derecho, superior izquierdo, inferior izquierdo, superior derecho e inferior derecho. Inicialmente, los últimos cuatro puntos mencionados, se consideran como las esquinas del arco real. A estos puntos se les denomina pre-esquinas.

Filtros binarios

En el perceptor del arco propuesto, existen tres filtros que descartan inmediatamente las regiones que no cumplen con algunas características de tamaño y forma. Estos filtros se basan en la información de los puntos clave de cada región y, si bien resultan ser reglas simples, son muy eficaces al momento de hacer una primera selección de las mejores regiones candidatas a arcos, lo que permite reducir la cantidad de información procesada y con ello optimizar el proceso. En términos generales, los filtros binarios se caracterizan por evaluar características sencillas y por ser poco restrictivos, ya que su principal objetivo es descartar regiones que se alejan mucho de la forma real del objeto que se desea detectar.

La primera regla binaria utilizada en el perceptor del arco, es asegurar que el alto y el ancho de la región sean mayores que ciertos umbrales. Esto debido a que el arco real es visto como máximo a cierta distancia de la cámara, lo que determina un tamaño mínimo en que este se proyectará en la imagen. El umbral se estima observando el tamaño real de la región en la imagen, en los casos donde el arco se observa a la mayor distancia posible. La segunda regla evalúa que la relación de aspecto del arco, determinada por el cociente entre el alto y el ancho, se mantenga acotada en torno a la relación de aspecto real. Finalmente, la tercera regla binaria evalúa que la relación entre el área del rectángulo que inscribe a la región candidata y el área efectiva de la región, esté acotada en torno a la relación real.

Filtrado por horizonte visual

Utilizando el horizonte visual, es posible filtrar regiones candidatas de acuerdo a las relaciones físicas y espaciales que posee el arco con su entorno, específicamente su altura. El filtrado por horizonte visual se asegura que las esquinas superiores de la región candidata se encuentren arriba de la línea proyectada en la imagen y las esquinas inferiores abajo. Si esto no se cumple, la región candidata queda inmediatamente descartada.

Dimensiones de la región candidata

La etapa más importante en el proceso de percepción de un arco, consiste en la evaluación de la forma que tiene la región candidata en la imagen la que se compara con el patrón geométrico del arco real. Para realizar este análisis, se divide conceptualmente el arco en tres partes independientes: dos postes y un travesaño. Usando las pre-esquinas, se estima cuáles debieran ser éstos tres segmentos en la región candidata y se configuran tres conjuntos de líneas de escaneo, equi-espaciadas entre sí y perpendiculares a cada segmento. Cada línea de escaneo corresponde a un segmento que parte desde un punto externo al poste asociado y lo debe atravesar completamente. Para determinar el largo necesario y la orientación de cada línea, se utilizan las pre-esquinas calculadas anteriormente. Cada línea de escaneo es una máquina de estados cuya misión es encontrar los píxeles de transición entre el color del arco y cualquier otro color, usando la imagen segmentada en colores generada en la etapa de segmentación de colores ya descrita. Cada línea de escaneo tiene la misión de encontrar dos puntos de transición, que en el caso ideal corresponderían a puntos de la frontera de cada lado de cada poste asociado a la línea. Asumiendo esta geometría, se guardan las transiciones detectadas en este orden. En la figura 4.7, se muestra un arco y las líneas de escaneo asociadas, junto con los puntos de transición encontrados.

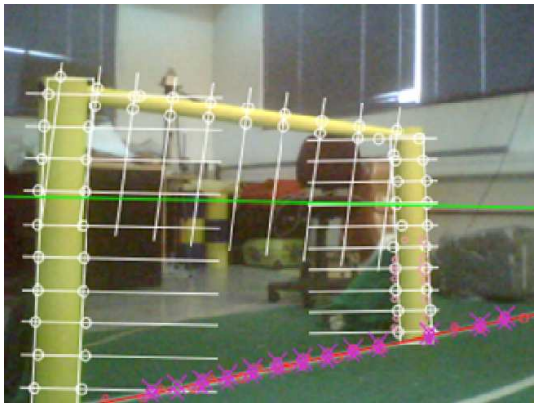


Figura 4.7: Las líneas de escaneo son la líneas blancas equidistantes entre sí. Los círculos blancos corresponden a los puntos de transición encontrados por las líneas de escaneo. La línea roja y los puntos rosados representan una detección de una línea del campo del juego.

Luego, se intenta definir el borde externo e interno de cada poste mediante la búsqueda de los modelos de líneas rectas que se ajusten a cada conjunto de puntos de transición. Esta búsqueda se hace mediante el algoritmo Ransac [5] el cual determina si un conjunto, o sub-conjunto, de puntos se ajustan a algún modelo en particular, en este caso modelos de rectas. De este modo, se pueden determinar los segmentos internos y externos de los postes representados por una región candidata, o bien, se puede descartar que éstos correspondan a postes, debido a la inconsistencia de los bordes al intentar ajustar una línea recta sobre ellos.

Utilizando las rectas externas de cada poste y las pre-esquinas inferiores, es posible

definir las esquinas inferiores del arco como el punto más cercano de la recta a la pre-esquina respectiva. Las esquinas superiores se estiman como las intersecciones de las rectas externas de los postes con la recta externa del travesaño. La distancia euclidiana entre estas esquinas determinan automáticamente el alto de cada poste, el largo del travesaño y sus respectivos anchos.

Un candidato a arco requiere, al menos, tener detectados uno de sus postes y una mínima parte del travesaño. Si esto no se cumple el candidato queda descartado.

Cálculo de pose

Una vez definidos el ancho y alto de cada poste es posible determinar su posición. Utilizando relaciones proporcionales entre el tamaño real de cada poste, los tamaños en la región candidata y geometría proyectiva, es posible determinar la posición de cada poste respecto al sistema de referencia de la cámara, y usando la información del estado de la cámara M es posible obtener la posición de cada poste respecto al sistema de referencia del robot. La posición del arco queda definida como el promedio algebraico de las posiciones de ambos postes. La diferencia de la posición de ambos postes genera un vector cuyo ángulo representa la orientación del arco. De esta forma queda determinada la pose del arco.

$$z_{arco} = \begin{bmatrix} (x_{posteizquierdo} + x_{postederecho})/2 \\ (y_{posteizquierdo} + y_{postederecho})/2 \\ atan(\frac{y_{posteizquierdo} - y_{postederecho}}{x_{posteizquierdo} - x_{postederecho}}) \end{bmatrix} \tag{4.30}$$

Arco incompleto

Si las regiones candidatas representan arcos incompletos, es decir uno de sus postes no se alcanza a ver en la imagen, aún es posible determinar la posición del poste faltante. Para esto es indispensable la detección del travesaño. Utilizando el método del cálculo de posición descrito en 4.1.4 se determina la posición de los extremos del segmento de travesaño detectado, la diferencia de ambas posiciones da a lugar a un vector cuyo ángulo θ_{arco} representa la orientación del arco de acuerdo al sistema de referencia del robot. Conociendo la orientación del arco θ_{arco}, la posición de uno de los postes $(x_{detectado}, y_{detectado})$ y el ancho real del arco A_{real} es posible determinar la posición del poste no detectado $(x_{nodetectado}, y_{nodetectado})$ utilizado la siguiente ecuación:

$$\begin{bmatrix} x_{nodetectado} \\ y_{nodetectado} \end{bmatrix} = \begin{bmatrix} x_{detectado} + A_{real} \cdot \sin(x_{detectado}) \\ y_{detectado} - A_{real} \cdot \cos(x_{detectado}) \end{bmatrix} \tag{4.31}$$

De esta forma se conocen las posiciones de ambos postes independientemente si se han detectado ambos o sólo uno de ellos.

Filtro por distancia entre postes

En caso de detectar ambos postes del arco se procede a calcular la distancia euclidiana entre sus posiciones, la que se compara con la distancia real entre los postes. Éstas distancias

deben ser similares bajo cierto rango definido, en caso de no ser así, se descarta la región candidata.

Filtro por pose

Una vez que se conoce la pose del arco, su distancia al robot nunca debe ser mayor que la medida diagonal de la cancha. En caso de ser así, el arco detectado corresponde a un falso positivo o se ha detectado un arco de otra cancha (caso frecuente en los escenarios RoboCup SPL), cualquier situación es desfavorable por lo que la región candidata se descarta.

Arco cercano

Cuando el robot se encuentra muy cercano a un arco, aproximadamente a menos de un metro, sólo logra ver secciones de alguno de los postes y no el travesaño por lo que resulta imposible detectar la pose del arco. Bajo esta condición un *behavior* del sistema de control se encarga de aumentar la inclinación de la cabeza. De esta manera es altamente probable que alguna de las esquinas superiores y una sección de travesaño aparezca lo suficiente para calcular la orientación del arco y la posición del poste visto. El perceptor es capaz de reconocer mediante el estado de la cámara M si la cabeza se encuentra buscando un arco cercano, si es así, el perceptor cambia los rangos y umbrales de los filtros binarios pues se espera ver el arco con un tamaño mayor al normal. Además, debido a la inclinación de la cabeza, el horizonte visual desaparece de la imagen por lo que la regla relacionada con él no se aplica. Por otra parte la distancia máxima del filtro por pose 4.2.1 se disminuye a aproximadamente un metro, descartando todos los falsos positivos que se pueden causar al no usar el filtro por horizonte visual.

Poste obstruido por otro robot

Es posible que algunos postes estén contenidos completamente en la imagen por lo que se les ha calculado la posición de acuerdo a su altura, pero pueden haber sido parcialmente obstruidos por robots en el juego, por lo que la altura calculada no corresponde a la real. Esta situación conlleva directamente a realizar un cálculo erróneo de la posición del poste. Para identificar esta situación, se evalúa la relación de aspecto del poste (altura/ancho), y se asegura que ésta sea menor que cierto umbral. Si no es así, entonces se calcula nuevamente su posición de acuerdo a su ancho. En caso de que el ancho no haya sido posible calcularlo debido a que el borde interno del poste no pudo ser modelado mediante RANSAC, el arco candidato al que pertenece el poste en cuestión, queda descartado.

4.2.2. Perceptor de línea

En el escenario de la SPL RoboCup podemos encontrar objetos relacionados con las marcas o líneas sobre la cancha. Estos objetos son: i) las mismas líneas de la cancha, ii) las esquinas, formadas por la intersección de dos líneas que terminan en dicha intersección y iii) las T-esquinas, formadas por dos líneas de la cancha que se intersectan, pero donde

sólo una de ellas termina en dicha intersección. Su percepción se basa en el alto contraste de intensidad que existe entre las líneas blancas y la cancha verde, de esta forma la información del color no es necesaria.

Detección de puntos de gradiente alto

El primer paso consiste en buscar puntos de alto contraste de intensidad a los cuales posteriormente se les calcula su gradiente de intensidad. Para encontrar estos puntos se define una serie de líneas de escaneo verticales y horizontales equi-espaciadas entre sí. La disposición de las líneas de escaneo se muestra en la figura 4.8 donde las líneas blancas corresponden a las líneas de escaneo y la línea verde al horizonte visual. Sólo se escanea el área bajo la línea del horizonte visual ya que se espera que las líneas se encuentren a nivel del suelo o baja altura, por otra parte esto ayuda a disminuir el tiempo de procesamiento al no es escanear toda la imagen y se disminuye la probabilidad de encontrar falsos positivos que puedan estar sobre la línea del horizonte visual.

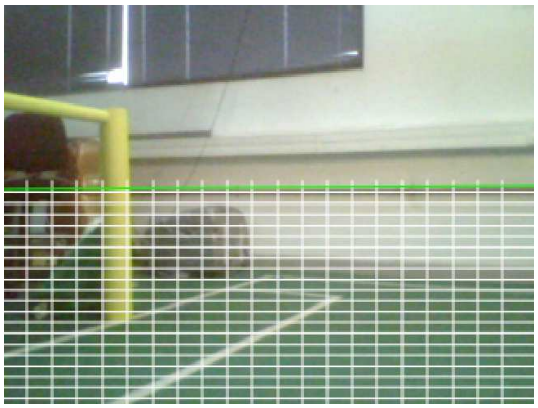


Figura 4.8: Grilla de escaneo para detección de puntos de gradientes altos. Las líneas blancas corresponden a las líneas de escaneo y la línea verde, al horizonte visual.

Cada línea de escaneo, a medida que avanza, calcula la diferencia de intensidad entre el píxel anterior y el siguiente. Si el valor absoluto de esta diferencia es mayor que cierto umbral se procede a calcular el gradiente de intensidad convolucionando la vecindad 3x3 del punto con las siguientes máscaras:

$$dx = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & +1 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.32}$$

$$dy = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & +1 & 0 \end{bmatrix} \tag{4.33}$$

Con esta operación, cada punto tendrá calculado su gradiente del cual se pueden desprender el módulo y la dirección. Al finalizar el proceso de escaneo se obtiene un conjunto

de puntos de alto gradiente de intensidad. La figura 4.9 muestra los puntos de alto gradiente detectados por las líneas de escaneo de la figura 4.8 y están representados por pequeños segmentos de líneas verdes, su longitud y orientación simbolizan la magnitud y orientación del gradiente respectivamente en ese punto.

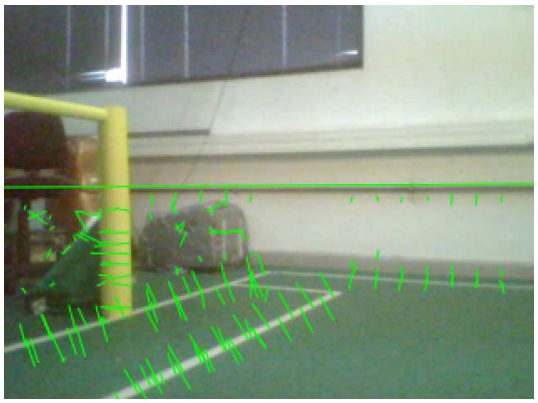


Figura 4.9: Puntos de altos gradientes detectados. El largo de cada segmento verde representa la magnitud del gradiente de la intensidad y la orientación la dirección de éste.

Perceptor de líneas simples

El perceptor de líneas se encarga de detectar segmentos de líneas presentes en la imagen. Para esto, utiliza el arreglo de puntos de alto gradiente generado en la etapa anterior. En primera instancia, el perceptor de líneas intenta agrupar los puntos en sub-arreglos bajo algunas condiciones de similitud entre puntos. Ésto se hace con el fin de no realizar una búsqueda tan exhaustiva de rectas dentro de un conjunto tan grande, lo que permite disminuir el tiempo de procesamiento y además disminuir la probabilidad de detectar falsos positivos.

Prácticamente siempre una línea de escaneo atraviesa una línea de la cancha capturada en la imagen, por lo que se espera que la línea de escaneo encuentre dos transiciones: cuando entra a la línea blanca y cuando sale. Es por esto que se construyen dos sub-arreglos iniciales: A para las transiciones de baja intensidad a alta intensidad y B para las transiciones de alta intensidad a baja intensidad. De cada sub-arreglo A y B se desprenderán más sub-arreglos hijos denominados a_i y b_i cuyos puntos que los componen poseen ciertas condiciones de similitud entre ellos. Una de éstas condiciones de similitud asume que los puntos de alto gradiente de una misma línea, poseen direcciones similares entre sí y, a la vez, perpendiculares a la línea (semejanza de dirección de gradiente). Otra condición asume que existirá una distancia prudente entre puntos de alto gradiente consecutivos en una misma línea (condición de distancia).

Posteriormente se verifica si cada sub-arreglo a_i y b_i se ajusta a un modelo de línea recta utilizando RANSAC. Si un sub-arreglo i es capaz de generar una recta entonces esta recta es almacenada como recta candidata. Si la recta r_i recién generada proviene de un arreglo a esta se almacena en un arreglo de rectas r_i^a y en caso contrario en r_i^b. Esto se hace con el

propósito de no perder la información acerca del tipo de transición.

Una línea de la cancha está formada por dos rectas, cada una de ellas representa sus bordes con la cancha. Cada borde define, naturalmente, una transición de intensidad, por lo que para determinar si una recta aparece en la imagen basta con comparar las rectas de r_i^a con r_i^b, si el par en cuestión satisface características de similitud basadas en ángulo y distancia es porque conforman una línea de la cancha. Esta comparación se hace de manera iterativa con cada par de rectas posibles entre r_i^a y r_i^b. La figura 4.10 muestra con segmentos rojos la detección final de dos líneas de la cancha.

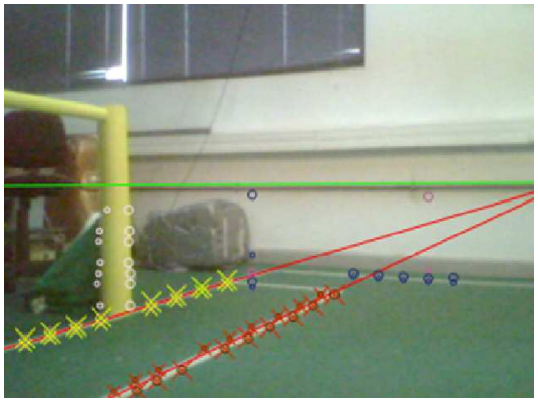


Figura 4.10: Detección de líneas en una imagen real. Los segmentos de línea rojos representan la detección de dos líneas de la cancha.

Es importante destacar que una línea de la cancha prácticamente nunca se captura completamente en la imagen, sino que sólo un segmento de ella. Es por esto que es imposible determinar con exactitud qué segmento de línea de la cancha se ha detectado. Dada esta situación, se procede a calcular la posición de los segmentos extremos respecto al sistema de coordenadas del robot utilizando el método descrito en 4.1.4. Con éstas posiciones se define una recta en el espacio del robot a la cual se calcula el punto más cercano de la recta al robot. Finalmente, éste punto corresponde a la posición de la línea detectada. No se considera pose ya que no se puede determinar desde qué lado de la línea se encuentra el robot con sólo información visual.

El perceptor de líneas no es capaz de identificar a qué línea de la cancha corresponde cada línea detectada, por lo que posteriormente se debe someter a un proceso especial que utilice la información de auto-localización para poder identificar a cuál línea del campo de juego corresponde.

Perceptor de esquinas y t-líneas

A partir de los arreglos r_i^a y r_i^b se construye un arreglo de líneas l_i formadas por pares de rectas que satisfacen condiciones de similitud menos restrictivas que las condiciones utilizadas en el perceptor de líneas descrito en el punto anterior.

Dentro de este arreglo l_i se buscan líneas que se intersecten entre sí dentro de la imagen.

Si la intersección de estos pares líneas en el espacio del robot es perpendicular, o al menos dentro de un rango, se aceptan como candidatos a esquina ó t-línea.

Para discernir si la intersección perpendicular encontrada es esquina o t-línea se configuran cuatro pequeñas líneas de escaneo. Éstas parten desde el punto de intersección y crecen en las cuatro direcciones de la intersección sobre cada línea. La finalidad de las líneas de escaneo es encontrar transiciones de alta a baja intensidad de los píxeles bajo ellas. Si se encuentran dos transiciones significa que dos líneas de la cancha se intersectan y se cortan en el punto de intersección por lo que la intersección corresponde a una esquina. Si se encuentra sólo una transición, es decir, una termina en la intersección y la otra continúa correspondería la intersección corresponde a una t-esquina. Cualquier otra situación de transiciones implica descartar la intersección como candidata. En las figuras 4.11 y 4.12 se muestran los modelos de esquinas y t-esquinas respectivamente y sus sistemas de referencia local definidos por convención.

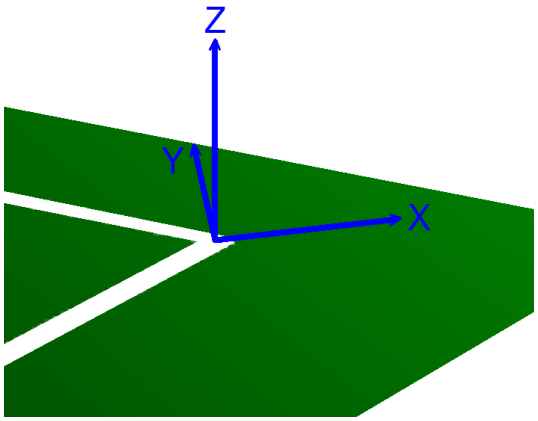


Figura 4.11: Modelo de de una esquina real y su sistema de referencia local.

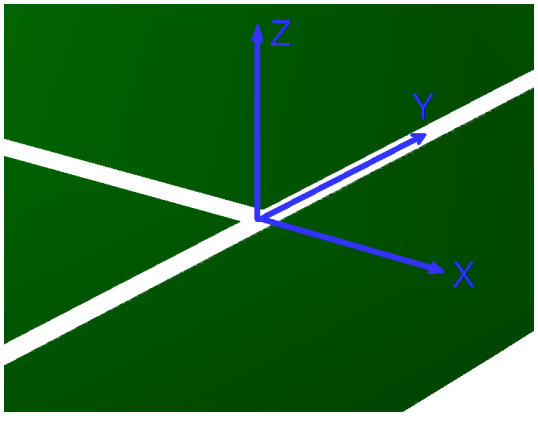


Figura 4.12: Modelo de una t-línea real y su sistema de referencia local.

La posición de éstos objetos se calcula proyectando el punto de intersección en el sis-

tema de referencia del robot mediante el método explicado en 4.1.4. La orientación de una esquina corresponde al promedio de las orientaciones de las líneas que la componen y la orientación de una t-línea corresponde a la orientación de la línea que se corta en la intersección.

4.3. Caracterización de la pose

El cálculo de la pose de los objetos se basa en la información del sensor de la cámara y del estado M de ésta. En un robot móvil humanoide, éstas informaciones poseen errores cuya fuente pueden ser: ruido del sensor de la cámara, ruido de los *encoders*, juegos de las articulaciones, imperfecciones del ensamblaje del robot, etc.

Por otra parte, la mayoría de los métodos de auto-localización y de estimación de estados necesitan conocer información sobre los errores asociados a las mediciones. De este modo es posible establecer un modelo observacional lo más realista posible.

Podemos definir las mediciones de pose, calculadas por el perceptor del objeto i que es detectado en el instante t, como una variable aleatoria Z_i^t de distribución normal de media μ_i^t y matriz de covarianza $\Sigma_{Z_i^t}$.

$$Z_i^t \sim N(\mu_i^t, \Sigma_{Z_i^t}) \tag{4.34}$$

Una medición puede definirse como un vector $\tilde{z}$ [a] cuya descomposición corresponde a:

$$\tilde{z} = \begin{pmatrix} \tilde{z}^x \\ \tilde{z}^y \\ \tilde{z}^\theta \end{pmatrix} \tag{4.35}$$

Éstas mediciones poseen dos tipos de error:

1. Error de *offset*. Éste corresponde a una componente de error constante para todas las mediciones de un mismo objeto en una misma pose. Caracterizar este error permite eliminarlo fácilmente de las mediciones haciéndolas más reales.

2. Dispersión. Este tipo de error está asociado a la precisión de las mediciones y puede ser caracterizado mediante una matriz de covarianzas.

Para caracterizar las salidas de los perceptores como una variable aleatoria de distribución normal, se realiza un análisis estadístico de cada perceptor. Para ello, se define una grilla G de celdas equi-espaciadas entre sí en el sistema de coordenadas del robot. Ésta grilla G representa un muestreo de todas las posibles poses del objeto en las cuales puede ser detectado.

[a] De aquí en adelante, asumiremos que cada medición, estimación o cálculo de pose se hacen en un instante t, para un objeto i en particular, por lo que no se mostrarán, explícitamente, los súper-índices t ni los sub-índices i. De ese modo se simplifican las notaciones a continuación.

Se procede a posicionar manualmente el objeto en cada celda c de la grilla G y se toman n mediciones $\tilde{z}$. El proceso anterior se realiza con el robot realizando el movimiento de balanceo de la caminata y moviendo la cabeza de lado a lado para considerar la variabilidad del proceso natural, pero siempre conservando una única pose. De este modo, cada celda de la grilla, tiene asociado un arreglo de mediciones de tamaño n, denominado por $\tilde{c}_z$:

$$\tilde{c}_z = \begin{pmatrix} \tilde{z}_1^x & \tilde{z}_2^x & \cdots & \tilde{z}_n^x \\ \tilde{z}_1^y & \tilde{z}_2^y & \cdots & \tilde{z}_n^y \\ \tilde{z}_1^\theta & \tilde{z}_2^\theta & \cdots & \tilde{z}_n^\theta \end{pmatrix} \tag{4.36}$$

En la ecuación anterior, $\tilde{c}_z$ corresponde al arreglo de mediciones. En el lado derecho, cada columna representa una medición $\tilde{z}$ expresada como vector, los sub-índices n corresponden al número de la medición y el súper-índice corresponde a la coordenada x, y ó θ. De este modo la grilla G corresponde a la base de datos para el análisis estadístico realizado a continuación.

Estimación de la media

Utilizando la base de datos descrita anteriormente, es posible determinar una función de corrección de pose que compense los errores de *offset* de cada medición $\tilde{z}$. Ésta función de corrección, denominada por $g(\cdot)$, es es propia de cada perceptor y debe utilizarse para calcular una medición corregida ($z = g(\tilde{z})$).

Definiendo $\overline{\tilde{c}}$ como el promedio de todas las mediciones $\tilde{z}$ asociadas a cada celda c de la grilla G tenemos que:

$$\overline{\tilde{c}} = \begin{pmatrix} \overline{\tilde{z}^x} \\ \overline{\tilde{z}^y} \\ \overline{\tilde{z}^\theta} \end{pmatrix} \tag{4.37}$$

Donde cada componente corresponde al promedio aritmético dado por:

$$\overline{\tilde{z}}^a = \frac{1}{n} \cdot \sum_{j=1}^{n} \tilde{z}_j^a \tag{4.38}$$

En la ecuación anterior a puede tomar los valores de cada componente x, y o θ. Luego podemos determinar la función $g(\cdot)$ mediante el ajuste de una función polinomial tal que minimice el error $e_g = |g(\overline{\tilde{c}}) - z^{real}|$ en cada celda.

Cabe destacar que la función $g(\cdot)$ tiene tres componentes, y su salida corresponde a una medición con el error de *offset* corregido:

$$z = g(\tilde{z}) = \begin{pmatrix} f_g^x(\tilde{z}^x) \\ f_g^y(\tilde{z}^y) \\ f_g^\theta(\tilde{z}^\theta) \end{pmatrix} \tag{4.39}$$

Cada una de éstas funciones f_g se muestran en la sección 5.3 con su respectivo gráfico.

Estimación de la covarianza

Para caracterizar el error de dispersión es posible determinar una matriz de covarianzas asociada a la variable aleatoria Z_i^t.

Utilizando la función de corrección $g(\cdot)$ de cada objeto i es posible eliminar el error de *offset* de todas las mediciones $\tilde{z}$ de la base de datos. Por lo que cada celda de la grilla quedaría como:

$$c = g(\tilde{c}) \tag{4.40}$$

Para los conjuntos de mediciones asociadas a cada celda c de la grilla G es posible calcular, de manera convencional, una matriz de covarianzas:

$$\Sigma_c = \begin{pmatrix} S_c^{xx} & S_c^{xy} & S_c^{x\theta} \\ S_c^{yx} & S_c^{yy} & S_c^{y\theta} \\ S_c^{\theta x} & S_c^{\theta y} & S_c^{\theta\theta} \end{pmatrix} \tag{4.41}$$

Las componentes del tipo S_c^{ab}, por su parte, se determinan como sigue:

$$S_c^{ab} = \frac{1}{n} \cdot \sum_{j=1}^{n} (z_j^a - \overline{z}_j^a) \cdot (z_j^b - \overline{z}_j^b) \tag{4.42}$$

Donde a y b pueden tomar los valores de x, y ó θ según sean los índices de cada componente de la matriz Σ_c.

Una vez determinadas las matrices de covarianza de cada una de las celdas c se debe generalizar para todo el espacio de poses posibles. Para ello, la estrategia seleccionada consiste en estimar una función $cov(\cdot)$, mediante el ajuste de funciones polinomiales para cada una de las componentes de la matriz de covarianzas:

$$cov(z) = \begin{pmatrix} S_z^{xx} & S_z^{xy} & S_z^{x\theta} \\ S_z^{yx} & S_z^{yy} & S_z^{y\theta} \\ S_z^{\theta x} & S_z^{\theta y} & S_z^{\theta\theta} \end{pmatrix} = \begin{pmatrix} f_{cov}^{xx}(z) & f_{cov}^{xy}(z) & f_{cov}^{x\theta}(z) \\ f_{cov}^{yx}(z) & f_{cov}^{yy}(z) & f_{cov}^{y\theta}(z) \\ f_{cov}^{\theta x}(z) & f_{cov}^{\theta y}(z) & f_{cov}^{\theta\theta}(z) \end{pmatrix} \tag{4.43}$$

Ésta metodología permite tener matrices de covarianza continuas, representativas y al mismo tiempo, rápidas de calcular para cualquier punto del espacio de poses.

Con esto, la salida de cada perceptor es una medición de la pose sin error de $offset$ y con una matriz de covarianzas asociada. Ésto convierte la salida de cada perceptor en una variable aleatoria normal Z de media z y matriz de covarianza $\Sigma_z = cov(z)$.

4.4. Identificación de objetos ambiguos

En la sección 1.3 podemos apreciar la distribución de los objetos líneas: líneas simples, esquinas y t-líneas en la cancha y su correspondiente número identificador asignado por

convención para diferenciar entre sí los objetos de una misma clase. El perceptor de líneas simples puede detectar si están presentes en la imagen y calcula la pose de éstas respecto al sistema de coordenadas del robot. Sin embargo, no es capaz de identificar a qué línea simple de la cancha corresponde debido a que son visualmente idénticas entre sí, o en otras palabras, no existe una característica visual que permita discernir a qué línea simple corresponde si la observamos sin ningún otro objeto en la imagen. Sucede exactamente lo mismo para los perceptores de esquinas y t-líneas pero no para el perceptor del arco ya que éste se identifica fácilmente por su color.

La implementación de la detección de los objetos líneas tiene como objetivo principal ayudar en el proceso de auto-localización del robot, ya que los arcos pocas veces aparecen en el campo visual de la cámara durante el juego. Lamentablemente, éstas detecciones no poseen una identificación que ayude al proceso de asociación de datos del algoritmo de auto-localización. Es por esto que surge la necesidad de implementar la identificación de objetos ambiguos basándose en la misma auto-localización del robot, siempre y cuando, ésta posea una medida de confidencia mínima.

Se puede pensar que, al necesitar que el robot esté auto-localizado, la detección e identificación de los objetos línea no presente utilidad real en la misma auto-localización. En la práctica no resulta ser así, frecuentemente entre percepciones de arco puede pasar un tiempo considerable. Durante éste tiempo el error odométrico acumulado causa grandes errores en la estimación del estado de la auto-localización. Sin embargo, durante este tiempo los objetos línea son frecuentemente vistos por su disposición en toda la cancha por lo que su detección e identificación pueden corregir continuamente la auto-localización evitando que el error aumente hasta el punto donde ya no es posible identificar los objetos línea. En otras palabras, la identificación ayuda en gran parte a mantener la localización durante los intervalos de tiempo entre percepciones de arcos.

El proceso de identificación consiste básicamente en la comparación del objeto visto con todos los objetos de la misma clase presentes en la cancha, seleccionando el objeto de mayor similitud en la pose. El primer paso del la identificación consiste en transformar la pose del objeto detectado desde el sistema de referencia del robot *rob* al sistema de referencia global *glo* utilizando la auto-localización del robot. De esta forma, la comparación de poses se hace en el mismo sistema de referencia global. Como se describió en la sección 4.3 los estados de los objetos detectados y la auto-localización se manipulan como variables aleatorias de distribución normal por lo que la transformación de sistemas de coordenadas posee dos partes: i) la transformación de la media o pose en sí y ii) el cálculo de la covarianza del resultado de la transformación.

La transformación de la pose del objeto detectado queda definida por una función f que se muestra a continuación:

$$\begin{bmatrix} z_x^{glo} \\ z_y^{glo} \end{bmatrix} = R(L_\theta^{glo}) \cdot \begin{bmatrix} z_x^{rob} \\ z_y^{rob} \end{bmatrix} + \begin{bmatrix} L_x^{glo} \\ L_y^{glo} \end{bmatrix} \tag{4.44}$$

$$z_\theta^{glo} = z_\theta^{rob} + L_\theta^{glo} \tag{4.45}$$

En las ecuaciones anteriores, $[z_x^{rob}, z_y^{rob}, z_\theta^{rob}]^T$ corresponde a la pose del objeto detectado

respecto al sistema de referencia del robot. $[L_x^{glo}, L_y^{glo}, L_\theta^{glo}]^T$ corresponde a la pose de la auto-localización definida en el sistema de referencia global. La nueva pose, $[z_x^{glo}, z_y^{glo}, z_\theta^{glo}]^T$ del objeto detectado, está definida en el sistema de referencia global.

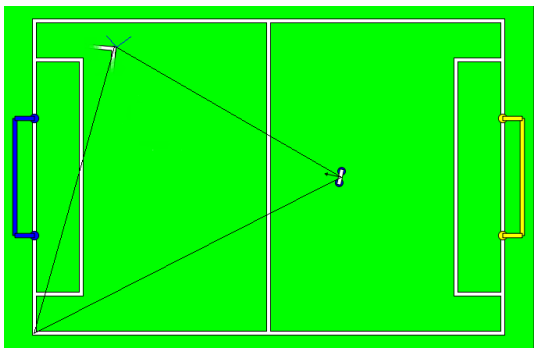


Figura 4.13: Esquina detectada, proyectada en el sistema de referencia global a través de la localización del robot.

El cálculo de la covarianza $\Sigma_{z^{glo}}$ asociada a la pose z^{glo} se obtiene propagando linealmente la covarianza del objeto detectado $\Sigma_{z^{rob}}$ y la covarianza de la localización $\Sigma_{L^{glo}}$. Para esto, se debe calcular el jacobiano de la función f respecto a $[z_x^{rob}, z_y^{rob}, z_\theta^{rob}, L_x^{glo}, L_y^{glo}, L_\theta^{glo}]$ resultando una matriz jacobiana J_f de 3x6:

$$J_f = \begin{bmatrix} \frac{f_x}{z_x^{rob}} & \frac{f_x}{z_y^{rob}} & \frac{f_x}{z_\theta^{rob}} & \frac{f_x}{L_x^{rob}} & \frac{f_x}{L_y^{rob}} & \frac{f_x}{L_\theta^{rob}} \\ \frac{f_y}{z_x^{rob}} & \frac{f_y}{z_y^{rob}} & \frac{f_y}{z_\theta^{rob}} & \frac{f_y}{L_x^{rob}} & \frac{f_y}{L_y^{rob}} & \frac{f_y}{L_\theta^{rob}} \\ \frac{f_\theta}{z_x^{rob}} & \frac{f_\theta}{z_y^{rob}} & \frac{f_\theta}{z_\theta^{rob}} & \frac{f_\theta}{L_x^{rob}} & \frac{f_\theta}{L_y^{rob}} & \frac{f_\theta}{L_\theta^{rob}} \end{bmatrix} \tag{4.46}$$

Calculado el jacobiano J_f este se premultiplica y postmultiplica a la concatenación de las matrices de covarianzas $\Sigma_{z^{rob}}$ y $\Sigma_{L^{glo}}$:

$$\Sigma_{z^{glo}} = J_f \cdot \begin{bmatrix} \Sigma_{z^{rob}} & 0 \\ 0 & \Sigma_{L^{glo}} \end{bmatrix} \cdot J_f^T \tag{4.47}$$

Una vez obtenidos z^{glo} y $\Sigma_{z^{glo}}$, se procede a realizar la comparación con cada posible objeto i de la cancha. Esta comparación consiste en la diferencia de las poses del objeto detectado z^{glo} y del objeto i de la cancha q_i^{glo}:

$$e_i^{glo} = |z^{glo} - q_i^{glo}| \tag{4.48}$$

En cada comparación i es posible determinar la innovación del error $I_{e_i^{glo}}$:

$$I_{e_i^{glo}} = e_i^{gloT} \cdot \Sigma_{z^{glo}}^{-1} \cdot e_i^{glo} \tag{4.49}$$

Y luego transformar esta innovación a una probabilidad según:

$$P_i(Id_{z_{rob}} = i) = e^{-I_{e_i^{glo}}} \tag{4.50}$$

Donde $P_i(Id_{z_{rob}} = i)$ es la probabilidad de que el número identificador del objeto línea detectado sea igual al objeto i de la cancha. Hasta el momento tenemos una probabilidad P_i asociada a cada objeto i de la cancha. P_i decrece a medida que el error e_i^{glo} aumenta, la sensibilidad de éste decrecimiento es inversamente proporcional a la matriz de covarianza $\Sigma_{z^{glo}}$. Mientras más grande es $\Sigma_{z^{glo}}$, P_i es menos sensible al error e_i^{glo}, ésto puede producir que los valores de P_i sean similares entre diferentes objetos i lo que dificultaría la identificación. En caso contrario, cuando $\Sigma_{z^{glo}}$ es pequeña, la sensibilidad de P_i aumenta frente al error e_i^{glo}. Ésto tiene sentido, ya que si $\Sigma_{z^{glo}}$ es pequeña significa que existe una alta certeza de la auto-localización y, por lo tanto, los valores de P_i tendrán grandes diferencias entre objetos i lo que facilitaría la identificación.

Una vez evaluados todos los P_i, para cada objeto i de la cancha, se analizan dos condiciones sobre éstos valores. La primera corresponde a exigir que el máximo valor de P supere cierto umbral para asegurarse de que la auto-localización es suficientemente *buena*. La otra es exigir que, el segundo mayor valor, esté por debajo de otro umbral, en otras palabras, debiese existir una mínima diferencia de P entre el primer mayor valor y el segundo mayor, de ésta forma se minimiza la probabilidad cometer errores en la identificación.

Finalmente, cumplidos los umbrales mencionados, se puede determinar que el objeto i de la cancha con mayor P corresponde al objeto detectado en cuestión y puede ser identificado por i. En caso contrario, el objeto no se identifica. Es importante mencionar que en el ciclo de operación del sistema de auto-localización del robot, los objetos líneas, esquinas y t-líneas, refuerzan la auto-localización, sí y sólo sí, ésta ya se encuentra *bien* auto-localizado. Si por cualquier eventualidad el robot móvil de des-localiza, el sistema deberá esperar a observar un arco nuevamente para poder auto-localizarse correctamente, y así volver a identificar objetos líneas.

En la figura 4.13, se muestra una posible detección visual de una esquina. Utilizando la localización del robot como un dato, dicha detección se proyecta en el sistema de referencia global. Luego, el sistema evaluará la relación espacial de dicha esquina proyectada en el sistema de referencia global, con cada una de las esquinas reales de la cancha, donde claramente la esquina más cercana es la que tendrá mayor probabilidad de corresponder a la esquina percibida.

Capítulo 5

Resultados

Contenido

5.1. Estado de la cámara

5.1.1. Horizonte visual

El horizonte visual es utilizado en varias etapas y de diferentes formas en el sistema de visión propuesto. Dado que representa una información espacial muy importante resulta ser muy útil para definir reglas en detección de objetos. A continuación se detallan las aplicaciones del horizonte visual.

Horizonte visual como referencia para calibrar el estado de la cámara

En la sección 5.1.2 se muestra cómo la línea de horizonte visual en la imagen es utilizada como referencia para calibrar el estado de la cámara.

Filtro de horizonte visual en el perceptor de arcos

En la descripción del perceptor de arcos (sección 4.2.1) se describe el uso de éste filtro. En la figura 5.4 se aprecia un ejemplo de un falso positivo que no cumple las reglas de este filtro, sus esquinas inferiores se encuentran sobre la línea del horizonte visual. En en la curva ROC de la figura 5.3, se muestra la notable disminución de los falsos positivos luego del uso del filtro de horizonte visual.

Configuración de líneas de detección de altos gradientes en los perceptores de objetos línea

En la figura 4.8 se muestra la disposición de las líneas de detección de altos gradientes que sólo se configuran bajo la línea del horizonte. En la sección 4.2.2 se encuentra un

detalle de esta etapa del perceptor de líneas. La tabla 5.1 nos muestra los tiempos de procesamiento en la detección de altos gradientes en dos casos: (i) sin el uso de horizonte, es decir, configurando las líneas de detección en toda la imagen y (ii) utilizando el horizonte como límite para estas líneas. Como se puede apreciar el tiempo de detección de gradientes disminuye, en promedio, a más de la mitad en el caso (ii).

	Imagen completa	Sólo bajo el horizonte
Tiempo	45 ms	22 ms

Cuadro 5.1: Tabla comparativa de tiempo de búsqueda de gradientes altos en la imagen completa y sólo bajo el horizonte.

Otras aplicaciones

También se puede hacer uso del horizonte visual para generar un filtro que opere de manera similar al filtro por horizonte del arco para perceptores de otros objetos como el de la pelota, beacons u otros objetos de interés del entorno y que no se encuentran detallados en esta tesis.

5.1.2. Calibración del estado de la cámara

Los resultados de la calibración de la cámara se muestran en dos situaciones diferentes. En la situación 1, el robot observa el arco celeste y se encuentra orientado hacia él capturando las imágenes de la figura 5.1. En la situación 2, el robot gira la cabeza en torno al eje z y observa las imágenes de la figura 5.2. En ambas situaciones, el robot tiene una pose global de (550.0, 200.0, $\pi/2$) y está de pie. A continuación se muestran dos resultados que son producto de la correcta calibración del estado M de la cámara.

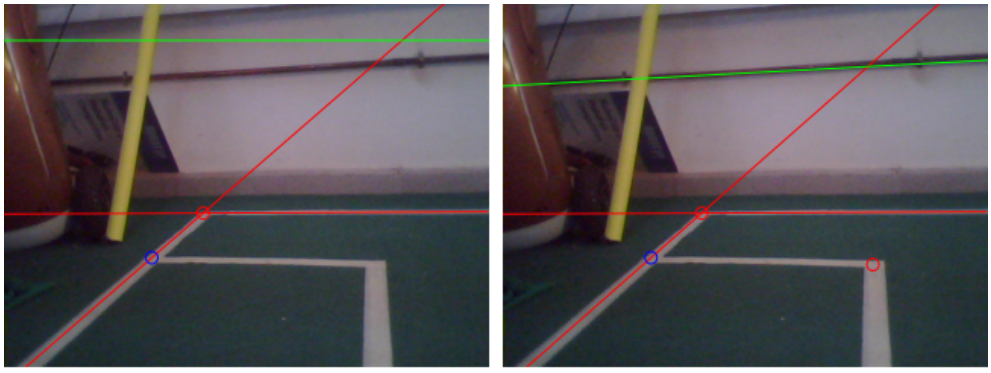
Corrección de la línea del horizonte visual

Tal como se describe en la sección 4.1.2 el horizonte visual está a la altura de la cámara del robot y es paralelo al plano del suelo. En las imágenes de las figuras 5.1 y 5.2, se puede apreciar el horizonte visual representado por una recta verde. En la imagen *a* de la figura 5.1, el horizonte se encuentra levemente por debajo de la altura de la cámara. En la imagen *b* de la figura 5.1, se aprecia el horizonte en su altura correcta luego de realizar el proceso de calibración del estado M de la cámara que se describe en la sección 4.1.3.

En las imágenes de la figura 5.2 se puede apreciar una cañería de cobre que se encuentra a la altura de la cámara del robot y está alineada con el plano del suelo. Dadas éstas condiciones de la cañería, es posible utilizarla como referencia para calibrar el estado M de la cámara. La imagen *a* de la figura 5.2 muestra el horizonte visual desalineado con la cañería de cobre, la imagen *b* de la figura 5.2 muestra el horizonte visual alineado con la cañería luego de haber realizado el proceso de calibración. Este ejemplo muestra claramente la necesidad de un sistema de calibración del estado M de la cámara.

a b

Figura 5.1: Situación 1. En ambas imágenes se pueden apreciar el horizonte visual, representado por una línea recta verde, y la detección de las líneas de la cancha representada por líneas rectas rojas. En la imagen *a*, el proceso de calibración del estado *M* de la cámara no se ha realizado y en la imagen *b* si. La calibración se puede apreciar en el cambio de altura del horizonte visual. En la imagen *b*, el horizonte visual se encuentra más cercano a la altura de la cámara del robot.



a b

Figura 5.2: Situación 2. En ambas imágenes se pueden apreciar: el horizonte visual (línea recta verde), la detección de dos líneas de la cancha (líneas rectas rojas), la detección de esquinas (círculos rojos) y la detección de t-líneas (círculos azules). En la imagen *a*, el proceso de calibración del estado *M* de la cámara no se ha realizado y en la imagen *b* si. La calibración se puede apreciar en el cambio de altura y alineación con la cañería de cobre que es utilizada como referencia del horizonte real

La calibración del estado *M* de la cámara, no sólo ayuda a corregir el horizonte visual, sino que también corrige el cálculo de la posición de los objetos líneas, esquinas y t-líneas, ya que éste depende directamente de *M*.

Corrección del cálculo de la posición de objetos

En la tabla 5.2 se muestran las posiciones de las líneas de la cancha detectadas en las imágenes de la figura 5.1. Luego de realizar la calibración del estado M de la cámara, la posición de ambas líneas se asemejan más a su posición real.

En la tabla 5.3 se muestran las posiciones de los objetos detectados en las imágenes de la figura 5.2. La calibración del estado M de la cámara resulta beneficioso ya que, para todos los casos, mejora el cálculo de la posición acercándolo al real.

Se espera que los pequeños errores restantes sean corregido con las funciones de corrección mostradas en 5.3.

Objeto	Sin calibración		Con calibración		Posición Real	
	x	y	x	y	x	y
Línea 1	264.28	5.17	250.01	6.53	250.0	0.0
Línea 2	597.04	5.01	521.71	6.08	480.0	0.0

Cuadro 5.2: Tabla comparativa de las posiciones de las líneas detectadas en las imágenes de la figura 5.1. Se puede apreciar la disminución del error luego de la calibración del estado M de la cámara.

Objeto	Sin calibración		Con calibración		Posición Real	
	x	y	x	y	x	y
Línea 1	-54.66	2.88	-59.6	0.34	-60.0	0.0
Línea 2	-36.23	167.87	-0.64	206.28	0.0	200.0
Esquina	-46.07	165.74	-58.44	206.1	-60.0	200.0
t-línea	-48.05	128.168	-58.73	154.23	-60.0	150.0

Cuadro 5.3: Tabla comparativa de las posiciones de los objetos detectados en las imágenes de la figura 5.1. Se puede apreciar la disminución del error luego de la calibración del estado M de la cámara.

5.1.3. Cálculo de la posición de un objeto

Las tablas 5.2 y 5.3 dan fe del buen funcionamiento del cálculo de la posición de objetos descrito en 4.1.4.

5.2. Desempeño de perceptores de objetos

Para evaluar el desempeño de los perceptores implementados se grabó un video (imágenes y *encoders*) de aproximadamente 4000 segundos con el robot moviendo la cabeza cons-

tantemente de izquierda a derecha. Además, el robot se posicionó manualmente en tres diferentes poses globales durante la grabación con el fin de maximizar la variabilidad de objetos proyectados en la imagen. A partir de éste video es posible contrastar las detecciones de los perceptores en el video versus una referencia que determine automáticamente qué objetos, en teoría, debiesen detectarse. Para obtener esta referencia se procedió a desarrollar un módulo que utiliza la información de las tres poses del robot para determinar la pose de todos los objetos de la cancha (arcos, líneas y t-esquinas) respecto al sistema de referencia del robot. Luego, para determinar si los objetos se encuentran en el campo visual de la cámara se premultiplica las poses transformadas de cada objeto por la matriz inversa del estado de la cámara M^{-1}. De este modo se obtienen las poses de los objetos respecto al sistema de referencia de la cámara. Finalmente, utilizando el modelo *Pin Hole* [1] es posible determinar el píxel en la imagen en donde se proyecta la pose del objeto:

$$x = -\frac{f \cdot Y}{X} y = -\frac{f \cdot Z}{X} \tag{5.1}$$

En la ecuación 5.2, X, Y y Z corresponden a las coordenadas del objeto en el sistema de referencia de la cámara. x e y corresponde a la columna y fila de la imagen donde se proyecta el punto (X, Y, Z). Finalmente, se debe verificar que las coordenadas (x, y) se encuentren dentro de los límites de la imagen, y además, que se proyecte sobre la imagen un área mínima tal que pueda ser detectado.

Con este módulo se obtiene la referencia o también llamado *Ground Truth* con el cual se contrastan las detecciones entregadas por los perceptores. Con ésta información es posible calcular las tasas de verdaderos positivos, falsos positivos y falsos negativos para cada uno de los perceptores.

Una vez obtenido el *Ground Truth* se procede a generar curvas ROC para representar el desempeño de cada perceptor. En cada una de ellas se ha variado como punto de funcionamiento el error mínimo aceptable en el algoritmo de regresión de rectas RANSAC, que es utilizado para la definición de bordes de los postes de los arcos, como para la construcción de las líneas de la cancha. Se ha decidido analizar el desempeño en función de éste parámetro pues ambos perceptores son bastantes sensibles a él, ya que la detección y armado de rectas es determinante para la detección de objetos.

5.2.1. Curvas ROC percepción de arco

El perceptor de arcos posee varios filtros dentro de los cuales existen tres filtros muy importantes para detectar arcos, éstos son: filtro de horizonte, filtro de distancia y filtro de distancia entre postes. A continuación se muestra el análisis de desempeño de cada uno de éstos tres filtros.

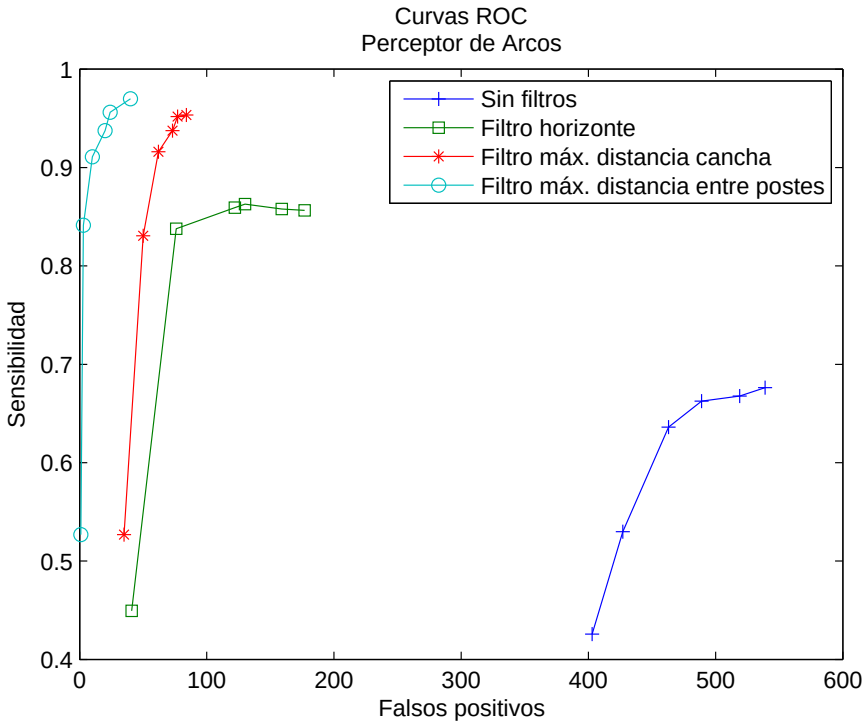


Figura 5.3: Curva ROC del perceptor del arco. Cada curva posee seis puntos de operación que representan los valores de píxeles de error mínimo para el algoritmo RANSAC. Éstos valores son: 1.0, 3.0, 5.0, 7.0, 9.0 y 11.0 pixeles, y están representados en cada curva por una marca.

Caso 1: sin filtros

En la figura 5.3 se puede ver fácilmente que la curva que representa el desempeño sin filtros muestra una alta cantidad de falsos positivos para cualquier punto de operación. El perceptor de arcos, una vez que determina una detección, deshecha automáticamente otros posibles candidatos. Es por ésto que la aparición de falsos positivos disminuye la sensibilidad. En otras palabras, cuando un falso positivo aparece es imposible detectar un arco que se proyecta correctamente en el campo visual.

Caso 2: con filtro de horizonte visual

La curva con filtro de horizonte muestra la disminución radical de los falsos positivos y, por consiguiente, el aumento de la sensibilidad. Ahora se han eliminado falsos positivos generados por ventanas, persianas o zonas segmentadas con color azul.

En la imagen *a* de la figura 5.4 se muestra la detección de un falso positivo cuando no se utiliza el filtro por horizonte visual. En la imagen *b* de la misma figura se aprecia la región segmentada que genera el falso positivo, ésta posee mayor área que la del arco correcto, es por ésto que es el primer candidato procesado y aceptado (recordemos que las

regiones candidatas son ordenadas de mayor a menor área). Con la utilización del filtro de horizonte visual éstos falsos positivos se eliminan inmediatamente al inicio del proceso del perceptor, por lo que el arco correcto continúa como primer candidato y así finalmente puede ser detectado.

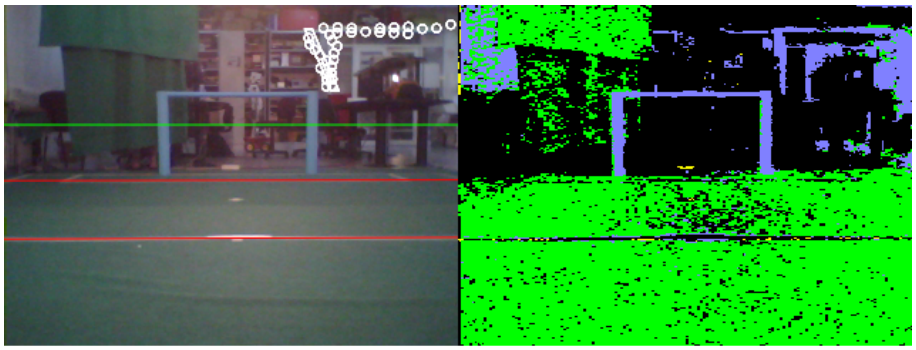


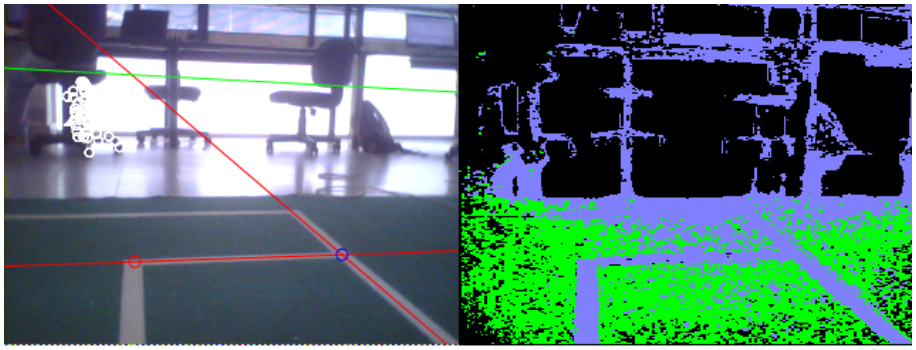
Figura 5.4: Falso positivo de arco sobre línea del horizonte. *a*, imagen original y la detección de un falso positivo sin la utilizanción del filtrado por horizonte visual. *b*, imagen segmentada donde se aprecia el mayor tamaño del falso positivo que el del arco correcto, por éste motivo no se detecta el arco correcto.

Caso 3: con filtro de distancia

Éste filtro elimina una cantidad importante de falsos positivos que tienen forma de pequeños arcos, y a los cuáles se les ha calculado una pose que está a una distancia mayor que la diagonal de la cancha. También consideran cuando el robot eleva la cabeza para buscar esquinas de arcos cercanos disminuyendo a un metro la distancia máxima permitida. La curva de la figura 5.3 que representa este caso, muestra la disminución de los falsos positivos y también un notable aumento de la sensibilidad.

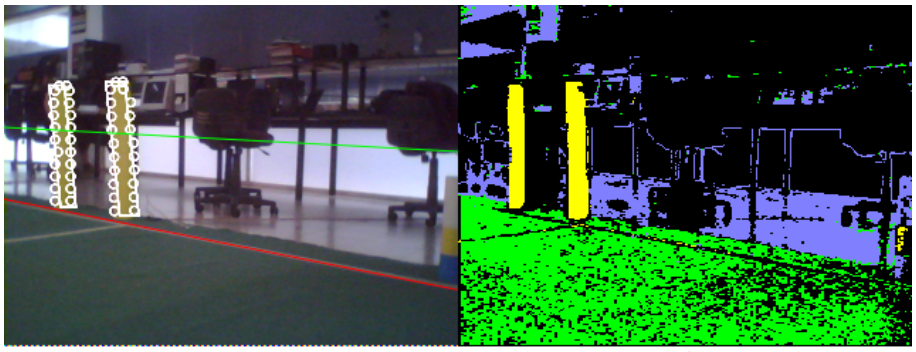
Caso 4: con filtro de distancia entre postes

Finalmente la curva que muestra este caso es la que posee la mayor área bajo la curva lo que representa el caso óptimo del perceptor. Los falsos positivos son mínimos y la sensibilidad es la más alta de todas. En el video este filtro ha eliminado los falsos positivos generados por dos postes puestos a una distancia de aproximadamente 70 cm entre ellos a un costado de la cancha por lo que aparece en algunas imágenes en el video de prueba. La figura 5.6 muestra una de éstas imágenes con el falso positivo descrito, al habilitar el filtro de distancia entre postes este falso positivo queda descartado. Cabe destacar que es frecuente que en situaciones de juego real en la competencia RoboCup los pantalones del público pueden segmentarse azul y tener forma de arco, ésta es la principal razón por la cual se implementó este filtro.



a b

Figura 5.5: Falso positivo de arco cuya posición se encuentra a 13.02 metros del robot. *a*, los círculos blancos muestran la detección de un falso positivo de arco al cual se le ha calculado una distancia de 13.02 metros respecto al sistema de referencia del robot, en esta imagen no se ha usado el filtro por distancia en el perceptor de arcos. *b*, se muestra la segmentación de la imagen y la alta probabilidad de encontrar regiones con forma de arco.



a b

Figura 5.6: Falso positivo de arco cuyos postes están separados por una distancia de 70 cm.. La imagen *a* muestra los círculos blancos muestran la detección de un falso positivo de arco generado por dos postes puestos en un borde de la cancha a 70 cm entre sí. La imagen *b* muestra la segmentación de la imagen y el parecido que tiene la segmentación de los postes amarillos con un arco real.

Éstas curvas ROC nos confirman estadísticamente las ventajas de los filtros aplicados. Por otra parte observando la mejor curva se ha logrado determinar, metodológicamente, el valor óptimo de error máximo permitido para el algoritmo RANSAC, en éste caso 7.0.

La tabla 5.4 muestra los porcentajes de verdaderos positivos, falsos positivos y falsos negativos para un min_{error} = 7.0 pixeles.

Indicador	Verdaderos Positivos	Falsos Positivos
Detecciones(%)	93.8%	1.44%

Cuadro 5.4: Tabla de detecciones del perceptor del arco utilizando el punto de operación min_{error} = 7.0. pixeles

5.2.2. Curvas ROC percepción de líneas

En el caso de la percepción de líneas, esquinas y t-líneas se ha decidido a analizar dos casos: sin filtro de distancia y con filtro de distancia. Éste filtro consiste en proyectar todos puntos de alto gradiente sobre el sistema de referencia del robot utilizando el método descrito en 4.1.4 asumiendo que están a altura cero. Posteriormente, se descartan todos aquellos que se encuentren a una distancia mayor a la diagonal de la cancha.

La figura 5.7 muestra las curvas ROC de ambos casos en seis diferentes puntos de operación. Éstos puntos de operación corresponden número de pixeles de error mínimo aceptado en el algoritmo RANSAC de las líneas, y son: 1.0, 3.0, 5.0, 7.0, 9.0 y 11.0 pixeles.

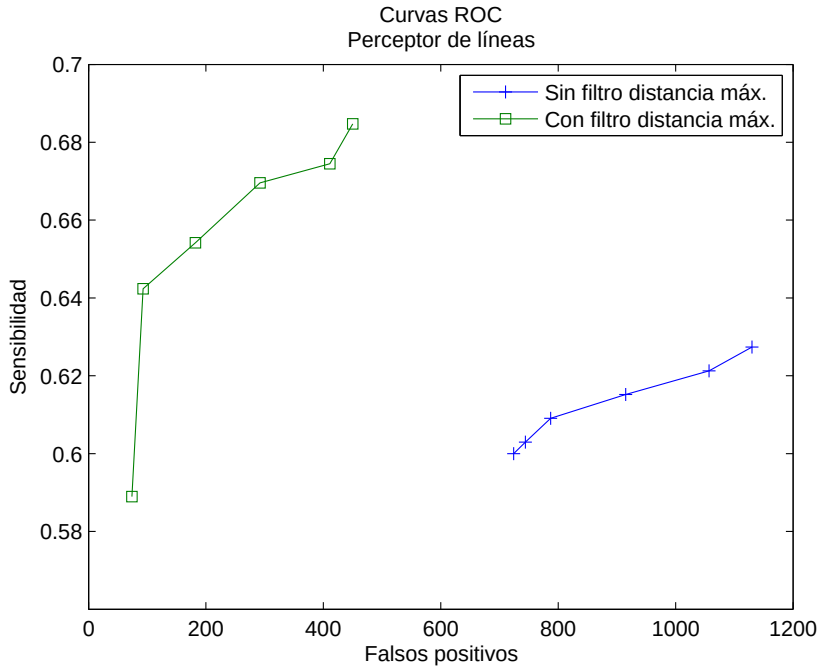


Figura 5.7: Curva ROC del perceptor del líneas. Cada curva posee seis puntos de operación que representan los valores de píxeles de error mínimo para el algoritmo RANSAC. Éstos valores son: 1.0, 3.0, 5.0, 7.0, 9.0 y 11.0 pixeles, y están representados en cada curva por una marca.

La figura 5.7 muestra el desempeño con y sin filtro de distancia aplicado sobre los

puntos de alto gradiente. Al comparar ambas curvas, se aprecia una gran disminución de los falsos positivos y a la vez un aumento de la sensibilidad del perceptor debido a una situación muy similar en el perceptor de arcos: los falsos positivos ocupan la máxima cantidad de candidatos admisibles en cierta etapa del perceptor imposibilitando la detección de líneas correctas.

Al observar la curva que representa el desempeño con el filtro de distancia habilitado podemos deducir, metodológicamente, que el mejor valor para el mínimo error aceptado en el algoritmo RANSAC es igual a 3.0 pixeles.

Indicador	Verdaderos Positivos	Falsos Positivos
Detecciones(%)	64.2%	28.4%

Cuadro 5.5: Tabla de detecciones del perceptor de líneas utilizando el punto de operación $min_{error} = 3{,}0$.

La tabla 5.5 muestra los porcentajes de verdaderos positivos, falsos positivos y falsos negativos para un min_{error} = 3.0.

Filtro de distancia para objetos línea

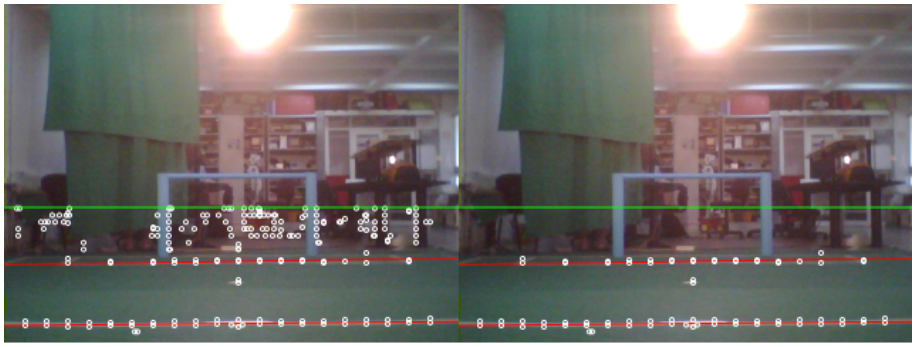


Figura 5.8: Eliminación de puntos fuera de la cancha (determinado por distancia). La imagen *a* muestra todos los puntos de alto gradiente que son candidatos a para formar líneas de la cancha. Se puede apreciar la gran cantidad de puntos que no pertenecen a líneas de cancha. La imagen *b* muestra cómo el filtro de distancia elimina casi todos los puntos fuera de la cancha.

En la figura 5.8 podemos apreciar cómo el filtro de distancia elimina puntos de altos gradiente que no corresponden a transiciones en la cancha o que se encuentran a una distancia mayor que el largo de la cancha. De esta forma, la aplicación del filtro disminuye enormemente la probabilidad de encontrar falsos positivos fuera de la cancha.

5.2.3. Curvas ROC percepción de esquinas y t-líneas

Las figuras 5.9 y 5.10 muestran las curvas ROC de desempeño del perceptor de esquinas y t-líneas respectivamente. Ambas figuras muestran dos casos a la vez: sin filtro de distancia y con filtro de distancia. Al igual que en los casos anteriores, las curvas ROC fueron determinadas en seis puntos de operación variando los pixeles de error mínimo aceptado en el algoritmo RANSAC, estos valores corresponden a: 1.0, 3.0, 5.0, 7.0, 9.0 y 11.0 píxeles.

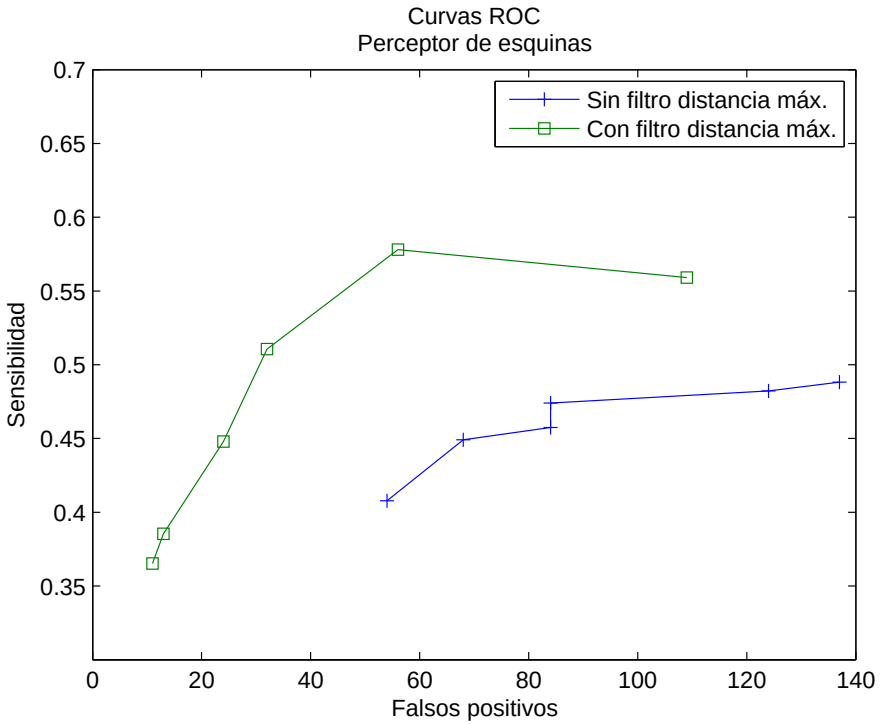


Figura 5.9: Curva ROC del perceptor de esquinas. Cada curva posee seis puntos de operación que representan los valores de píxeles de error mínimo para el algoritmo RANSAC. Éstos valores son: 1.0, 3.0, 5.0, 7.0, 9.0 y 11.0 pixeles, y están representados en cada curva por una marca.

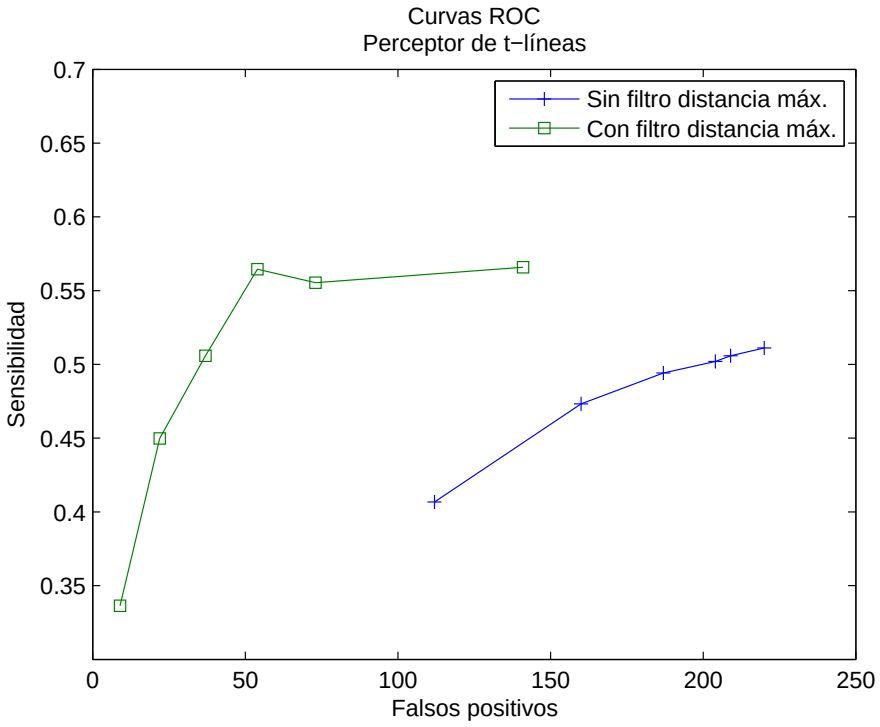


Figura 5.10: Curva ROC del perceptor de t-líneas. Cada curva posee seis puntos de operación que representan los valores de píxeles de error mínimo para el algoritmo RANSAC. Éstos valores son: 1.0, 3.0, 5.0, 7.0, 9.0 y 11.0 pixeles, y están representados en cada curva por una marca.

Para la percepción de esquinas y t-líneas, la aplicación del filtro de distancia disminuye notablemente la cantidad de falsos positivos. Su sensibilidad aumenta a medida que se eliminen los falsos positivos, ya que éstos dejan de ocupar las vacantes de las detecciones finales al igual que en los casos anteriores de arcos y líneas.

Al observar las curvas que representan el desempeño de ambos perceptores con el filtro de distancia, podemos deducir metodológicamente que el mejor valor para el mínimo valor de error de píxeles aceptado en el algoritmo RANSAC es igual a 7.0 pixeles para los ambos.

Indicador	Verdaderos Positivos	Falsos Positivos
Detecciones(%)	51.1%	3.78%

Cuadro 5.6: Tabla de detecciones del perceptor de esquinas utilizando el punto de operación $min_{error} = 7{,}0$. pixeles

Indicador	Verdaderos Positivos	Falsos Positivos
Detecciones(%)	56.5%	4.04%

Cuadro 5.7: Tabla de detecciones del perceptor de t-líneas utilizando el punto de operación $min_{error} = 7{,}0$. pixeles

Las tablas 5.6 y 5.7 muestran los porcentajes de verdaderos positivos y falsos positivos para un min_{error} = 7.0 pixeles.

5.3. Caracterización de la pose

El objetivo de caracterizar la pose es generar una función de corrección de la media y una función generadora de varianzas de la percepción de cada objeto. A continuación se muestran los resultados.

5.3.1. Perceptor de arcos

En la sección 4.3 se explica cómo se toman los datos y cómo se ajusta cada una de las siguientes funciones de corrección de la media y las funciones generadoras de matrices de covarianza para cada perceptor. Cada función mostrada en adelante corresponde a una función polinomial de grado dos, tres ó cuatro, tal que minimiza el menor *RMS* mediante un ajuste por mínimos cuadrados.

Funciones correctoras de la posición del arco detectado

Estas funciones de corrección se suman a cada componente de la posición calculada por el perceptor, quedando de ésta forma una posición final más cercana a la real, y lo más importante, cercana al valor esperado.

$$f_g^x(\tilde{z}^x,\tilde{z}^y) = -2{,}546\cdot 10^{-15} - 6{,}45\,x + 1{,}102\cdot 10^{-16}\,y + 7{,}654\cdot 10^{-16}\,x^2 + 1{,}918\cdot 10^{-16}\,xy + 1{,}416\cdot 10^{-15}\,y^2 \quad (5.2)$$

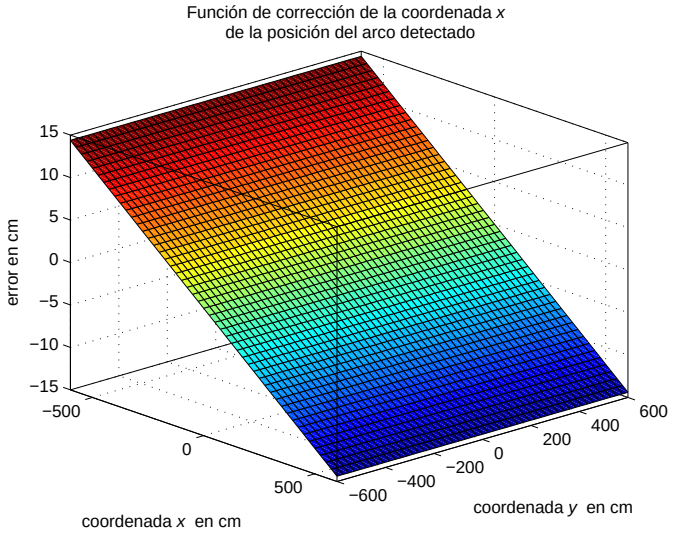


Figura 5.11: Función polinomial de corrección del error de la componente x de la posición del arco.

$$f_g^y(\tilde{z}^x,\tilde{z}^y) = -0{,}864 + 1{,}019\cdot 10^{-18}\,x - 0{,}05083\,y \quad (5.3)$$

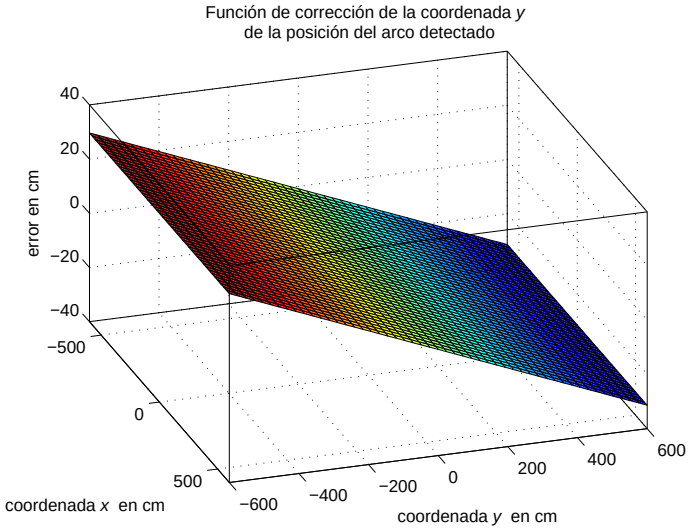


Figura 5.12: Función polinomial de corrección del error de la componente y de la posición del arco.

Funciones generadoras de la matriz de covarianza de la percepción del arco

Las figuras 5.13 hasta 5.18 corresponden a cada una de las funciones generadoras de cada componente de la matriz de covarianzas asociada a la detección de un arco. Recordemos que ésta matriz es cuadrada y tiene los elementos simétricos respecto a la diagonal principal. Es por ésto que se requieren conocer sólo seis de sus nueve elementos, los demás tienen su homónimo. Éstas covarianzas, como es de esperarse, crecen a medida que aumentan las distancias. Éste conocimiento de incerteza permite a los estimadores de estado y a la auto-localización funcionar de una manera mucho más realista en su respectiva etapa correctiva.

$$\begin{aligned} f_{cov}^{xx}(z^x, z^y) \quad = \quad & 460{,}6 - 8{,}114 \cdot 10^{-16} x - 1{,}388 \cdot 10^{-16} y + 0{,}008007\, x^2 + \\ & 9{,}474 \cdot 10^{-20} x y + 0{,}001999\, y^2 + 1{,}46 \cdot 10^{-21} x^3 - \\ & 2{,}369 \cdot 10^{-21} x^2 y + 1{,}12 \cdot 10^{-21} x y^2 + \\ & 2{,}728 \cdot 10^{-21} y^3 \end{aligned} \tag{5.4}$$

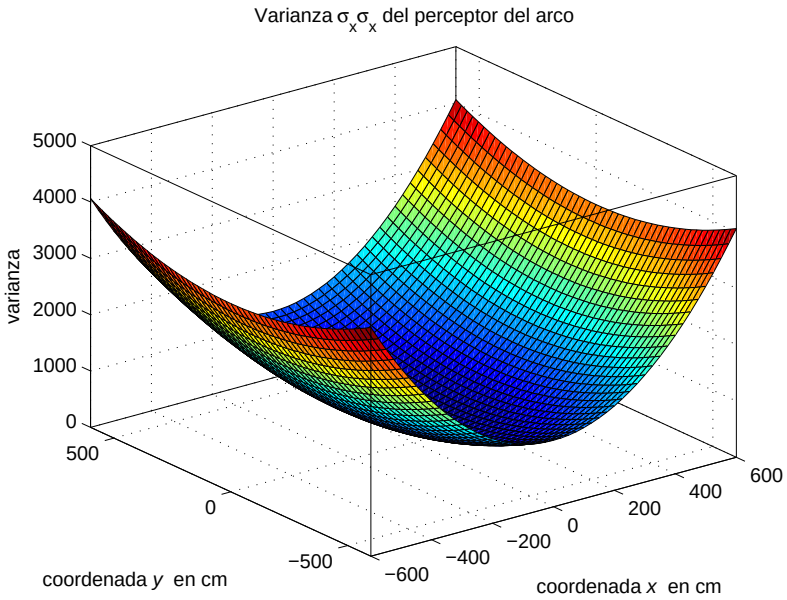


Figura 5.13: Función polinomial generadora de la componente $\sigma_x\sigma_x$ de la matriz de covarianzas de una detección de arco en función de la posición del arco.

$$
\begin{aligned}
f_{cov}^{yy}(z^x, z^y) \quad = \quad & 248{,}1 - 0{,}1154\,x - 0{,}00586\,y + 0{,}001842\,x^2 + \\
& 5{,}362e-005\,x\,y + 0{,}004264\,y^2 + 3{,}943\cdot 10^{-7}\,x^3 + \\
& 2{,}436\cdot 10^{-7}\,x^2\,y + 4{,}152\cdot 10^{-7}\,x\,y^2 - \\
& 4{,}372\cdot 10^{-9}\,y^3
\end{aligned} \tag{5.5}
$$

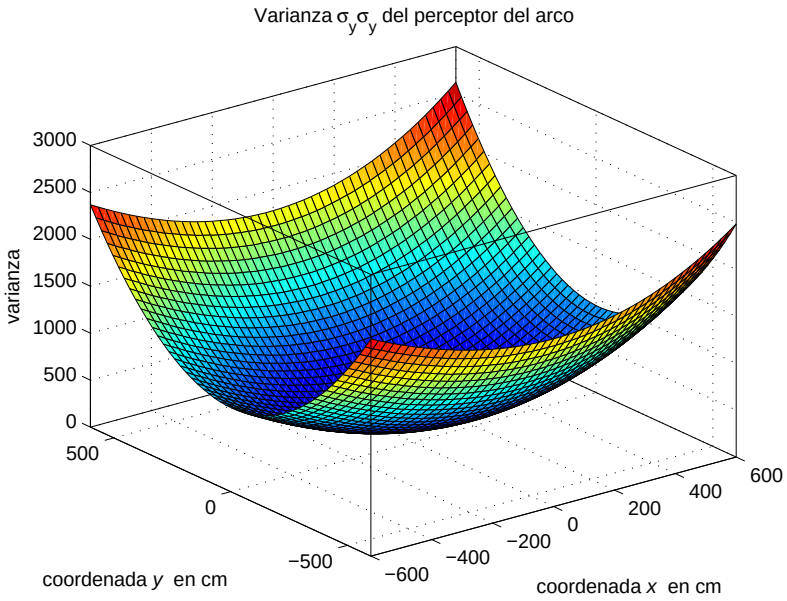


Figura 5.14: Función polinomial generadora de la componente $\sigma_y\sigma_y$ de la matriz de covarianzas de una detección de arco en función de la posición del arco.

$$
\begin{aligned}
f_{cov}^{\theta\theta}(z^x, z^y) \quad = \quad & 0{,}01384 - 5{,}387\cdot 10^{-18}\,x + 8{,}237\cdot 10^{-19}\,y + 0{,}002792\,x^2 + \\
& 4{,}6\cdot 10^{-19}\,xy + 0{,}02482\,y^2 + 3{,}271\cdot 10^{-18}\,x^3 - \\
& 1{,}291\cdot 10^{-18}\,x^2 y + 3{,}418\cdot 10^{-18}\,xy^2 - \\
& 1{,}736\cdot 10^{-18}\,y^3
\end{aligned} \tag{5.6}
$$

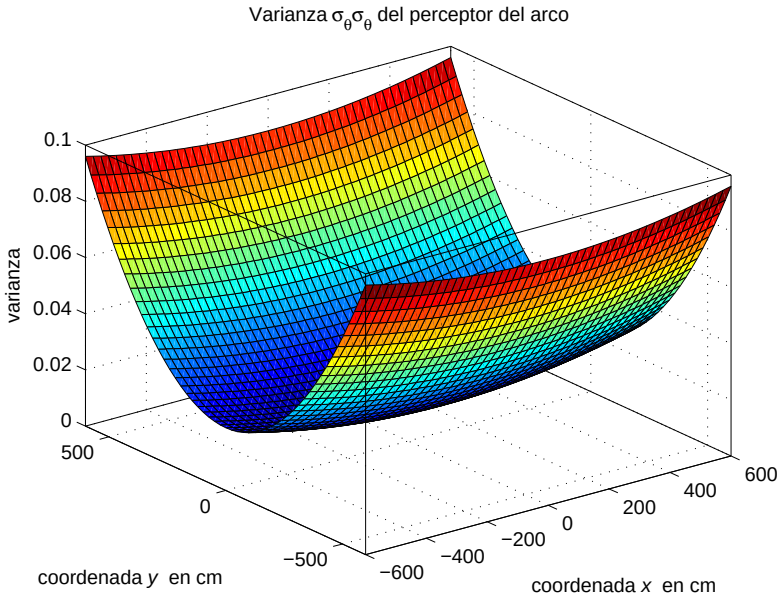


Figura 5.15: Función polinomial generadora de la componente $\sigma_{tetha}\sigma_{tetha}$ de la matriz de covarianzas de una detección de arco en función de la posición del arco.

$$
\begin{aligned}
f_{cov}^{xy}(z^x, z^y) \quad = \quad & 233 - 4{,}793 \cdot 10^{-14}\, x - 2{,}589 \cdot 10^{-14}\, y + 82{,}68\, x^2 + \\
& 5{,}056 \cdot 10^{-14}\, xy + 81{,}46\, y^2 + 4{,}581 \cdot 10^{-30}\, x^3 + \\
& 1{,}725 \cdot 10^{-15}\, x^2 y + 1{,}046 \cdot 10^{-29}\, xy^2 + \\
& 2{,}213 \cdot 10^{-14}\, y^3
\end{aligned} \tag{5.7}
$$

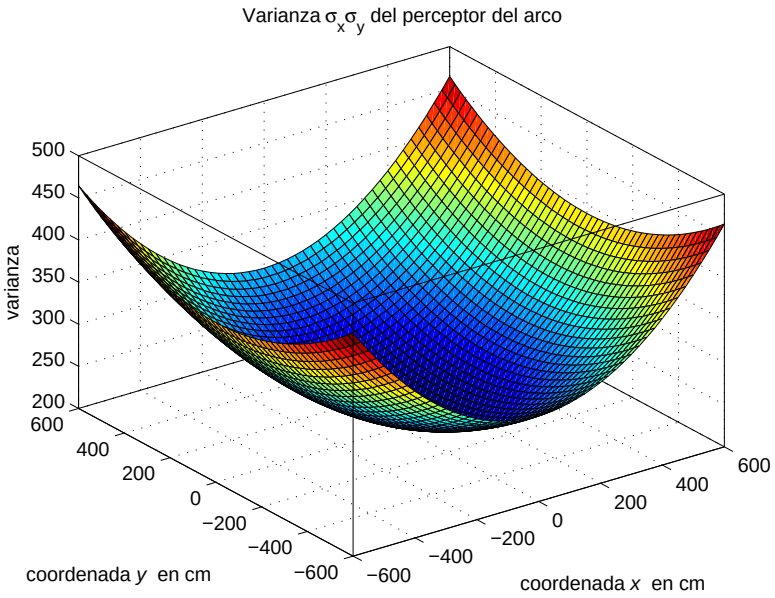


Figura 5.16: Función polinomial generadora de la componente $\sigma_x\sigma_y$ de la matriz de covarianzas de una detección de arco en función de la posición del arco.

$$
\begin{aligned}
f_{cov}^{y\theta}(z^x, z^y) \quad = \quad & 1{,}657 - 9{,}752\cdot 10^{-16}\,x + 2{,}911\cdot 10^{-16}\,y + -0{,}4063\,x^2 + 2{,}351\cdot 10^{-15}\,xy - \\
& 3{,}093\,y^2 + 4{,}089\cdot 10^{-16}\,x^3 - 5{,}972\cdot 10^{-17}\,x^2 y + 5{,}86\cdot 10^{-16}\,xy^2 - \\
& 1{,}258\cdot 10^{-16}\,y^3 + 0{,}03649\,x^4 - 7{,}431\cdot 10^{-16}\,x^3 y + \\
& 0{,}2865\,x^2 y^2 - 9{,}773\cdot 10^{-16}\,xy^3 + 0{,}8026\,y^4
\end{aligned} \tag{5.8}
$$

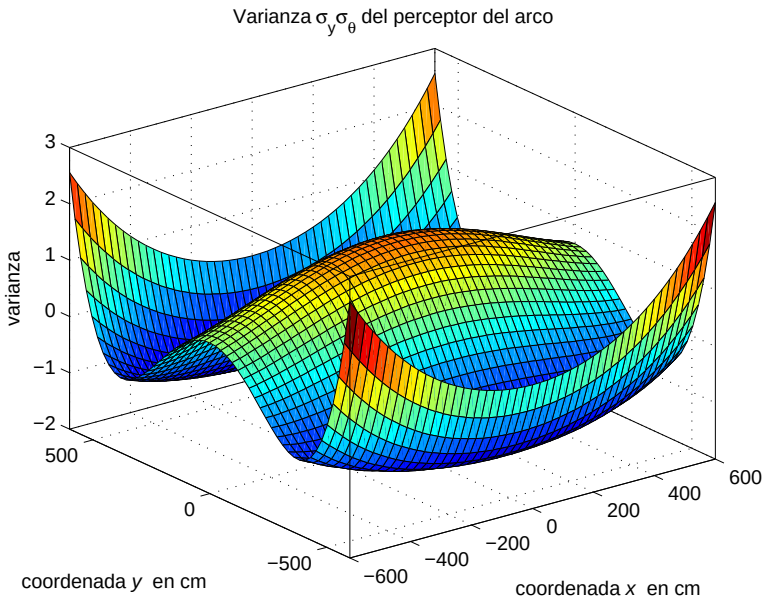


Figura 5.17: Función polinomial generadora de la componente $\sigma_y\sigma_{theta}$ de la matriz de covarianzas de una detección de arco en función de la posición del arco.

$$
\begin{aligned}
f_{cov}^{x\theta}(z^x, z^y) \quad = \quad & -0{,}2673 + 3{,}688 \cdot 10^{-16} x - 4{,}113 \cdot 10^{-17} y - 1{,}027 x^2 - \\
& 1{,}812 \cdot 10^{-16} xy - 0{,}7029 y^2 - 7{,}535 \cdot 10^{-17} x^3 + \\
& 2{,}995 \cdot 10^{-16} x^2 y + 5{,}873 \cdot 10^{-18} xy^2 - \\
& 4{,}417 \cdot 10^{-17} y^3
\end{aligned} \tag{5.9}
$$

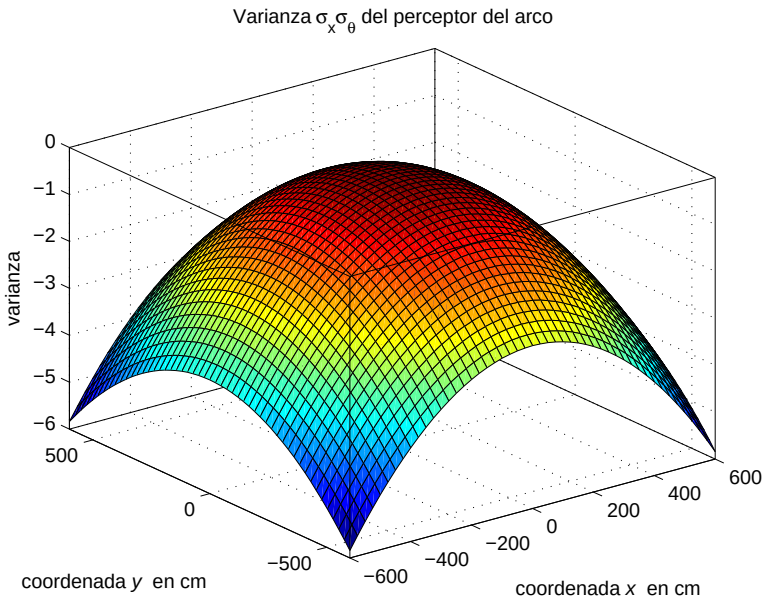


Figura 5.18: Función polinomial generadora de la componente $\sigma_x \sigma_{theta}$ de la matriz de covarianzas de una detección de arco en función de la posición del arco.

5.3.2. Perceptores de objetos líneas

De la misma manera que para el perceptor de arcos, para el perceptor de líneas se ha generado una función correctora de la medida de la pose.

Funciones correctoras de la posición del objeto línea detectado

$$f_g^x(\tilde{z}^x,\tilde{z}^y) = -3{,}379 - 0{,}07968\,x + 1{,}379\cdot 10^{-18}\,y + 2{,}831\cdot 10^{-5}\,x^2 - 6{,}514\cdot 10^{-20}\,xy + 0{,}0001099\,y^2 \quad (5.10)$$

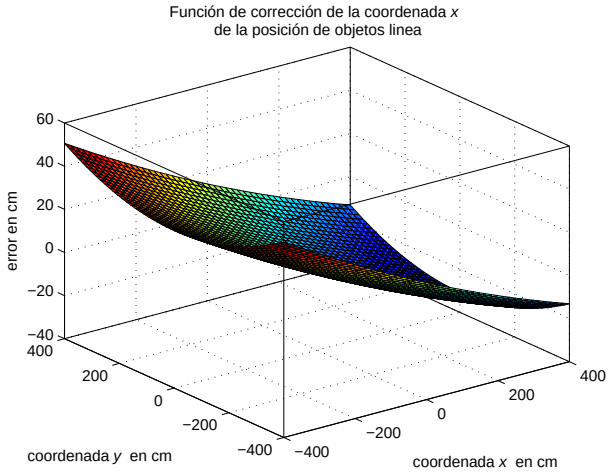


Figura 5.19: Función polinomial de corrección del error de la componente x de la posición del objeto línea.

$$f_g^y(\tilde{z}^x,\tilde{z}^y) = -0{,}3088 - 1{,}52\cdot 10^{-18}\,x - 0{,}07684\,y - 1{,}36\cdot 10^{-5}\,x^2 + 2{,}817\cdot 10^{-21}\,xy + 1{,}136\cdot 10^{-5}\,y^2 \quad (5.11)$$

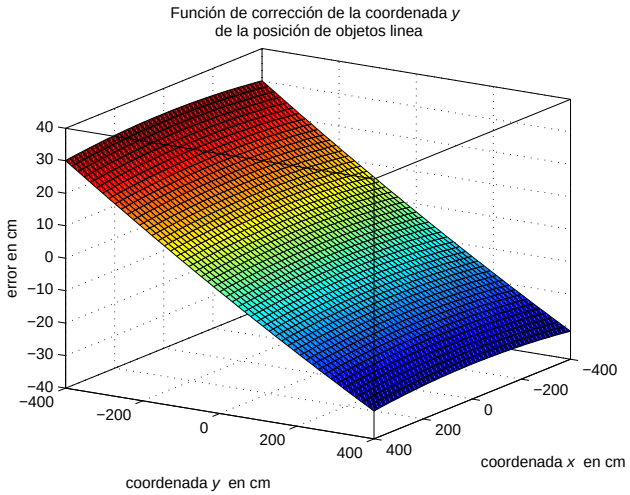


Figura 5.20: Función polinomial de corrección del error de la componente *y* de la posición del objeto línea.

Funciones generadoras de la matriz de covarianza de la percepción de objetos línea

$$f_{cov}^{xx}(z^x, z^y) = 12{,}61 + 1{,}431 \cdot 10^{-16} x + 5{,}663 \cdot 10^{-17} y + 0{,}005792 x^2 - 3{,}352 \cdot 10^{-20} xy + 0{,}0008914 y^2 \quad (5.12)$$

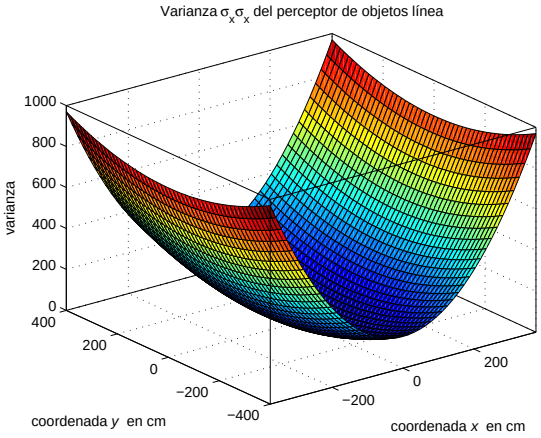


Figura 5.21: Función polinomial generadora de la componente $\sigma_x\sigma_x$ de la matriz de covarianzas de una detección de un objeto línea en función de su posición.

$$f_{cov}^{yy}(z^x, z^y) = 8{,}583 + 1{,}246 \cdot 10^{-16} x - 5{,}471 \cdot 10^{-17} y + 0{,}0006525 x^2 + 1{,}544 \cdot 10^{-19} xy + 0{,}003878 y^2 \quad (5.13)$$

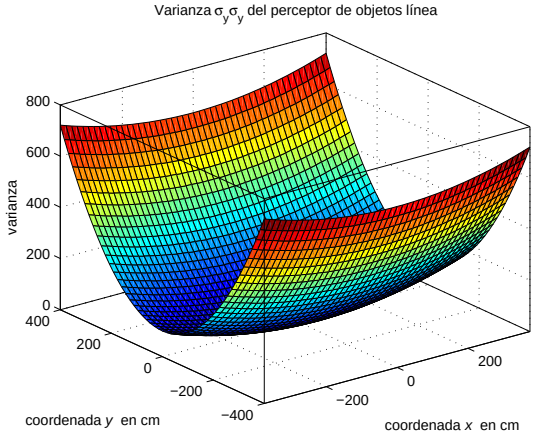


Figura 5.22: Función polinomial generadora de la componente $\sigma_y\sigma_y$ de la matriz de covarianzas de una detección de un objeto línea en función de su posición.

$$f_{cov}^{xy}(z^x, z^y) = 5{,}271 + 7{,}236 \cdot 10^{-18} x + 2{,}831 \cdot 10^{-18} y - 0{,}0004181\, x^2 - 2{,}972 \cdot 10^{-20} xy - 0{,}0004946\, y^2 \quad (5.14)$$

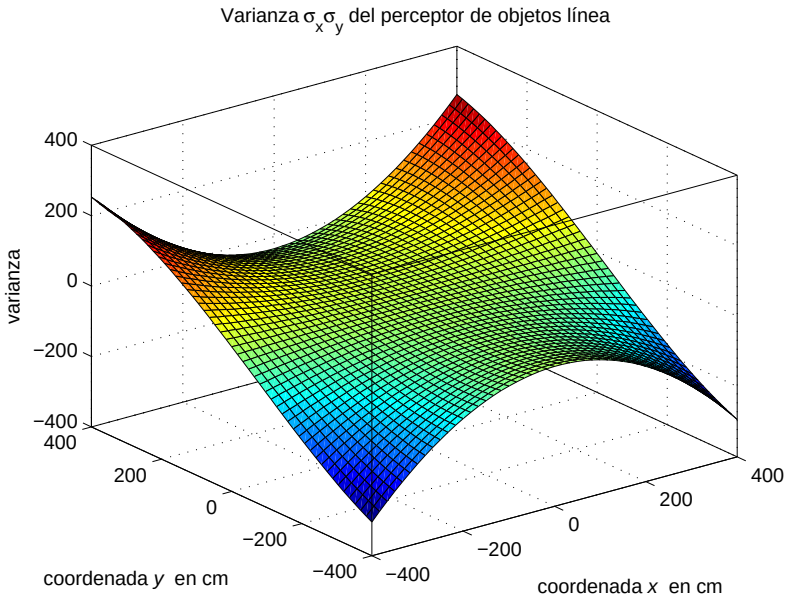


Figura 5.23: Función polinomial generadora de la componente $\sigma_x\sigma_y$ de la matriz de covarianzas de una detección de un objeto línea en función de su posición.

5.4. Identificación de objetos ambiguos y su uso en auto-localización

El objetivo de la detección de líneas y su identificación es mantener correctamente estimada la auto-localización durante los lapsos de tiempo que no se detectan los arcos. Para evaluar esto se capturan los datos de un simulador donde un robot juega durante 200 segundos. Estos datos corresponden a: i) el error de la localización, ii)la detección de cada una de las líneas y iii) si se logran identificar o no.

Como se explicó en la sección 4.4, el sistema de identificación de líneas, esquinas y t-esquinas basa su funcionamiento en la comparación del objeto candidato con cada objeto del mapa de la cancha. Éste procedimiento es bastante dependiente de la auto-localización y su matriz de covarianzas. Si la auto-localización no ha convergido, la covarianza del estimador de la auto-localización es alta, lo que provoca que todos los posibles objetos de la cancha tengan una alta probabilidad de ser correctos al compararlos con el objeto detectado (la covarianza de la localización aparece en el denominador de la ecuación 4.4 que calcula la innovación, y que al tender a 0, hace tender a 1 la función exponencial que estima la probabilidad señalada mostrada en la ecuación 4.4). Bajo ésta situación, todos los posibles objetos de la cancha adquieren una alta probabilidad, de esta manera se imposibilita una identificación segura del objeto detectado. Ésto sucede durante los primeros 45 segundos de la simulación y se puede observar en la figura 5.24 que las líneas son detectadas (marcadas con una *x*) pero no identificadas.

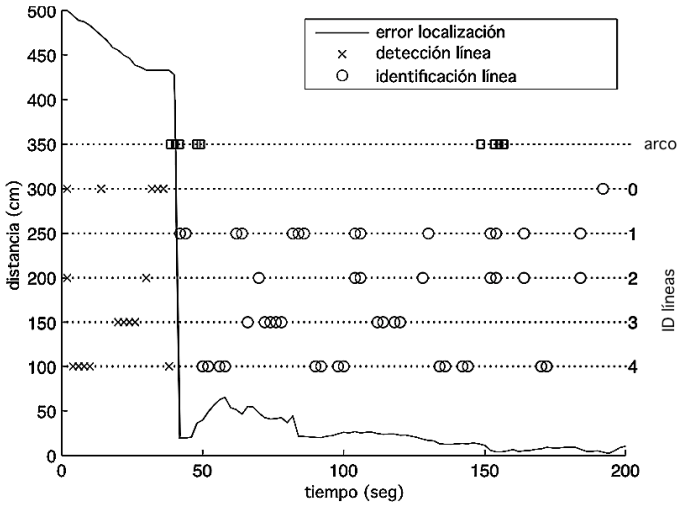


Figura 5.24: Ejemplo del proceso de identificación de líneas.

Una vez visto un arco, en torno al segundo 40, la auto-localización converge y el sistema comienza a identificar correctamente las líneas detectadas. En la figura 5.24 se puede apreciar esta transición, cuando las líneas comienzan a ser identificadas son marcadas con un *o*

Otra cosa destacable de esta simulación, es lo que sucede durante los segundos 50 y

150. Durante este lapso de tiempo no se detecta ningún arco pero sí se detectan e identifican líneas que se utilizan para corregir la auto-localización logrando mantener su error bajo en el tiempo.

Capítulo 6

Conclusiones

Este trabajo presenta un sistema de detección de objetos para un robot móvil humanoide que juega fútbol robótico. Se han desarrollado los perceptores correspondientes para detectar los objetos de la cancha basándose en la información visual de la cámara del robot. Además, se han desarrollado sistemas complementarios que ayudan a la detección de objetos, y su respectivo cálculo de pose, haciendo que el proceso de detección sea más robusto. También, cada detección es acompañada por una medida de incerteza, de esta forma es modelada como una variable aleatoria lo que permite su uso directo en sistemas de estimación de estados como la auto-localización. Por otra parte, debido a que los objetos de interés son ambiguos, fue necesario incorporar un sistema de identificación basado en la información de la auto-localización.

6.0.1. Estado de la cámara

Se ha desarrollado el estado completo de la cámara respecto al sistema de referencia local del robot, éste entrega información espacial muy importante tanto para la detección de objetos como para el cálculo de la pose de cada uno de ellos. Del estado de la cámara fue posible desprender el horizonte visual y proyectarlo sobre la imagen, este permite: utilizarlo como referencia para calibrar de manera bastante precisa el estado de la cámara, diseñar reglas físicas de detección para los perceptores y también para acelerar la búsqueda de puntos de alto gradiente ya que acota el área de búsqueda en la imagen. Además, es utilizado por otros perceptores como la pelota para acotar el área de búsqueda y así también eliminar falsos positivos. Con todo esto podemos decir que el horizonte visual es una información muy potente la cual siempre se debe tener presente al momento de desarrollar sistemas de percepción en robots móviles humanoides.

También fue necesario desarrollar un sistema de calibración del estado de la cámara lo que permitió incrementar bastante la exactitud de los cálculos de la pose de los objetos detectados. Esto da cuenta de la importancia de las calibraciones en sistemas robóticos móviles y el grado de sensibilidad de los procesos en un robot real. Es posible calcular la pose de los objetos detectados basándose en la información dada por el estado de la cámara, cuya calibración permite un alto grado de exactitud. Con los resultados obtenidos es posible

decir que el objetivo específico referente a calcular la pose de cada objeto detectado con exactitud fue logrado con éxito.

6.0.2. Percepción

Se han desarrollado cuatro tipos de perceptores: arcos, líneas, esquinas y t-líneas. El perceptor de arcos ha logrado un excelente desempeño teniendo una alta tasa de verdaderos positivos versus una baja tasa de falsos positivos. Éste buen desempeño queda evidenciado cuantitativamente en los resultados expuestos. Desde un punto de vista cualitativo, también es posible decir que posee un buen desempeño, éste es capaz de detectar los arcos observándolo desde cualquier posición, distancia y ángulo. Particularmente un arco cercano, a menos de un metro, es difícil detectarlo y calcularle su pose ya que sólo se proyecta una reducida parte de él, sin embargo el perceptor de arcos es capaz de adaptarse a ésta situación y pudiendo detectarlo sin problema. Ésta situación de este tipo se dio en un gol hecho durante las competencias de la RoboCup del año 2010 en Singapur. El perceptor de líneas no posee una tasa de detección tan alta como el perceptor de arcos debido a varios motivos: las líneas lejanas muchas veces no alcanzan a proyectarse sobre los suficientes píxeles en la imagen ya que son un objeto pequeño y sin volumen si lo comparamos con el arco, muchos objetos del entorno poseen forma de línea recta lo que obliga a utilizar parámetros bastante restrictivos con el fin de disminuir los falsos positivos lo que también hace disminuir la tasa de verdaderos positivos al mismo tiempo. Si bien la tasa de falsos positivos es alta en esta etapa de percepción, luego, en la etapa de identificación esta tasa disminuye considerablemente ya que los falsos positivos no pueden ser identificados. La percepción de objetos esquinas y t-líneas poseen una tasa de detección parecida a las líneas simples, sin embargo, la tasa de falsos positivos disminuye considerablemente ya que las condiciones adicionales de detección, como el requerimiento de perpendicularidad que es una condición bastante restrictiva. Con lo resultados obtenidos podemos decir que el objetivo específico relacionado con la detección de objetos fue logrado exitosamente.

6.0.3. Caracterización de la pose

Se han generado funciones correctoras de la posición del arco, líneas, esquinas y t-líneas y funciones generadoras de la matriz de covarianza en función de la posición de los objetos. Como es de esperarse, a medida que la distancia al objeto detectado aumenta, las funciones correctoras y generadoras de la matriz de covarianza crecen debido a que los objetos cada vez se proyectan sobre un área más pequeña sobre la imagen haciéndola más sensible al ruido del sensor de la cámara. Esta caracterización permite considerar las mediciones como variables aleatorias con distribución normal y así ser entradas directas a los procesos de estimación de estados, específicamente a un filtro de kalman extendido. Antiguamente se utilizaban matrices de covarianza fijas lo que hacía que la estimación no fuese estable cuando la medición es ruidosa, como por ejemplo el caso del arco lejano. Esta nueva información permite que las mediciones ruidosas ponderen relativamente menos en la estimación haciendo el proceso más cercano a la realidad. De esta forma se mejora el

desempeño del sistema de estimación de estados y auto-localización el cual correspondía a uno de los objetivos de esta tesis.

6.0.4. Identificación de objetos ambiguos

El proceso de identificación de objetos ambiguos es dependiente de la auto-localización, es decir, funciona correctamente si ésta posee una matriz de covarianza relativamente baja. Una vez que comienza la identificación, la auto-localización tiende a ser más robusta ya que ahora no sólo los arcos son observaciones válidas para el proceso de auto-localización, sino que también lo son líneas, esquinas y t-líneas identificadas. Dados lo resultados mostrados en el capítulo anterior podemos decir que el objetivo de identificación de objetos fue logrado con éxito.

El sistema presentado en esta tesis ha sido usado en las RoboCup 2010 en Singapur y RoboCup 2011 en Estambul donde funcionaron muy bien en situaciones reales de juego. Este sistema también ha sido utilizado como base para el desarrollo de la tesis doctoral titulada ”Visión Computacional Robótica Basada en Contexto” [14], en donde se propone una metodología para hacer uso de diferentes fuentes de información contextual para mejorar el rendimiento de un sistema de detección visual tal como el desarrollado en esta tesis.

Bibliografía

[1] ABDEL-AZIZ, Y. I., AND KARARA, H. M. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range photogrammetry* (1971), vol. 1, p. 18.

[2] ALDEBARAN. Aldebaran robotics, www.aldebaran-robotics.com.

[3] ARAI, T. Araibo technical report. University of Tokyo and Chuo University, 2005.

[4] CZARNETZKI, S., KERNER, S., URBANN, O., HOFMANN, M., STUMM, S., AND SCHWARZ, I. Nao devils dortmund team report 2010, 2010.

[5] FISCHLER, M. A., AND BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24* (June 1981), 381–395.

[6] FU, S., AND GONZALEZ, C. Robótica: control, detección, visión e inteligencia, 1988.

[7] GUERRERO, P., RUIZ-DEL SOLAR, J., AND DAZ, G. Probabilistic decision making in robot soccer. In *Lecture Notes in Computer Science* (2007), vol. 5001, pp. 29–40.

[8] GUERRERO, P., RUIZ-DEL SOLAR, J., FREDES, J., AND PALMA-AMESTOY, R. Automatic on-line color calibration using class-relative color spaces. In *RoboCup 2007: Robot Soccer World Cup XI*, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds., vol. 5001 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 246–253.

[9] HESTER, T., QUINLAN, M., STONE, P., AND SRIDHARAN, M. Tt-ut austin villa 2009: Naos across texas, 2009.

[10] INOUE, L. J., INOUE, J., KOBAYASHI, H., ISHINO, A., AND SHINOHARA, A. Jolly pochie team description paper. Department of Informatics, Kyushu University, 2005.

[11] KRISHNA, K., AND MURTY, M. N. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B 29*, 3 (1999), 433–439.

[12] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60* (2004), 91–110. 10.1023/B:VISI.0000029664.99615.94.

[13] MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), L. M. L. Cam and J. Neyman, Eds., vol. 1, University of California Press, pp. 281–297.

[14] PALMA, R. *Visión Computacional Robótica Basada en Contexto*. PhD thesis, Universidad de Chile, 2011.

[15] QUINLAN, M. Nubots team report, 2005.

[16] ROBOCUP. Robocup, www.robocup.org.

[17] RÖFER, T. German team technical report. Center for Computing Technology, Bremen University, 2005.

[18] RÖFER, T., LAUE, T., MÜLLER, J., BÖSCHE, O., BURCHARDT, A., DAMROSE, E., GILLMANN, K., GRAF, C., DE HAAS, T. J., HÄRTL, A., RIESKAMP, A., SCHRECK, A., SIEVERDINGBECK, I., AND WORCH, J.-H. B-human team report and code release 2009, 2009.

[19] RÖFER, T., LAUE, T., AND WEBER, M. German team report 2005, 2005.

[20] RUIZ-DEL SOLAR, J., VALLEJOS, P., LASTRA, R., LONCOMILLA, P., ZAGAL, J., MORÁN, C., AND SARMIENTO, I. Uchile1 technical report, 2005.

[21] RUIZ-DEL SOLAR, J., VALLEJOS, P., PARRA, I., TESTART, J., ASENJO, R., HEVIA, P., VÉLEZ, C., AND LARRAÍN, F. Uchile roadrunners 2008 team description paper, 2008.

[22] SAKAMOTO, H. Hajime research institute, www.hajimerobot.co.jp.

[23] SERRA, J. Image analysis and mathematical morphology. *Computer Graphics and Image Processing 20*, 1 (1982), 96 – 97.

[24] VERSCHAE, R. *Object detection using nested cascades of boosted classifiers: a learning framework and its extension to the multi-class case*. PhD thesis, Universidad de Chile, 2010.

Capítulo 7

Anexo 1

En [6] se describe la cinemática directa como un método generalizado y sistemático para describir y representar la localización de los elementos de un brazo articulado con respecto a un sistema de referencia fijo. El problema cinemático directo se reduce a encontrar una matriz de transformación que relaciona el sistma de coordenadas ligado al cuerpo al sistema de coordenadas de referencia. Se utilizan matrices de rotación 3x3 para describir las operaciones rotacionales del sistema ligado al cuerpo con respecto al sistema de referencia. Se utilizan entonces coordenadas homogéneas para representar vectores de posición en un espacio tridimensional, y las matrices de rotación se amplían a matrices de transformación homogénea 4x4 para incluir las operaciones traslacionales del sistema ligado al cuerpo. Para un sistema móvil que posee varias articulaciones es necesario postmultiplicar todas las matrices de transformación (rotaciones o traslaciones) de acuerdo a todas las rotaciones y traslaciones que se encuentren desde el sistema de referencia hasta el sistema móvil. Para nuestro caso particular requerimos definir cuatro matrices de transformación: la matriz de transformación homogénea del pie derecho al centro del torso del robot denominada por P_{der}, la del de pie izquierdo al centro del torso del robot denominada por P_{izq}, la del centro del torso del robot a la cámara superior denominada por C_{sup} y la del centro del torso del robot a la cámara inferior denominada por C_{inf}. Las matrices de traslación quedan en función del largo de los segmentos del robot los cuales son siempre fijos, la figura 7.2 muestra las secciones del robot, sus nombres asignados por convención y sus longitudes en milímetros. Las matrices de rotación quedan en función del valor dado por los sensores de posición angular llamados *encoders*, éstos varían de acuerdo a los movimientos del robot y de cada articulación, la figura 7.1 muestra la ubicación y los nombres de cada articulación.

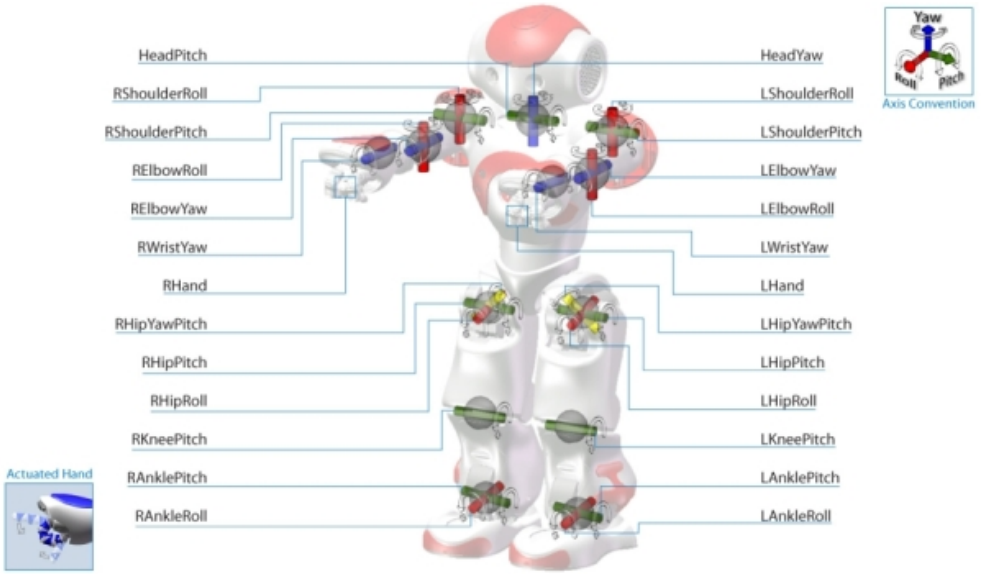


Figura 7.1: Articulaciones del robot Nao.

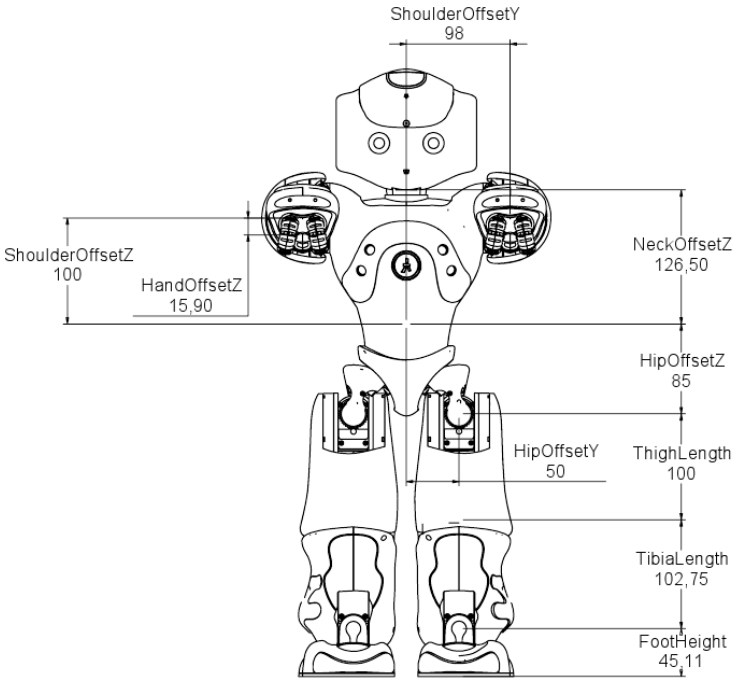


Figura 7.2: Medidas en milímetros de las medidas de las secciones móviles del robot.

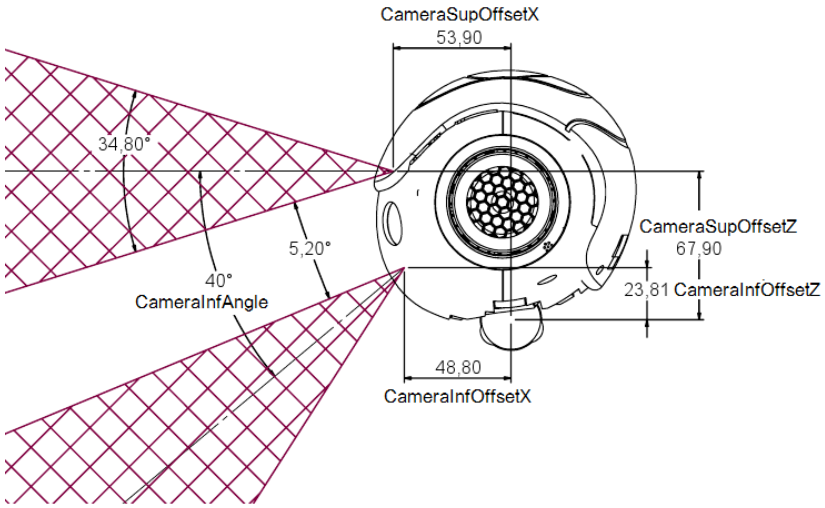


Figura 7.3: Disposición de las cámaras en la cabeza del robot.

La ecuación 7 muestra el armado de la matriz de transformación homogénea P_{der}. En ella las matrices de traslación están representadas por T con un subíndice que indica el eje de traslación, de manera similar las matrices de rotación están representadas por R con un subíndice que indica el eje de rotación de la articulación. Del mismo modo se encuetran armadas las matrices de transformación P_{izq}, C_{sup} y C_{inf} cuyas ecuaciones son 7, 7 y 7 respectivamente. En las ecuaciones 7, 7, 7 y 7 se han reemplazado T y R por su correspondiente matriz homogénea de traslación o rotación quedando en función del valor del largo de cada sección y del valor del estado angular de cada articulación involucrada. No se muestran las matrices de transformación homogénea finales debido al tamaño de éstas, además en la práctica la implementación posee funciones para cada tipo de transformación las que se evalúan numéricamente antes de multiplicarlas. Cabe destacar que las ecuaciones finales de cada transformación P y C quedan en función de los valores del estado angular de las articulaciones ya que las secciones del robot son de tamaño fijo y se conocen de antemano. También las matrices de rotación $R_x(-RHipAngle)$, $R_x(RHipAngle)$, $R_x(-LHipAngle)$ $R_x(LHipAngle)$ y $R_y(CameraInfAngle)$ son matrices de rotación fijas de 45°ya que representan rotaciones que posee la estructura del robot específicamente en su cadera. Del mismo modo para la rotación $R_y(CameraInfAngle)$ de C_{inf} ya que la cámara inferior posee una rotación respecto al sistema de referencia del robot, ésta puede observarse en la figura 7.3.

$$\begin{aligned} P_{der} \quad = \quad & T_z(FootHeight)R_x(RAnkleRoll)R_y(RAnklePitch)R_z(TibiaLenght)R_y(RKneePitch) \\ & T_z(ThighLenght)R_y(RHipPitch)R_x(RHipRoll)R_x(-RHipAngle)R_z(HipYawPitch) \\ & R_x(RHipAngle)T_y(HipOffsetY)T_z(HipOffsetZ); \end{aligned} \tag{7.1}$$

$$
\begin{aligned}
P_{der} = {} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & FootHeight \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(RAnkleRoll) & -\sin(RAnkleRoll) & 0 \\ 0 & \sin(RAnkleRoll) & \cos(RAnkleRoll) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(RAnklePitch) & 0 & \sin(RAnklePitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(RAnklePitch) & 0 & \cos(RAnklePitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & TibiaLenght \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(RKneePitch) & 0 & \sin(RKneePitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(RKneePitch) & 0 & \cos(RKneePitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & ThighLenght \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(RHipPitch) & 0 & \sin(RHipPitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(RHipPitch) & 0 & \cos(RHipPitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(RHipRoll) & -\sin(RHipRoll) & 0 \\ 0 & \sin(RHipRoll) & \cos(RHipRoll) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(RHipAngle) & \sin(RHipAngle) & 0 \\ 0 & -\sin(RHipAngle) & \cos(RHipAngle) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(HipYawPitch) & -\sin(HipYawPitch) & 0 & 0 \\ \sin(HipYawPitch) & \cos(HipYawPitch) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(RHipAngle) & -\sin(RHipAngle) & 0 \\ 0 & \sin(RHipAngle) & \cos(RHipAngle) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & HipOffsetY \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & HipOffsetZ \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{7.2}
$$

$$
\begin{aligned}
P_{izq} = {} & T_z(FootHeight) R_x(LAnkleRoll) R_y(LAnklePitch) R_z(TibiaLenght) R_y(LKneePitch) \\
& T_z(ThighLenght) R_y(LHipPitch) R_x(LHipRoll) R_x(LHipAngle) R_z(HipYawPitch) \\
& R_x(-LHipAngle) T_y(-HipOffsetY) T_z(HipOffsetZ);
\end{aligned} \tag{7.3}
$$

$$
\begin{aligned}
P_{izq} = {} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & FootHeight \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(LAnkleRoll) & -\sin(LAnkleRoll) & 0 \\ 0 & \sin(LAnkleRoll) & \cos(LAnkleRoll) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(LAnklePitch) & 0 & \sin(LAnklePitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(LAnklePitch) & 0 & \cos(LAnklePitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & TibiaLenght \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(LKneePitch) & 0 & \sin(LKneePitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(LKneePitch) & 0 & \cos(LKneePitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & ThighLenght \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(LHipPitch) & 0 & \sin(LHipPitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(LHipPitch) & 0 & \cos(LHipPitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(LHipRoll) & -\sin(LHipRoll) & 0 \\ 0 & \sin(LHipRoll) & \cos(LHipRoll) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(LHipAngle) & -\sin(LHipAngle) & 0 \\ 0 & \sin(LHipAngle) & \cos(LHipAngle) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(HipYawPitch) & -\sin(HipYawPitch) & 0 & 0 \\ \sin(HipYawPitch) & \cos(HipYawPitch) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(LHipAngle) & \sin(LHipAngle) & 0 \\ 0 & -\sin(LHipAngle) & \cos(LHipAngle) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -HipOffsetY \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & HipOffsetZ \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{7.4}
$$

$$
C_{sup} = T_z(NeckOffsetZ) R_z(HeadYaw) R_y(HeadPitch) T_z(CameraSupOffsetZ) T_x(CameraSupOffsetX) \tag{7.5}
$$

$$
\begin{aligned}
C_{sup} = & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & NeckOffsetZ \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(HeadYaw) & -\sin(HeadYaw) & 0 & 0 \\ \sin(HeadYaw) & \cos(HeadYaw) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(HeadPitch) & 0 & \sin(HeadPitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(HeadPitch) & 0 & \cos(HeadPitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & CameraSupOffsetZ \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & CameraSupOffsetX \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{7.6}
$$

$$
C_{inf} = T_z(NeckOffsetZ)R_z(HeadYaw)R_y(HeadPitch)T_z(CameraSupOffsetZ)T_x(CameraSupOffsetZ)R_y(CameraInfAngle) \tag{7.7}
$$

$$
\begin{aligned}
C_{inf} = & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & NeckOffsetZ \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(HeadYaw) & -\sin(HeadYaw) & 0 & 0 \\ \sin(HeadYaw) & \cos(HeadYaw) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} \cos(HeadPitch) & 0 & \sin(HeadPitch) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(HeadPitch) & 0 & \cos(HeadPitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & CameraInfOffsetZ \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & CameraInfOffsetX \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(CameraInfAngle) & 0 & \sin(CameraInfAngle) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(CameraInfAngle) & 0 & \cos(CameraInfAngle) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{7.8}
$$