DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS QUE PERMITAN REALIZAR LANZAMIENTOS DE TIROS PENALTIS CON ROBOTS NAO

JULIÁN DAVID DÍAZ MILLÁN

WILFER CASTAÑO TORO

INSTITUCIÓN UNIVERSITARIA LOS LIBERTADORES

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ DC

2014

DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS QUE PERMITAN REALIZAR LANZAMIENTOS DE TIROS PENALTIS CON ROBOTS NAO

JULIÁN DAVID DÍAZ MILLÁN

WILFER CASTAÑO TORO

TRABAJO DE INVESTIGACIÓN PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

ANDRES CAMILO JIMENEZ ALVAREZ

JHON PEATERSON ANZOLA.

LUIS ALEJANDRO CAYCEDO VILLALOBOS

INSTITUCIÓN UNIVERSITARIA LOS LIBERTADORES

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ DC

2014

Nota de aceptación

_____

_____

_____

_____

_____

_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Bogotá DC septiembre 3 del 2014

DEDICATORIA

Dedicamos este trabajo a Dios, a nuestros padres y hermanos que nos brindan su apoyo día a día para salir adelante y cumplir una más de nuestras metas que nos proponemos, a todas las personas que nos apoyaron de una u otra forma en este proceso.

AGRADECIMIENTOS

A Dios por darnos la oportunidad de tener una experiencia enriquecedora, a nuestros padres, hermanos, que nos brindaron su apoyo incondicional para cumplir una de nuestras más importantes metas en la vida y a todas aquellas personas que nos dieron su apoyo en este proceso de aprendizaje.

Agradecemos a la institución universitaria los libertadores en la faculta de ingeniería electrónica y a cada uno de los docentes que nos aportaron su conocimiento en las distintas áreas.

Agradecemos a los ingenieros Luis Alejandro Caicedo Villalobos, Andrés Camilo Jiménez Álvarez, Jhon Peaterson Anzola, quienes nos aportaron sus conocimientos y estuvieron presentes para el desarrollo de este proyecto.

Tabla de contenido

LISTA DE ILUSTRACIONES

Pág.

LISTA DE TABLAS

Pág.

INTRODUCCIÓN

El siguiente trabajo tiene como objetivo diseñar un algoritmo basado en objetivos que permita al robot humanoide nao H21 realizar cobros desde el punto penalti.

Dentro de los trabajos de investigación realizados y consultados acerca del robot humanoide NAO en el futbol robótico, no se han encontrado proyectos que realicen Lanzamientos de tiros penales, en los cuales se pueda evaluar la efectividad de cobro.

Este trabajo está enfocado al desarrollo y la participación del futbol robótico, teniendo como referente la RoboCup, que permita a la institución universitaria los libertadores y el área de ingeniería Electrónicaparticipar y demostrar los avances obtenidos con los robots humanoides NAO.

El trabajo se centra en generar movimientos y algoritmos, utilizando herramientas incorporadas en el humanoide NAO, que permitan buscar e identificar el balón y el arco para realizar los cobros de tiro penal teniendo en cuenta librerías de visión artificial, (OpenCV).

1. ESTADO DEL ARTE

Durante las últimas décadas la robótica ha evolucionado a grandes pasos, presentando sistemas totalmente autónomos, los cuales son capaces de desenvolverse por si mismos en cualquier entorno desconocido y totalmente cambiante. Actualmente empresas Como Honda motor y Aldebaran robotic se centran en desarrollar robots humanoides Como ASIMO y NAO respectivamente, que puedan desarrollar funciones que son propias de los humanos Como caminar, subir escaleras sin perder el equilibrio, determinar la distancia y manipular objetos, jugar futbol, entre otros.

El robot humanoide NAO creado por la compañía francesa Aldebaran robotic es actualmente utilizado en gran parte Del mundo para fines académicos y es empleado en el evento de la RoboCup.

1.1 ROBOT NAO AYUDA PARA EL AUTISMO

En la Universidad tecnológica de Huejotzingo, Puebla, Gerardo Velázquez encargado Del área de robótica y creador Del Proyecto que busca mediante el humanoide NAO que los niños que sufren de autismo puedan interactuar y a través del robot.

El robot NAO fue programado para realizar algunas rutinas de baile y movimientos que enseñen a los niños a reconocer los objetos, desplazarse y realizar una tarea o actividad y desarrollar la imaginación y motricidad.
“El robot se convierte en su amigo, no lo ven Como un doctor si no como un niño con el que pueden jugar. El también profesor de la carrera de robótica asegura que los pequeños con este síndrome de aislamiento y que se han sometido a esta terapia en Huejotzingo han tenido progresos de hasta 40%.” (Estrada, junio de 2012)

Ilustración 1 Robot nao ayuda a niños autistas

1.2 TRABAJOS REALIZADOS EMPLEANDO EL ROBOT NAO

1.2.1 Análisis cinemático con el robot nao

Al día de hoy se han venido desarrollando distintos proyectos de investigación en cuanto a robots humanoides se refiere, creando maquinas autónomas totalmente reprogramables capaces de realizar actividades que al ser humano se le dificultan de una u otra manera

Para realizar un análisis cinemático[1] del robot nao se utiliza la metodología de Denavit-Hartenberg[2] en el que permite que cada matriz solo dependa de cuatro parámetros en lugar de seis para definir la posición final o inicial del robot. Mediante la cinemática directa es posible conocer en qué posición se encuentra ubicado el robot, para ello es necesario conocer lo que son el espacio articular y los parámetros geométricos del robot. (Contreras Tapia Luis M., 2012)

De acuerdo al algoritmo de D-H, se escoge adecuadamente los sistemas de coordenadas para cada eslabón, siendo posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características de cada eslabón

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permitan relacionar el sistema de referencia del elemento i con el sistema del elemento i-1. Las transformaciones en cuestión son las siguientes:

Rotación alrededor del eje Zi-1 un ángulo qi.
Traslación a lo largo de Zi-1 una distancia di; vector di (0, 0, di).
Traslación a lo largo de Xi una distancia ai; vector ai (0, 0, ai).
Rotación alrededor del eje Xi, un ángulo ai.(Alberto)

A partir de los 4 parámetros de Denavit-Hartenberg mencionados anteriormente, conforman el siguiente algoritmo para la resolución del problema cinemático directo:

[1]La cinemática es la rama de la física que estudia las leyes del movimiento de los cuerpos sin considerar las causas que lo originan y se limita, esencialmente, al estudio de la trayectoria en función del tiempo.
[2]Denavit-Hartenberg propusieron en 1955 un método matricial que permite establecer de manera sistemática un sistema de coordenadas (Si) ligado a cada eslabón i de una cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.

DH1	Numerar los eslabones comenzando con 1 (primer eslabón móvil dela cadena) y acabando con n (último eslabón móvil). Se numerara como eslabón 0 a la base fija del robot.
DH2	Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad y acabando en n).
DH3	Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
DH4	Para i de 0 a n-1, situar el eje Zi, sobre el eje de la articulación i+1.
DH5	Situar el origen del sistema de la base (S0) en cualquier punto del eje Z0. Los ejes X0 e Y0 se situaran dé modo que formen un sistema dextrógiro con Z0.
DH6	Para i de 1 a n-1, situar el sistema (Si) (solidario al eslabón i) en la intersección del eje Zi con la línea normal común a Zi-1 y Zi. Si ambos ejes se cortasen se situaría (Si) en el punto de corte. Si fuesen paralelos (Si) se situaría en la articulación i+1.
DH7	Situar Xi en la línea normal común a Zi-1 y Zi.
DH8	Situar Yi de modo que forme un sistema dextrógiro con Xi y Zi.
DH9	Situar el sistema (Sn) en el extremo del robot de modo que Zn coincida con la dirección de Zn-1 y Xn sea normal a Zn-1 y Zn.
DH10	Obtener Øi como el ángulo que hay que girar en torno a Zi-1 para que Xi-1 y Xi queden paralelos.
DH11	Obtener di como la distancia, medida a lo largo de Zi-1, que habría que desplazar (Si-1) para que Xi y Xi-1 quedasen alineados.
DH12	Obtener ai como la distancia medida a lo largo de Xi (que ahora coincidiría con Xi-1) que habría que desplazar el nuevo (Si-1) para que su origen coincidiese con (Si).
DH13	Obtener ai como el ángulo que habría que girar entorno a Xi (que ahora coincidiría con Xi-1), para que el nuevo (Si-1) coincidiese totalmente con (Si).
DH14	Obtener las matrices de transformación i-1Ai.
DH15	Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot T = 0Ai, 1A2... n-1An.
DH16	La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido ala base en función de las n coordenadas articulares.

Tabla 1. Algoritmo para la resolución del problema cinemático directopor el método de Denavit-Hartenberg

1.2.2 Tele operación del robot NAO mediante dispositivos móviles Android

Con el desarrollo tecnológico que se ha venido presentando últimamente no solo en la robótica sino que también en dispositivos móviles teléfonos inteligente y tabletas, es muy común acceder a cualquier sitio desde nuestros dispositivos móviles, de igual manera se pretende controlar y manejar otros dispositivos desde nuestros teléfonos o tabletas.

Para realizar los métodos de tele operación se tiene en cuenta dos principios: el modelo de control que ejerce el usuario sobre el robot y la información que este recibe de vuelta. A partir de esto podemos definir los métodos a emplear.

1.2.2.1Control bilateral

Este tipo de control se emplea cuando en el sistema existe algún tipo de realimentación asía el operador, proveniente del esclavo[3] al momento de realizar una tarea. Este procedimiento se lleva a cabo para impedir que el robot esclavo se dañe o se ejerzan fuerzas excesivas. El usuario realiza un movimiento en el lado maestro[4] se obtiene la posición y esta es transmitida al esclavo, adaptándose a la del maestro.

1.2.2.2Control supervisado y coordinado

Estos sistemas son aquellos que realizan acciones de un nivel mucho más alto que el mencionado anteriormente ejerciendo al operador que realice una acción de supervisión. Existe un canal de comunicación entre el esclavo y maestro mediante el cual se realiza un proceso de enviar instrucciones y respuestas. El maestro es quien realiza la acción de control de una manera indirecta ya que existe retardo debido al canal donde se comunican ambos.

“Actualmente es el sistema de tele operación más extendido y sobre el cual se estánrealizando un mayor número de investigaciones. Permite la realización de pruebas conmodelos 3D.”(Gálvez Cobo, 2012)

El sistema operativo de Apple, iOS, cuenta con una aplicación, icontrolNao, creada por Klaus Engel[5] un sistema de tele operación que permite al usuario mediante predeterminadas funciones realizar distintos movimientos

[3] el esclavo es quien responde a la petición del maestro, si les corresponde, el proceso de pregunta/respuesta de un equipo maestro a uno esclavo se lo conoce como transacción

[4] el maestro es quien gobierna los ciclos de comunicación, toda iniciativa de comunicación es llevada a cabo por este equipo

[5] CEO Nacido en Duisburg, Alemania, 21 de abril 1956 , Presidente de la Junta Ejecutiva de Evonik Industries AG, Essen

directamente sobre el robot nao. Además esta aplicación es capaz de reconocer los Nao que se encuentran cerca y se conectan automáticamente.

1.3ROBOT NAO E INTELIGENCIA ARTIFICIAL

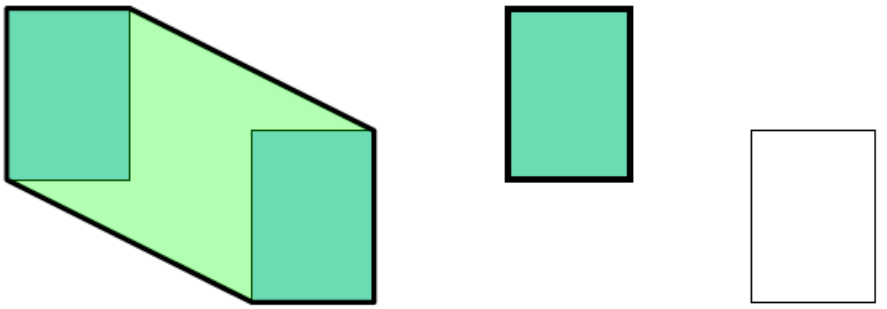
1.3.1 Optimización y modelado de un robot NAO usando técnicas de IA

Para que un robot humanoide tenga un funcionamiento útil y eficaz es necesario tener en cuenta las diversas e importantes características que poseen estos sistemas autónomos. Una de las características más importantes es la forma y velocidad con la que se desplaza el robot sin perder el equilibro. Esta velocidad dependerá de los parámetros establecidos en el módulo de movimiento y la superficie en la que se desplace. La optimización de este sistema no se centra solo en la velocidad de la caminata sino que también que este no sea inestable y tenga una trayectoria en línea recta para ello se emplean algoritmos que den solución a dicho problema.

1.3.1.1 Algoritmo ZMP (Zero-moment point)

Esta técnica de controlactualmente es una de las más extendidas y utilizadas a pesar de que fue introducida por primera vez hace más de 35 años, este algoritmo establece un criterio de estabilidad dinámica para el robot que genera patrones de locomoción.

Para poder dar una definición puntual de este tipo de algoritmo se tomara una definición de un artículo de la universidad de chile.
“ZMP (Zero Moment Point, punto de momento nulo) se puede definir como el punto en el suelo tal que el momento neto de las fuerzas externas no tiene componente sobre los ejes horizontales. Cuando existe dentro del polígono de soporte, el contacto entre el suelo y el pie es estable. Cuanto más cercano esté al centro de la superficie de soporte, más robustez se conseguirá. Cuando ZMP está fuera del polígono de soporte, el robot se inclina rotando sobre alguno de los bordes de dicho polígono.”(Gómez, Optimización y modelado de un robot bípedo NAO usando técnicas de IA, 2012).



(a) *Soporte doble (dos pies)* (b) *Soporte simple (un pie)*

Ilustración 2Polígono de soporte según fase de locomoción

El algoritmo de ZMP se basa en los siguientes parámetros:

1. Se definen una serie de posiciones de los pasos deseados.
2. Para encontrar las trayectorias de un paso se sigue un proceso de optimización. Las restricciones que debe cumplir la optimización deben ser:
   - Condición ZMP (equilibrio dinámico).
   - Cumplimiento de las posiciones deseadas para cada paso.
   - Mínimos y máximos movimientos articulares.
   - Criterios estilísticos: ya que el modelo a resolver es de dimensión muy elevada, es necesario definir más criterios. Éstos pueden ser del estilo trayectoria del pie cicloidal, altura de las caderas dentro de un rango, etc.
   - La función a optimizar puede incluir aspectos energéticos (minimizar la energía aportada por los actuadores en un ciclo), de robustez (ZMP lo más cercano posible al centro del polígono de soporte), y otros dependientes de cada caso concreto.

2 MARCO TEÓRICO

2.1 ROBOT NAO H21

El robot humanoide Nao H21 presentado por la compañía francesa Aldebaran Robotics, es la plataforma perfecta para los propósitos académicos en diversos temas, desde la robótica y la interacción humano-robot. El robot Nao H21 se puede utilizar para enseñar a través de experimentos prácticos y la programación.Programación visual simple para elaborar módulos integrados, la versatilidad de Nao y su entorno de programación permite a los usuarios explorar una amplia variedad de temas en cualquier nivel de complejidad de la programación y la experiencia.(ARANCIBIA J. M., 2013). Las características y especificaciones del robot nao H21 se pueden ver en el anexo 3.

Ilustración 3 Robot nao H21

2.2 PROCESAMIENTO DIGITAL DE IMÁGENES

Antes de mencionar el proceso de segmentación y procesamiento de imágenes es necesario destacar la importancia de la representación del color y como esta influye notablemente en el modo en que se emplean y su eficacia.

2.2.1 Modelo RGB

Este modelo de color es el espacio de color más extendido y el que utiliza la gran mayoría de cámaras de video y fotográficas para construir una imagen de color, gracias a esto es su importancia en la parte de visión artificial ya que trabaja con el mismo espacio de color con el que trabajan las cámaras con las que se realiza la captura de imágenes.

Este modelo de color está representado mediante un cubo donde un color viene definido por la mezcla de valores de intensidad de tres colores primarios como lo son el rojo, verde y azul. Cada uno de los colores viene determinado por una coordenada en el cubo. El color negro se representa por (r=0, g=0, b=0) y el blanco será (r=255, g=255, b= 255)

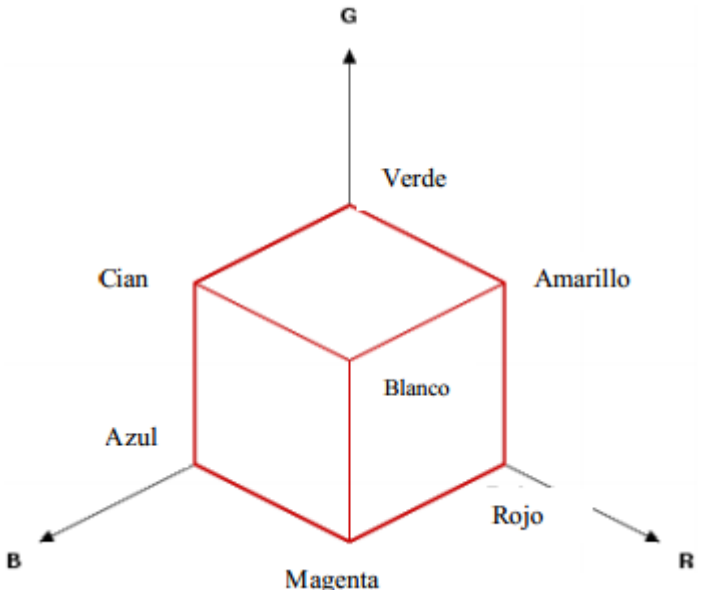


Ilustración 4 Representación del modelo RGB

2.2.2 Modelo HSV

El modelo de color HSV representa uno de los espacios de coordenadas más clásicos e intuitivos, el modo de interpretación geométrico viene determinado por un cono de base quasi-hexagonal. De este modo cada color trabaja con 3 componentes básicos: matiz, saturación y brillo.

Estas tres magnitudes pueden tener los siguientes valores:

H (color en concreto). Valores de 0-360º. La gama cromática se representa en una rueda circular y este valor expresa su posición.

S (Saturación). Valores de 0-100%. De menos a más cantidad de color.

B (Brillo). Valores de 0-100%. De totalmente oscuro a la máxima luminosidad.

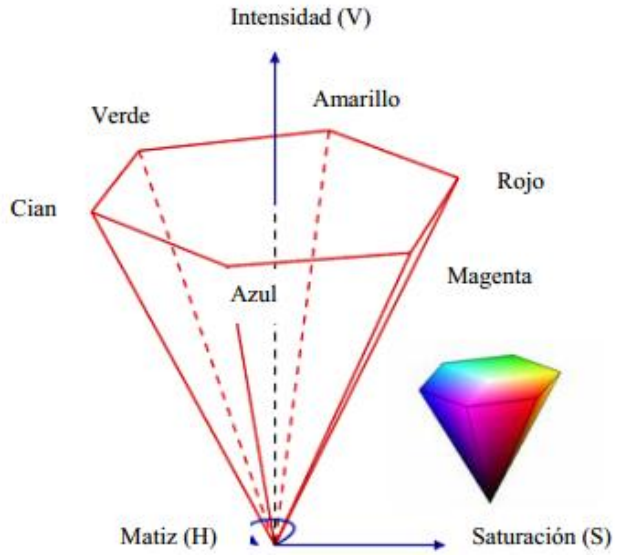


Ilustración 5Representacion del modelo HSV

2.2.3 Segmentación

La segmentación de la imagen es la primera de las operaciones y/o transformaciones que se llevan a cabo sobre la imagen recibida desde el humanoide en nuestro caso el robot nao H21. Los procesos de segmentación son de vital importancia cuando se trata de detectar objetos en entornos desestructurados. En el caso que nos ocupa, la segmentación haciendo uso de la información de color adquiere importancia en ámbitos muy diversos, desde el tratamiento de secuencias de vídeo para la detección de objetos o individuos en escenas móviles hasta la detección para realizar tracking (seguimiento), pasando por la detección de objetos para la manipulación mediante robots.

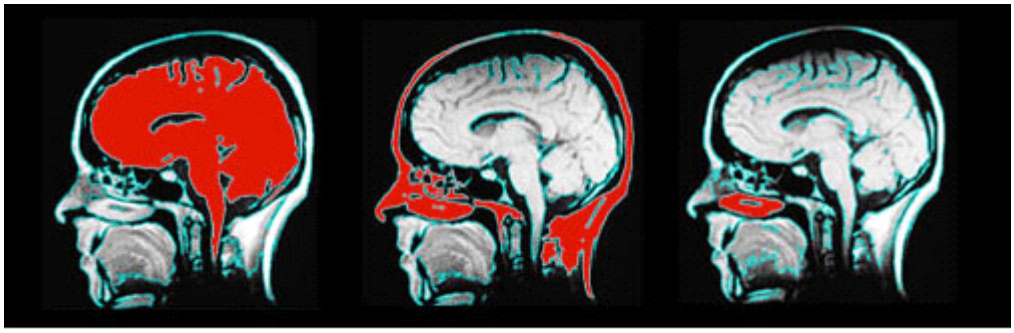


Ilustración 6 Proceso de segmentación de imágenes

2.2.3.1Segmentación basada en Pixeles

Este método de segmentación toma en cuenta sólo el valor de gris de un pixel, para decidir si el mismo pertenece o no al objeto de interés. Para ello, se debe encontrar el rango de valores de gris que caracterizan dicho objeto, lo que requiere entonces la búsqueda y el análisis del histograma de la imagen. El objetivo de este método, es el de encontrar de una manera óptima los valores característicos de la imagen que establecen la separación del objeto de interés, con respecto a las regiones que no pertenecen al mismo; debido a esta característica y si los valores de gris del objeto y del resto de la imagen difieren claramente entonces el histograma mostrará una distribución, con dos máximos distintos, lo que debiera generar, la existencia de una zona del histograma ubicada entre los dos máximos, que no presenten los valores característicos, y que idealmente fuera igual a cero.(MALPARTIDA, 2003)

2.3 MECANISMOS / TÉCNICAS DE TRACKING

2.3.1 Tracking basado en modelo

Este tipo de técnicas se basan en el conocimiento del modelo del objeto. Este modelo puede ser un modelo CAD o una proyección del objeto.Esta técnica es más robusta que la basada en características.

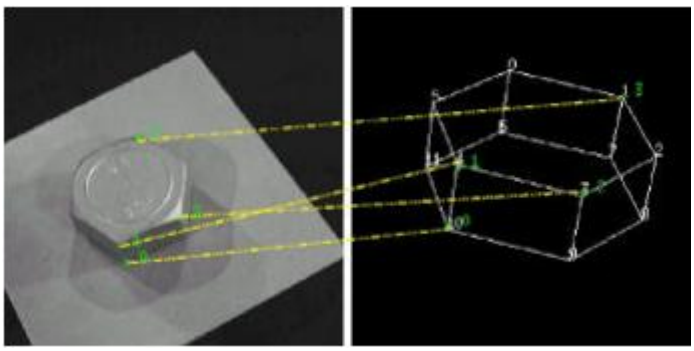


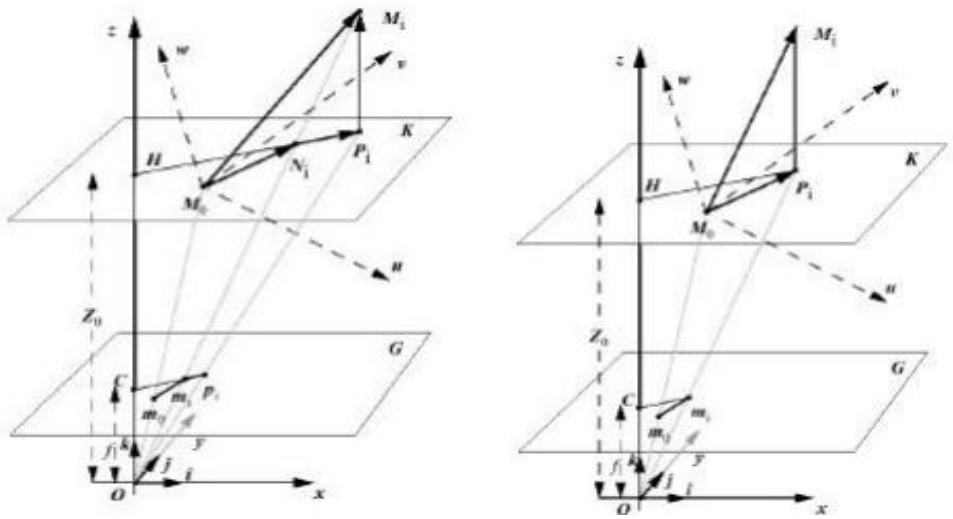
Ilustración 7 modelado de un objeto

Con la información aportada por una sola imagen, se puede obtener la pose de un objeto si conocemos su modelo.Para ello podemos emplear alguno de los algoritmos como: DeMenthon, Lowe.

Algoritmo de DeMenthon

Este algoritmo empieza con una aproximación de la proyección perspectiva, refinándola de forma iterativa hasta que el error sea menor que un determinado umbral. Para ello hemos de indicarle 4 puntos no coplanares, teniendo encuenta que la precisión de la pose obtenida dependerá de la distancia de uno de los puntos hasta el plano definido por los otros tres.

Ilustración 8 Modelo de Tracking por Algoritmo de DeMenthon



Lowe's pose estimator

Lowe propone una re parametrización de las ecuaciones de proyección para simplificar el cálculo y expresar la traslación en términos de coordenadas de la cámara.La estimación de la pose basada en modelo usando cuatro puntos se llama "problema de la perspectiva de cuatro puntos" (P4P)

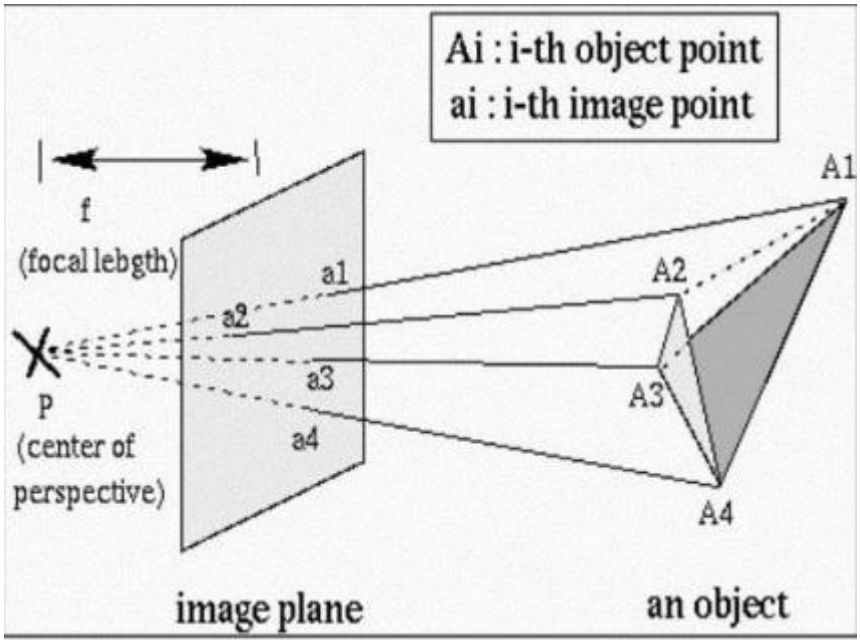


Ilustración 9 modelos de Lowe`s para sistema de tracking

2.3.2 Tracking basado en características

Este tipo de tracking se basa en las características extraídas de la imagen y no en la búsqueda de un modelo conocido.La idea fundamental en la que se basa el tracking de características es '¿por qué hacer tracking del objeto entero cuando se puede obtener el mismo resultado haciendo tracking solo de las características?Este planteamiento suele ser computacionalmentemás eficiente que el basado en modelo, pero es menos robusto.

El paso más importante antes de hacer el tracking es determinar las características a seguir. Se calcula para ello el vector gradiente de la imagenen ese punto y con los valores propios, Una vez seleccionadas las características buenas a seguir, es sencillo llevar un registro del movimiento de cada característica, con lo que tenemos determinada la trayectoria de movimiento del objeto.

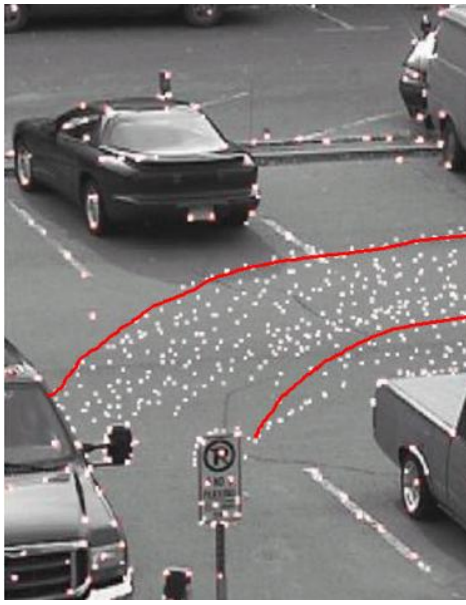


Ilustración 10trayectoria del movimiento del objeto

Predicción de la trayectoria

Se realiza cuando los requerimientos dinámicos ola velocidad del objeto a seguir es lo suficientemente alta como para que su localización entre una imagen y la siguiente de la secuencia sea muy diferente.La predicción proporciona información sobre la posible pose cuando se esté analizando una ventana de la imagen.(Carlos Pérez, 2003)

2.4 MOMENTOS

Los momentos son propiedades numéricas que se pueden obtener de una determinada imagen. El uso de momentos permite conocer bastante información de una imagen y tiene la ventaja de que no solo usa los bordes de una figura, sino que tiene en cuenta todos los píxeles de la misma. Se usan principalmente para reconocer una forma en una imagen.

2.4.1 Momentos de Legendre

La utilización de los momentos de Legendre en la manipulación de imágenes surge con la necesidad de nuevas aplicaciones, siendo la más importante de ellas la reconstrucción de una imagen a través de las características matemáticas que nos proporcionan los momentos.La reconstrucción mediante el uso de los momentos geométricos es muy costosa y sobre todo propensa a grandes errores de cálculo, lo cual la hace prácticamente inviable y poco fidedigna.(Daniel Bolaño Asenjo)

Los momentos de Legendre de orden (p,q) se definen por:

$$\lambda_{pq} = \frac{(2p+1)(2q+1)}{4} \int_{-1}^{+1} \int_{-1}^{+1} P_p(x) P_q(y) f(x,y) dxdy$$

Donde el polinomio de Legendre de orden p viene dado por:

$$P_p(x) = \frac{1}{2^p p!} \frac{d^p}{dx^p} (x^2 - 1)^p, x \in [-1,1]$$

2.4.2 momentos invariantes de HU

El inconveniente de los momentos generales reside en el hecho de que varía con la posición del objeto dentro del plano de la imagen, así como con el tamaño relativo del objeto, que depende de la distancia entre la cámara y el objeto. Los cambios en la posición del objeto se deben exclusivamente a las operaciones de giro o traslación.

De los momentos de segundo y tercer orden, pueden derivarse 7 de los llamados"momentos invariantes" que, no dependen del tamaño ni la posición del objeto, pudiendo ser usados para la identificación de objetos. Los 7 momentos invariantes se pueden ver en el anexo 4(MALPARTIDA, 2003)

2.5 CALIBRACIÓN DE CÁMARAS

2.5.1 Calibrado clásico del modelo pin-hole

En Visión Artificial se emplea frecuentemente el llamado modelo pin-hole, o alguna variación del mismo. Este modelo se basa en la forma en que se generan las imágenes en una cámara oscura, es decir, cuando tenemos un plano sobre el que se proyecta la luz que atraviesa un pequeño orificio enfrentado a este plano. En el modelo pin-hole se considera que todos los rayos de luz se proyectan sobre el sensor tras atravesar un único punto del espacio. Este punto es el llamado punto focal o centro de perspectiva de la cámara.

El modelo de la cámara pin hole describe la relación matemática que existe entre las coordenadas de un punto 3D y su proyección en el plano dela imagen, donde la apertura de la cámara es descrita como un punto infinitesimalmente pequeño y no se utiliza ningún tipo de lente para enfocar la luz. Este tipo de modelo puede ser utilizado como una aproximación de primer orden en el trazo de un mapa de escena 3D a una imagen 2D.

2.5.2 Calibrado del modelo pin-hole con distorsión.

Por lo general, una cámara real provista de óptica introduce distorsiones que no se consideran en el modelo pin-hole, y que pueden llegar a afectar de modo significativo la precisión de las transformaciones de perspectiva. Habitualmente, la distorsión tangencial se ignora por ser poco significativa, y tan solo se tiene en cuenta la distorsión radial. El método de calibrado que se emplea generalmente cuando se quiere tener en cuenta la distorsión radial es el método en dos fases o alguna variación del mismo.

2.5.3 Modelo de Tsai

En el método de Tsai, o método de las dos fases se emplea como base el modelo clásico de cámara pin-hole, corrigiendo el mismo a partir de un modelo de la distorsión. El método de Tsai considera cuatro pasos en la transformación de las coordenadas de un punto de la escena a las coordenadas en la imagen del mismo.

2.6 ROBOCUP

La RoboCup surge a partir de una iniciativa de diversos grupos de investigación interesados en el estudio y desarrollo de la IA (Inteligencia Artificial) particularmente, en el comportamiento inteligente que pueden exhibir agentes artificialessimulados en software o implementados en hardware (robots). Más que una competición, la RoboCup se plantea como meta lograr que a mediados del sigloXXI (año 2050), sea posible enfrentar al equipo campeónmundial de fútbol asociado con un equipo de agenteshumanoides completamente autónomos, que sean capaces de vencerlo. Desde 1997 cada año seorganiza un evento académico en el que se presentan losúltimos adelantos, prototipos, estrategias, técnicas yherramientas en forma de torneos, competencias y simulaciones. Sebusca de esta forma fomentar la investigación y eldesarrollo de nuevas tecnologías inteligentes, no solamenteenmarcadas dentro del juego del balompié, sino tambiéncon posibilidad de aplicación a otros dominios comonavegación autónoma y tareas de búsqueda y rescate en desastresnaturales.(Vega, 2013)

En la Liga Humanoide, equipos de investigación compiten en diferentes áreas de investigación, estilo de caminar, correr y golpear la pelota, mantenimiento del equilibrio, percepción visual de la pelota, percepción de los otros jugadores y el campo, auto-localización, y el juego en equipo. La competencia se divide en varias ligas, de acuerdo al tamaño de los robots: Simulación, tamaño pequeño, tamaño mediano y plataforma estándar.

En la plataforma estándar (Standar Platform League) solo puede ser utilizado un tipo de robot para todos los equipos. En la actualidad el robot indicado por los organizadores es el robot NAO que sustituyó al robot AIBO de Sony de cuatro patas, esta liga se centra más en el desarrollo de software.(Gómez,

Ilustración 11 futbol robótico evento de la RoboCup

Optimización y modelado del movimiento de un robot bípedo NAO usando técnicas de IA, 2012)

3. METODOLOGÍA

3.1 CONEXIÓN PYTHON Y ROBOT NAO

Para poder comenzar a controlar el NAO directamente desde Python es necesario establecer una conexión entre la máquina y el humanoide, para esto es necesario conocer el SDK Naoqui y el procedimiento para manipular el robot desde la máquina.

NAO se suministra con un framework de desarrollo llamado NaoQi. Éste es un programa que implementa una gran cantidad de funciones de alto nivel y se encarga de comunicarse con los dispositivos de bajo nivel. Para ejecutar órdenes sobre el robot Nao es necesario hacerlo a través de NaoQi. Se puede hacer de forma remota (desde el PC) usando lenguajes de programación como C++, Python o URBI. También se pueden compilar programas para ejecutarse desde Nao. Existen también varios entornos de simulación, siendo Webots el más usado en este tipo de robots. Webots es un entorno de desarrollo para modelar, programar y simular robots móviles. La programación del robot NAO se lleva a cabo utilizando brokers, módulos y bibliotecas.

La API de Python para la NAO nos permite crear módulos de Python que se pueden ejecutar de forma remota o en el robot. El enfoque básico es: Importar un ALProxy, Crear un ALproxy al módulo que desea utilizar, Llamar un método Esto se puede ver en la siguiente imagen:

```
from naoqi import ALProxy
tts = ALProxy("ALTextToSpeech", "<IP of your robot>", 9559)
tts.say("Hello, world!")
```

Normalmente, cada módulo es una clase dentro de una librería. Cuando la librería se carga desde el autoload.ini, automáticamente se crea una instancia del módulo.Un módulo puede ser local o remoto.

Si es remoto, se compila como un archivo ejecutable, y se puede ejecutar fuera del robot. Los módulos remotos son más fáciles de usar y se puede depurar con facilidad desde el exterior, pero son menos eficientes en términos de velocidad y en el uso de la memoria.

Si es local, se compila como una biblioteca, y sólo puede ser utilizado en el robot. Sin embargo, son más eficientes que un módulo de control remoto.

3.2ACTIVACIÓN Y CAPTURA DE IMÁGENES DESDE EL ROBOT NAO H21

El robot nao H21 cuenta con dos cámaras, en la parte frontal y en la parte del mentón, la cámara que siempre estará activa por defecto será la cámara superior del robot, para nuestro caso es necesario poder activar la cámara del

mentón, ya que por medio de esta podrá obtener un mejor campo visual a la hora de buscar e identificar el objetivo que será para este caso el balón.
El procedimiento empleado se puede visualizar en la figura 12 en la cual se pueden apreciar las diferentes etapas para realizar la ejecución del cobro de tiro penalti con el robot nao H21.

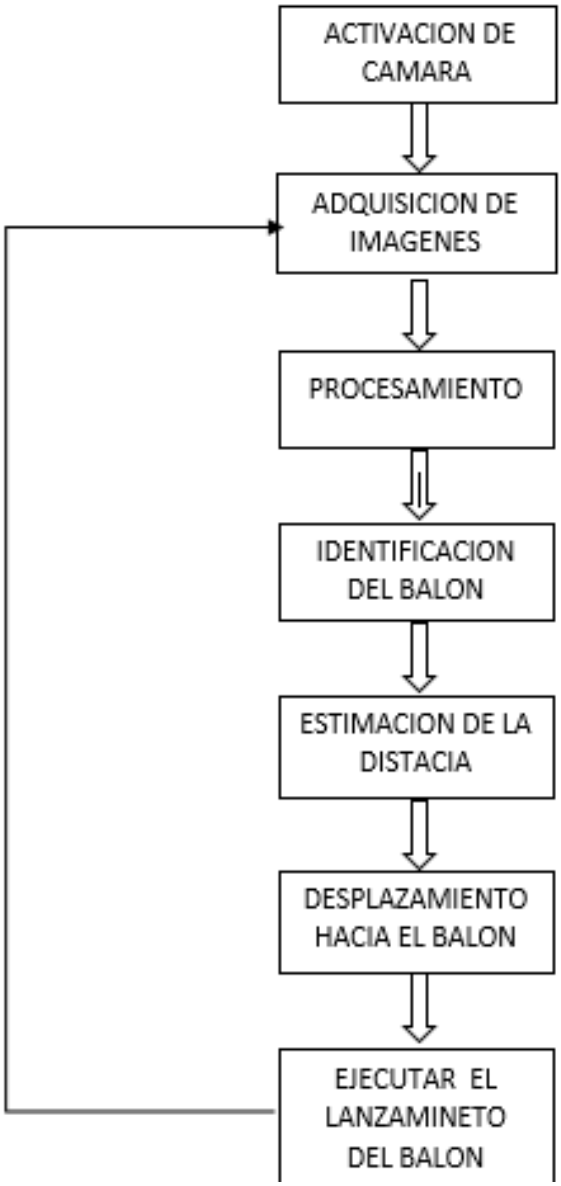


Ilustración 12 Diagrama de flujo del procedimiento principal

3.2.1 Selección de cámara del mentón y captura de imágenes

La cámara ubicada en la parte del mentón del robot nao permite obtener mayor rango de visión para detectar y realizar un proceso de tracking del balón, para ello se activa la cámara del mentón desde Python teniendo en cuenta la

documentación suministrada por la comunidad de Aldebarán robotic[6] . Este procedimiento se verá en el anexo 5.

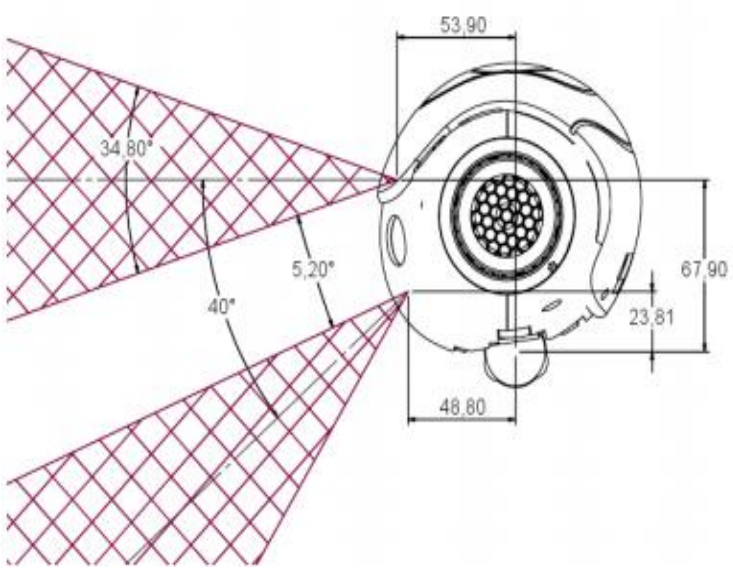


Ilustración 13camaras del NAOH21

Captura de imágenes

Para realizar este proceso inicialmente se utilizó una cámara web de un portátil para realizar las primeras pruebas y a continuación se hizo uso de las dos cámaras de las que dispone el robot NAO.
Para realizar la captura de imágenes a través de la cámara del robot nao desde Python, se toma una imagen y se muestra en un screen,PIL[7]. se guarda en un formato de imagen determinado, el código realizado en Python se puede observar en el anexo 5.

3.2.2 Procesamiento

A partir de la librería de open cv llamamos la imagen que fue capturada anteriormente desde el nao para realizar los siguientes procesos:
Primero se aplica un filtro para atenuar el ruido presentado en la imagen con la siguiente función, **cv2 . GaussianBlur (im, (15,15) ,10).** La cual es un filtro gaussiano que es un efecto de suavizado para mapas de bits generado por software de edición gráfica.

El siguiente procedimiento es convertir la imagen del espacio de color RGB a HSV, esto se logra mediante la siguiente función, **cv2.cvtColor**

[6] Comunidad Aldebarán robotic documentación y software robot nao en : https://community.aldebaran.com/doc/1-14/dev/python/examples/vision/get_image.html

[7] PIL, El Python Imaging Library (PIL) añade capacidades de procesamiento de imágenes para el intérprete de Python. Esta biblioteca es compatible con muchos formatos de archivo, y proporciona capacidades de procesamiento de imágenes y gráficos potentes.

(open_cv_image, cv2.COLOR_BGR2YUV), ya que el modelo HSV nos permite manipular los valores dematiz, saturación y brillo según la imagen que estemos tratando,este proceso se puede observar en la figura 14.

Ilustración 14Tranformacion de RGB a HSV respectivamente

A continuación se realiza un filtrado del color que deseemos reconocer, para esto se crean dos matrices una para obtener el color más alto y otra para el color más bajo respectivamente. cv2.inRange (hsv, matriz 1_, matriz 2__), con esta función se realiza el barrido de la imagen obteniendo el color deseado.

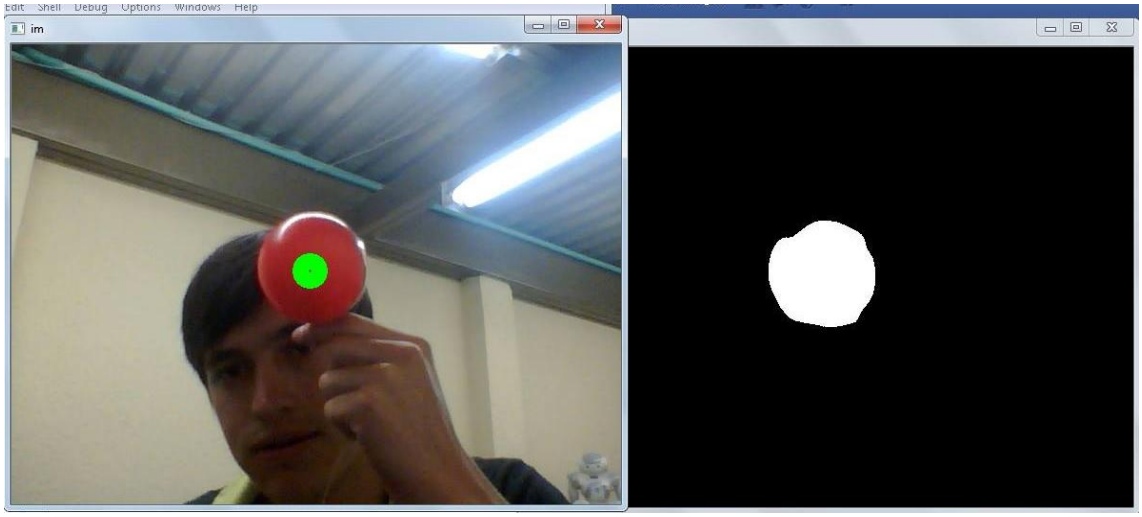


Ilustración 15 imanen en RGB y mascara HSV

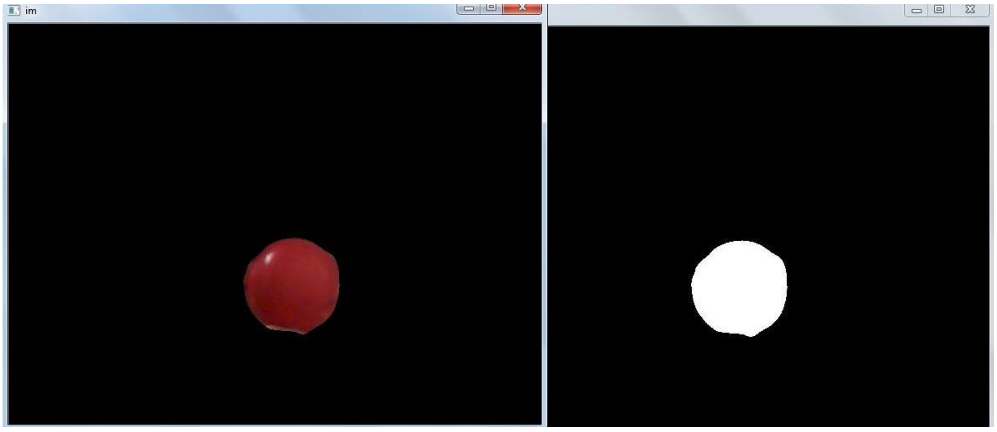


Ilustración 16 imagen original filtrada en HSV y mascara en HSV

Teniendo el color filtrado de la imagen procedemos a hallar el área del balón y las coordenadas donde está ubicado. Para este caso utilizamos los momentos de la imagen, en el anexo 4 estará la explicación de los momentos de la imagen. El área es calculada para estimar la distancia a la que se encuentra el balón respecto a la cámara del robot.

La librería de Open cv nos permite hallar los momentos de la imagen empleando la siguiente función. *M= cv2.moments (mascara en HSV),* donde obtenemos como resultado los siguientes valores de la imagen.

M ['m00'] = 3802560.0
Este momento cuenta los pixeles que conforman el balón y nos determina el área del balón.

M ['m10'] = 960652320.0
Este momento dividido en el momento que nos determina el área del balón nos dará la coordenada en x.

M ['m01'] = 1127106630.0
Este momento dividido en el momento que nos determina el área del balón nos dará la coordenada en y.

Los momentos invariantes son utilizados para determinar la forma del objeto, desde la librería de open cv podemos realizar el cálculo de los momentos invariantes con la siguiente función: *HU**=cv2.HuMoments (M).*** Este proceso nos arroja los siguientes resultados:

```
[  6.26552102e-04]
 [  8.43829657e-11]
 [  1.66448663e-12]
 [  1.26493889e-15]
 [  1.31742270e-29]
 [ -8.35099317e-21]
 [ -5.65273930e-29]]
```

A partir de los resultados obtenidos en las pruebas realizadas en el laboratorio podemos concluir que el primer momento es el más invariante y el que mejor se adecua a nuestro proyecto, cuando se está detectando un objeto de forma circular los momentos 4 y 5 son cero (0)

3.2.3 Estimación de la distancia

Para hallar la distancia desde la cámara ubicada en el mentón del robot NAO H21 hacia el balón, se realiza tomando una serie de fotos en la que se varía la distancia del balón respecto a la cámara del NAO, se hace un conteo de pixeles en cada una de las imágenes y se procede a realizar la gráfica.

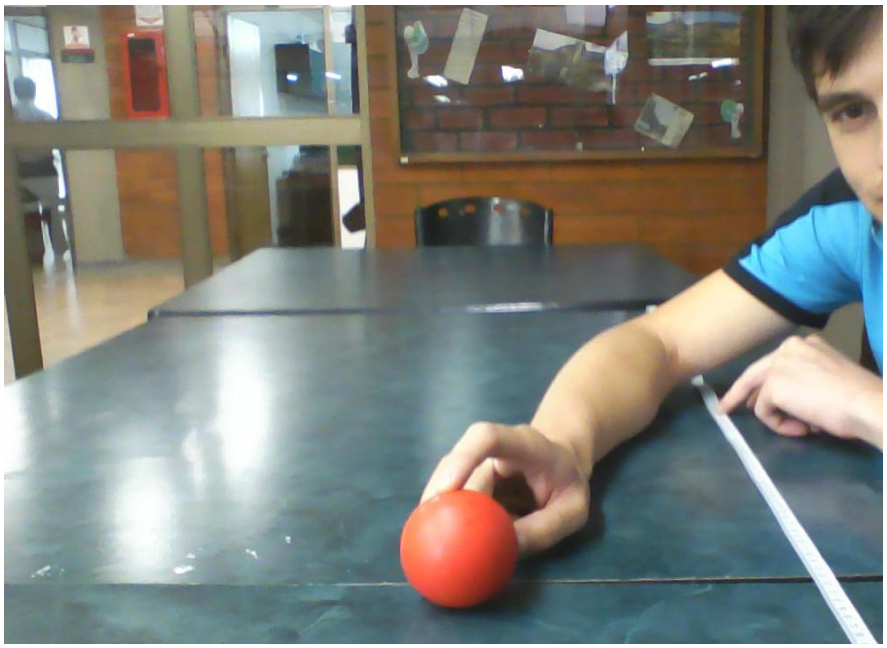


Ilustración 17 imagen tomada a 50 cm

Ilustración 18 imagen tomada a 70 cm

Ilustración 19 imagen tomada a 90 cm

Ilustración 20 imagen tomada a 110 cm

Ilustración 21 imagen tomada a 130

Después de haber realizado el conteo de pixeles se grafica estos valores respecto a la distancia, la gráfica obtenida se muestra a continuación:

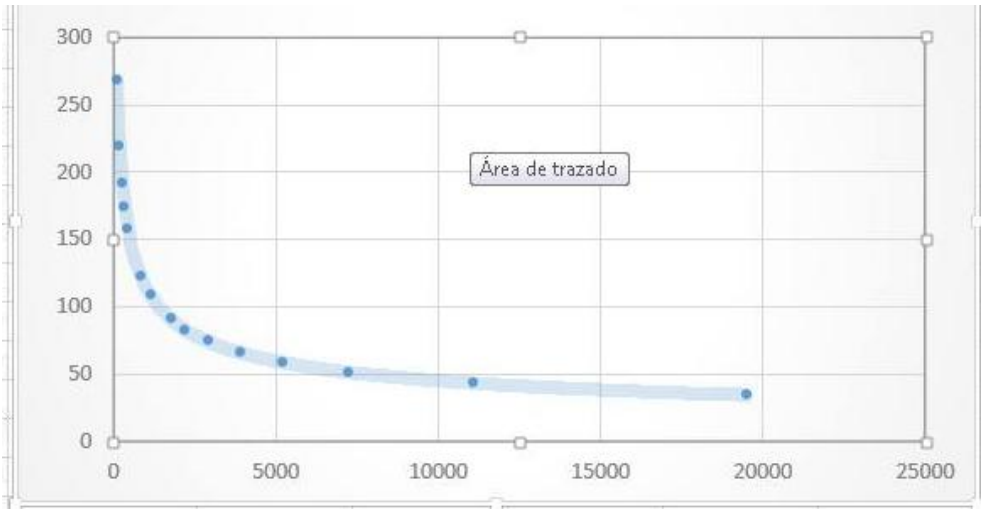


Ilustración 22 grafica (distancia vs nº pixeles)

A partir de los datos obtenidos y la gráfica número 22 se realiza una aproximación exponencial.

Para determinar la distancia se hace uso del triángulo de Pitágoras, donde se conocen las distancias de la cámara del robot nao ubicada en el mentón hacia el piso y la distancia de la cámara hacia el balón, hallada por medio de la aproximación exponencial. A partir de estos valores se realiza el cálculo de la distancia del pie del robot NAO hacia el balón.

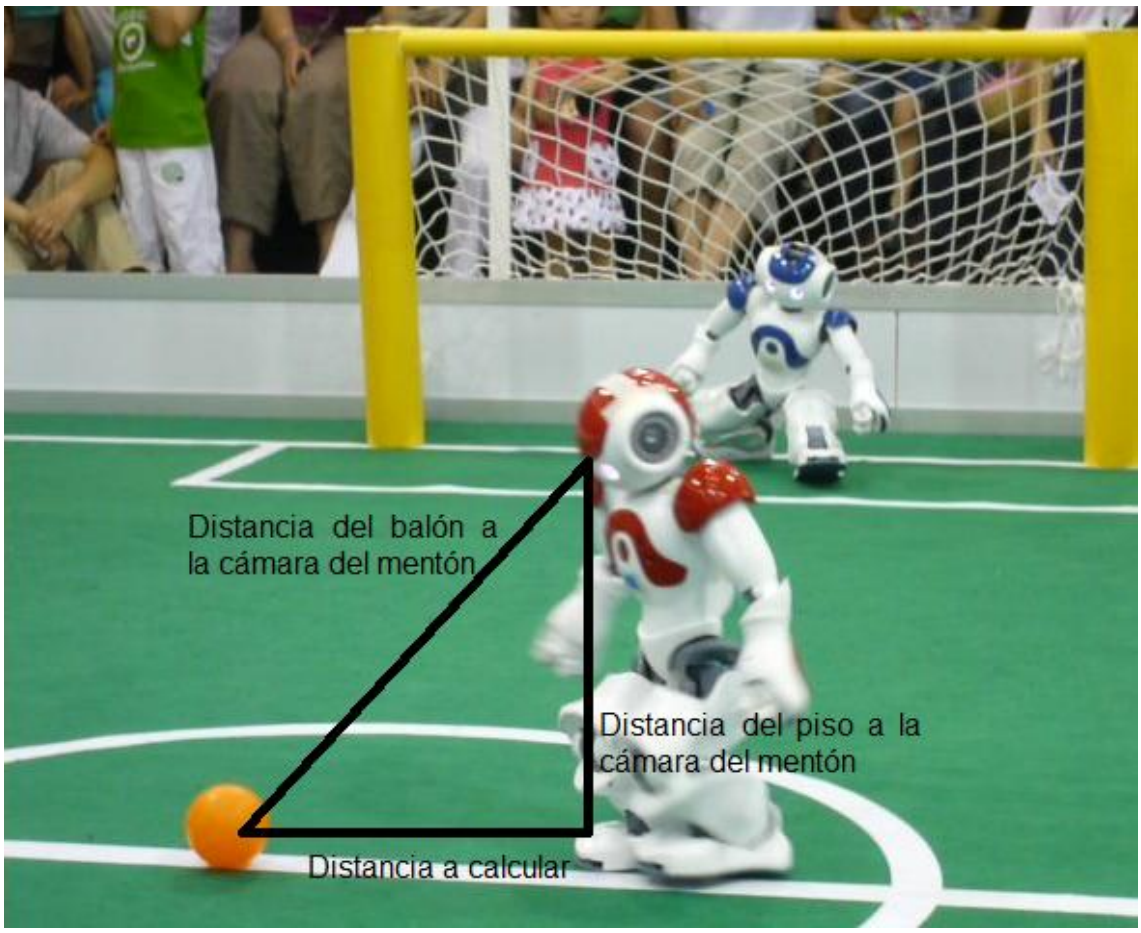


Ilustración 23cálculo de la distancia

En el anexo número 6 se encuentra el código empleado para la estimación de la distancia.

3.2.4 Desplazamiento hacia el balón

Teniendo en cuenta la ubicación del balón y el cálculo de la distancia hallado anteriormente el robot realiza un proceso de búsqueda mediante un algoritmo de tracking que se encuentra en el anexo 7

Este algoritmo permite al robot ubicarse enfrente del balón, cualquiera que sea su posición de partida en la cancha, para desplazarse y poder realizar el lanzamiento del tiro penalti.

Ilustración 24 búsqueda y desplazamiento hacia el balón

CONCLUSIÓN

Durante este trabajo de investigación se desarrollaron algoritmos escritos en el lenguaje de programación de Python, contando con la librería de visión artificial de OpenCV, que permitieron cumplir con el objetivo principalde diseñar un algoritmo que permitiera al robot humanoide NAO H21 realizar cobros de tiro penalti.

Este objetivo se cumplió teniendo en cuenta algunos conceptos fundamentales como, visión artificial, procesamiento y filtrado de imágenes, reconocimiento de objetos, logrando que el robot NAO realizara un proceso de tracking para buscar e identificar el balón, a su vez se desarrolló un módulo de desplazamiento hacia el balón, teniendo en cuenta el cálculo de la distancia para poder ubicarse siempre en la misma posición y de este modo poder efectuar el cobro de tiro penalti de una manera optima

Referencias

Addison-Wesley. (1993). “A Guided Tour to Computer Vision”.

Alberto, I. N. (s.f.). *Matrices Homogéneas.* Obtenido de http://proton.ucting.udg.mx/~cin/robotic/tarease/dh/dh.htm

ARANCIBIA, J. M. (Marzo de 2013). Obtenido de http://www.tesis.uchile.cl/bitstream/handle/2250/113110/cf-yanez_ja.pdf?sequence=1

ARANCIBIA, J. M. (MARZO 2013). *DETECCION ROBUSTA DE OBJETOS EN ROBOTS HUMANOIDES.* CHILE.

Baeza, A. P. (septiembre de 2010). *Departamento de Ingeniería de Sistemas y Automática.*

Carlos Pérez, O. R. (23 de Mayo de 2003). *Técnicas de tracking.* Obtenido de http://www.disc.ua.es/controlvisual/ponencias/SVCPCMumh.pdf

Contreras Tapia Luis M., G. V. (2012). *Analisis Cinem ´atico Robot NAO Aldebaran.* Obtenido de http://zeus.lci.ulsa.mx/portales/cidit/papers/Paper_01_04.pdf

Daniel Bolaño Asenjo, J. J. (s.f.). *MOMENTOS.*

Estrada, L. A. (junio de 2012). Un robot ayuda a los niños autistas. *QUO*.

Gálvez Cobo, J. D. (27 de septiembre de 2012). *Universidad Carlos III de Madrid.* Obtenido de http://e-archivo.uc3m.es/handle/10016/16336

Gómez, J. B. (2012). *Optimización y modelado de un robot bípedo NAO usando técnicas de IA.*

Gómez, J. B. (2012). *Optimización y modelado del movimiento de un robot bípedo NAO usando técnicas de IA.*

MALPARTIDA, E. A. (2003). *SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBOT.* peru.

Vega, S. A. (2013). *Balompié Simulado: Un Nuevo Desafío para Agentes Artificiales Inteligentes.*

ANEXOS

Anexo 1

Introducción OpenCV

OpenCV (Open Source Computer Vision) es una librería de visión por computador desarrollada por Intel en 1999. Una de sus principales ventajas es poder ejecutarse en ordenadores de bajas características. Esta librería ha sido desarrollada para poder ser empleada en diferentes lenguajes de programación como C++, Python, Java

OpenCV es principalmente una librería que implementa algoritmos para las técnicas de la calibración (Calibración de la Cámara), detección de rasgos, para rastrear (Flujo Óptico), análisis de la forma (Geometría, Contorno que Procesa), análisis del movimiento (Plantillas del Movimiento, Estimadores), reconstrucción 3D (Transformación de vistas), segmentación de objetos y reconocimiento (Histograma, etc.).

OpenCV en cuanto a análisis de movimiento y seguimiento de objetos, ofrece una funcionalidad interesante. Incorpora funciones básicas para modelar el fondo para su posterior sustracción, generar imágenes de movimiento MHI (Motion History Images) para determinar dónde hubo movimiento y en qué dirección, algoritmos de flujo óptico, etc.

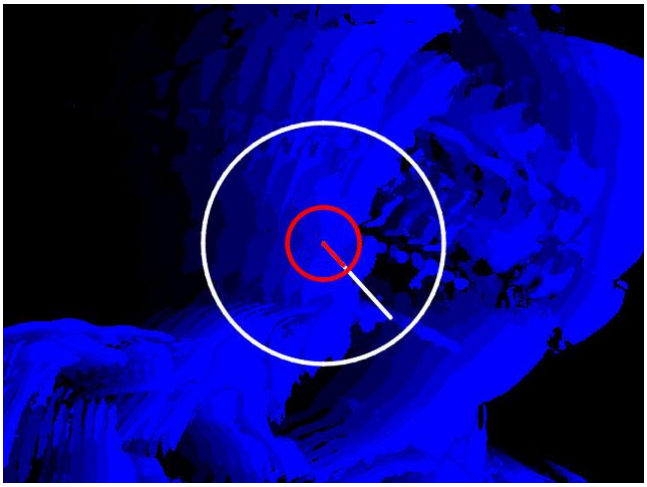


Ilustración 25 imágenes detección de movimiento con open cv

Anexo 2

Instalación OpenCV para Python en Windows

Para utilizar OpenCV en Python es necesario realizar los siguientes pasos.

1.- Descargar OpenCV en su versión más reciente desde el siguiente link de descarga: http://opencv.org/

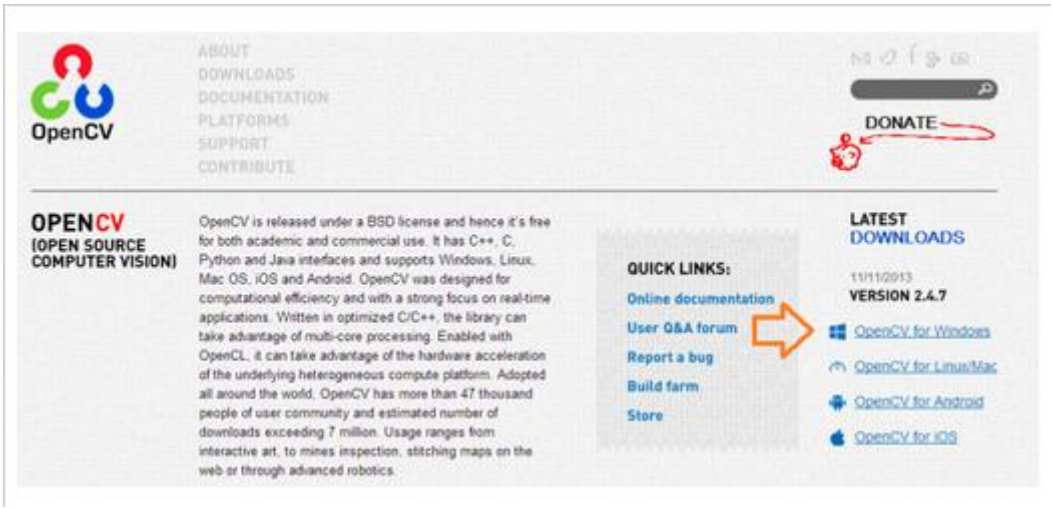


Ilustración 26 Descargar OpenCV

2.- Instalar OpenCV preferiblemente en la ruta de C:\

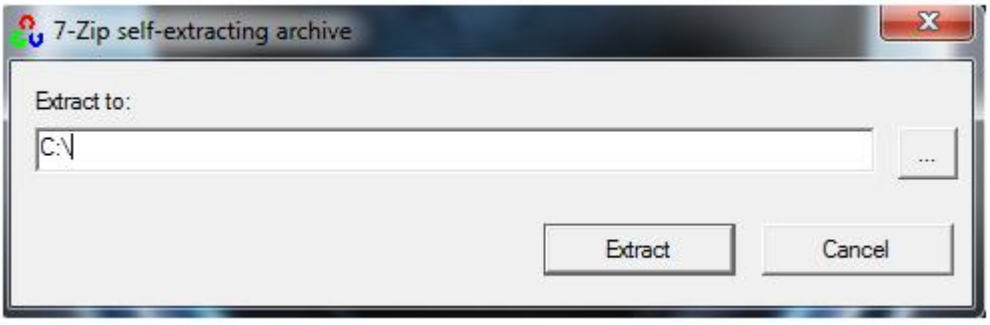


Ilustración 27 Instalar OpenCV

3.- Descargar Python 2.7.x para Windows (OpenCV es más estable con la versión de 32 bits) desde: http://www.python.org/

Ilustración 28Descargar Python 2.7.x

4.- Instalar Python 2.7.x dando la ruta por defecto, con ello Python se instalará en C:/Python27

5.- Descargar Numpy y Matplotlib desde los siguientes links:

- http://sourceforge.net/projects/numpy/files/NumPy/1.8.0/numpy-1.8.0-win32-superpack-python2.7.exe/download?use_mirror=softlayer-ams

- http://sourceforge.net/projects/matplotlib/files/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe/download?use_mirror=softlayer-dal

6.- Instalar Numpy y Matplotlib con los valores por defecto.

7.-dirigirse a C:\opencv/build/python/2.7/x86 y copiar el archivo cv2.pyd a C:/Python27/lib/site-packeges

8.- Abrir **Python** (command line).

Ilustración 29 Python command line

9. Escribir en la ventana de comandos de Python lo siguiente:

```
>>> import cv2

>>> print cv2.__version__
```

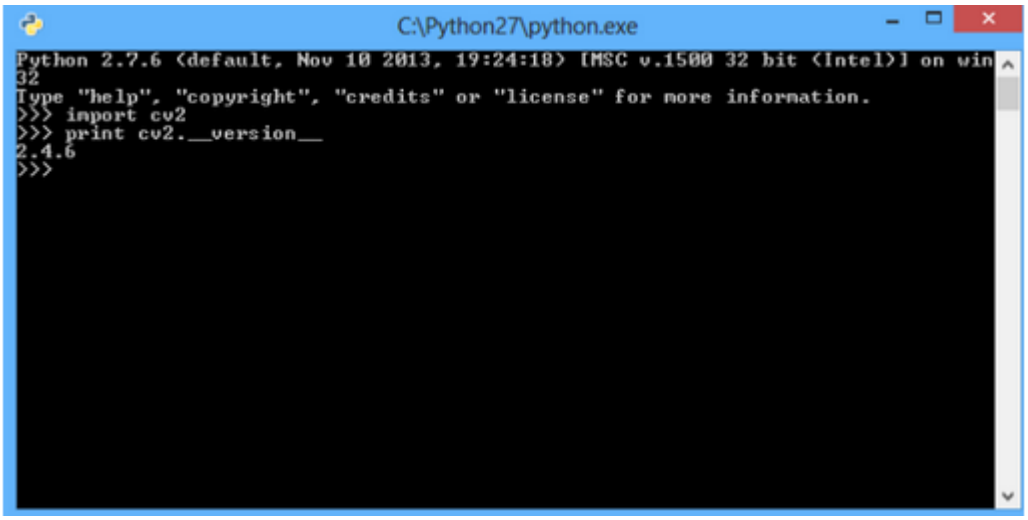


Ilustración 30 Agregar biblioteca Open CV

Al finalizar este proceso ya podremos utilizar OpenCV mediante Python.

Anexo 3

CARACTERÍSTICAS TÉCNICAS ROBOT NAO H21

-21 Grados de Libertad
-Caminado omnidireccional
-CPU ATOM Z530 1.6 GHz
-Memoria Flash de 256 MB SDRAM / 2 GB
-Sensor de Inercia con Giroscopio de dos ejes y Acelerómetro de 3 ejes.
-1x Puerto Ethernet RJ45 – 10/100/1000 base T y Wi-Fi IEEE 802.11b/g
-2x Cámaras de video (960p@30fps), mejor sensibilidad en VGA. Visión horizontal de 239°, visión vertical de 68°. Resolución de alta definición.
-Capacidad de procesamiento de visión
-Reconocimiento de objetos
-Detección y reconocimiento de rostros
-Texto a voz:
-Dos altavoces y síntesis vocal multi-idioma (Español e Inglés precargados)
-Cuatro micrófonos y reconocimiento de voz multi-idioma (Español e Inglés precargados)
-Soporta múltiples lenguajes de programación.
-Cuenta con software especial de programación y simulación.

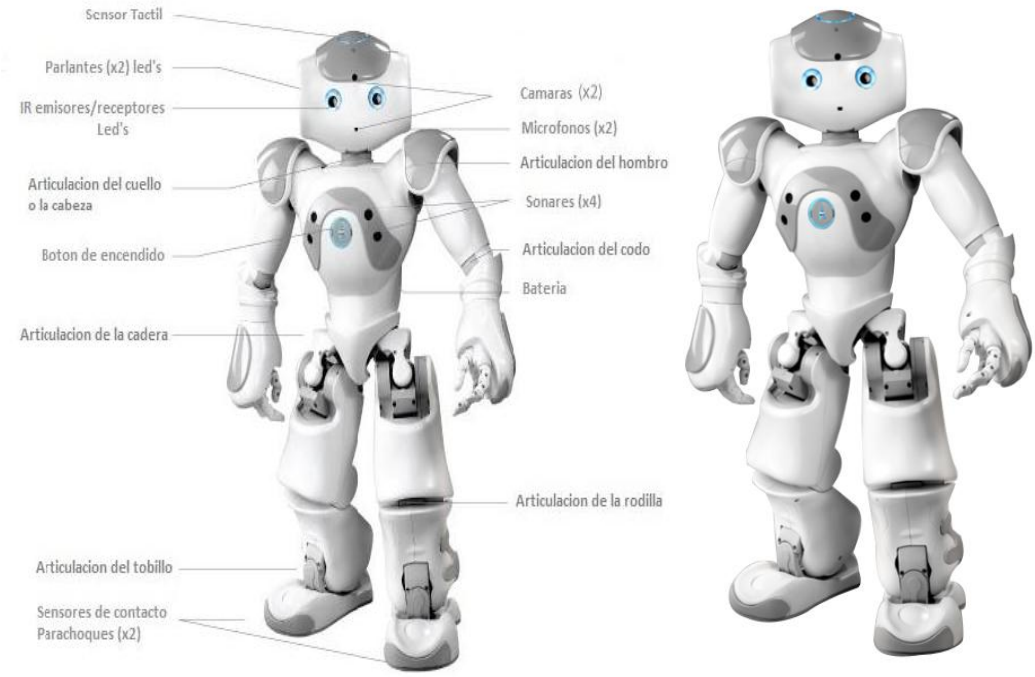


Ilustración 31especificaciones robot nao h21

Anexo 4.

Los 7 momentos invariantes de HU

De los momentos de segundo y tercer orden, pueden derivarse 7 de los llamados "momentos invariantes" que, no dependen del tamaño ni la posición del objeto, pudiendo ser usados para la identificación de objetos. El siguiente conjunto de momentos invariantes se puede obtener usando únicamente los momentos centrales normalizados de orden 2 y 3

$$\phi_1 = n_{20} + n_{02}$$

$$\phi_2 = (n_{20} - n_{02})^2 + 4n_{11}^2$$

$$\phi_3 = (n_{30} - 3n_{12})^2 + (3n_{21} - n_{03})^2$$

$$\phi_4 = (n_{30} + 3n_{12})^2 + (n_{21} + n_{03})^2$$

$$\phi_5 = (n_{30} - 3n_{12})(n_{30} + n_{12})\left[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\right] + (3n_{21} - n_{03})(n_{21} + n_{03})\left[3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\right]$$

$$\phi_6 = (n_{20} - n_{02})\left[(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\right] + 4n_{11}(n_{30} + n_{12})(n_{21} + n_{03})$$

$$\phi_7 = (3n_{21} - n_{03})(n_{30} + n_{12})\left[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\right] + (3n_{12} - n_{30})(n_{21} + n_{03})\left[3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\right]$$

Ilustración 32 7 momentos invariantes de HU

Anexo 5.

Mediante el siguiente algoritmo se realiza la activación de la cámara del robot nao H21 ubicada en el mentón,

*iactivar_camara.py - C:\Use

File Edit Format Run Options Windows Help

```
#librerias de python
import cv2 as cv
import numpy
from naoqi import ALProxy
import vision_definitions
def  main():
    IP = "colocar IP del nao"
    PORT = 9559
    camProxy = ALProxy("ALVideoDevice", IP, PORT)
    camProxy.kCameraSelectID=18
    camProxy.setParam(camProxy.kCameraSelectID, 1)
    resolution = 2    # VGA
    resolution = vision_definitions.kQVGA

    colorSpace = 11   # RGB
    #captura = cv.VideoCapture(1)
    videoClient = camProxy.subscribe("python_GVM1", resolution, colorSpace, 30)

    while(1):
        naoImage = camProxy.getImageRemote(videoClient)
        imageWidth = naoImage[0]
        imageHeight = naoImage[1]
        array = naoImage[6]

      # Create a PIL Image from our pixel array.
        im = Image.fromstring("RGB", (imageWidth, imageHeight), array)
        open_cv_image = numpy.array(im)
        open_cv_image = open_cv_image[:, :, ::-1].copy()


        cv.imshow('im',open_cv_image)
        k = cv.waitKey(5) & 0xFF
        camProxy.releaseImage(videoClient)
        if k == 27:
            break
    cv.destroyAllWindows()
    camProxy.unsubscribe(videoClient)

main()
```

Ilustración 33 algoritmo de activación de cámara del mentón

Anexo 6

En el siguiente Algoritmo se demuestra el proceso de tracking, el cálculo de la distancia hacia el balón para llegar a una medida determinada y proceder a realizar el cobro del tiro penalti

Parte 1

```
# librerias
import Image
import numpy
from naoqi import ALProxy
import vision_definitions
import sys
import math
from naoqi import ALProxy
import time
import cv2
import cv
import random
robotIp = "colocar IP del nao"
PORT = 9559
camProxy = ALProxy("ALVideoDevice", robotIp, PORT)
motion  =  ALProxy ( "ALMotion" ,  robotIp ,  9559 )
camProxy.kCameraSelectID=18 #camara del menton
camProxy.setParam(camProxy.kCameraSelectID, 1)
def StiffnessOn(proxy):
    pNames = "Body"
    pStiffnessLists = 1.0
    pTimeLists = 1.0
    proxy.stiffnessInterpolation(pNames, pStiffnessLists, pTimeLists)
try:
    motionProxy = ALProxy("ALMotion", robotIp, 9559)
except Exception, e:
    print "Could not create proxy to ALMotion"
    print "Error was: ", e
try:
    postureProxy = ALProxy("ALRobotPosture", robotIp, 9559)
except Exception, e:
    print "Could not create proxy to ALRobotPosture"
    print "Error was: ", e
try:
    proxy = ALProxy("ALMotion", robotIp, 9559)
except Exception, e:
    print "Could not create proxy to ALMotion"
    print "Error was: ", e
   #colocar nao en rigidez
StiffnessOn(motionProxy)
postureProxy.goToPosture("StandInit", 0.6)
```

Parte 2

```
resolution = 2    # VGA
resolution = vision_definitions.kQVGA
colorSpace = 11   # RGB
videoClient = camProxy.subscribe("NAOo", resolution, colorSpace, 30)
motionProxy.moveInit()
while(1):
    naoImage = camProxy.getImageRemote(videoClient)
    imageWidth = naoImage[0]
    imageHeight = naoImage[1]
    array = naoImage[6]

  # Crear una imagen de PIL de nuestra matriz de píxeles.
    im = Image.fromstring("RGB", (imageWidth, imageHeight), array)
    open_cv_image = numpy.array(im)
    open_cv_image = open_cv_image[:, :, ::-1].copy()

    im_suavi = cv2 . GaussianBlur ( open_cv_image,( 15,15 ),10)#filtro
    hsv = cv2.cvtColor(im_suavi, cv2.COLOR_BGR2YUV)
    #Variables
    izq=0
    ar=0
    cx=0
    cy=0
    x1=0
    a=0
    rojo_ = numpy.array([50,0,200])
    rojo__ = numpy.array([255,255,255])
    mask =cv2.inRange(hsv,  rojo_,rojo__)
    res = cv2.bitwise_and(open_cv_image ,open_cv_image ,mask=mask)

    area = cv.CountNonZero(mask)
    M = cv2.moments(mask)
    HU=cv2.HuMoments ( M)
    try:
        cx = (M['m10']/M['m00'])
        cy = (M['m01']/M['m00'])
    except:
        print ""
    else:
        print ""
        cx=int (cx)
```

Parte 3

```
        i=0

        if cx >=160:
            i=160-cx
            #i=cx-320

        else:
            i=160-cx
            #i=cx-320
        izq=(i*60)/320
##
        j=0
        if cy >=120:
            j=cy-120
        else:
            j=cy-120
        ar=(j*40)/240
        g=((izq*0.59)/25)
        a=(ar*0.45)/20
        cv2.circle(open_cv_image ,(cx,cy), 10, (0,255,0),thickness=15, lineType=8, shift=0)
    dd=0
    dist=0
    try:

        dist=(1861.5*(pow(area,-0.404)))

    except:
        print "error"
    else:
        print " "
    x=dist-46

    print x

    try:
        x1=math.sqrt(x)
    except:
        print 'erroe'

    else:
        print x1,dist
```

Parte 4

```
    x1=int (x1)
    if x1 in range (-7,1):
         print 'llege'
         d1=0
         g=0
    else :

        d1=((x1*0.8)/(x1+0.1))
        print 'dist',d1,x1

    if cx in range(156,164) and cy in range (116,124):

        print 'siiii', cx,cy

    else:

        X         = random.uniform(d1,d1)   #camina
        Y         = random.uniform(0,0)    #lados
        Theta     = random.uniform(g,g)    #vueltas
        Frequency = random.uniform(0.1,0.1)
        motionProxy.setWalkTargetVelocity(X, Y, Theta, Frequency)


        cd=motionProxy.setAngles("HeadYaw", random.uniform(g,g), 0.05) #mov izq-der
        ary=motionProxy.setAngles("HeadPitch", random.uniform(a,a),0.05)#mow arri-aba
        time.sleep(0.2)


        print 'noooo'
        print g,a
    cv2.imshow('mascara',res)
    cv2.imshow('im',open_cv_image )
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
      break
 # Desactivar el traking de la cabeza
motionProxy.stopMove()
cv2.destroyAllWindows()
camProxy.unsubscribe(videoClient)
```