CHAPTER X

Sistema de Visión del Humanoide HOAP-3 para la Detección e Identificación de Objetos Mediante Librerías OpenCV

A.PEÑA[1], D.HERNÁNDEZ GARCÍA[1], M.GONZÁLEZ-FIERRO[1], P.PIERRO[1] y C. BALAGUER[1]

[1]Robotics Lab, Universidad Carlos III de Madrid.

En este artículo presentamos un sistema de visión en el que se identifican varios objetos de distintas formas y colores y se calcula la distancia del objeto al robot. La plataforma utilizada para implementar el algoritmo ha sido el humanoide HOAP-3. El algoritmo es capaz de detectar simultáneamente varios objetos y sus respectivas distancias a la vez. Así mismo se ha creado una interfaz donde se muestran los resultados al usuario.

1 Introducción

Las aplicaciones robóticas futuras prevén un mundo en que los robots interactúen naturalmente con los humanos en el hogar, la oficina, centros de entretenimiento, etcétera, ayudando en la realización de actividades comunes y mejorando la calidad de vida. Aunque continuamente ocurren grandes avances en el campo de la robótica, aún estamos lejos de conseguir un robot capaz de tal destreza y autonomía. Para llegar a este nivel de adaptación de los robots al mundo humano se pueden considerar dos grandes aspectos, la capacidad de los robots para desplazarse e interactuar con el entorno, y la capacidad del robot para percibir y comprender ese entorno. En los humanos uno de los sistemas de percepción más importante es la visión. En este trabajo se desarrolló un sistema de visión por computador, basado en las librerías OpenCV, que permite a un robot, HOAP-3, detectar e identificar una serie de objetos en su entorno.

La visión por computador es campo de investigación de un gran interés. Existen multitud de trabajos relacionados con visión artificial en humanoides (González, 2009), (Blower, 2001), muchos de ellos combinando obtención de distancias, detección de objetos y teleoperación (Pierro et al., 2009), (Herrero-Pérez, 2009), (Gil, 2004). La fabricación de cámaras y equipos de procesamiento más baratos y más avanzados y el desarrollo y distribución de mejores herramientas y algoritmos de visión han permitido el crecimiento de este campo. Las librerías de OpenCV (Open Source Computer Vision Library) son unas de estas herramientas que ha jugado un papel esencial en el crecimiento de la visión por computador permitiendo realizar trabajos en visión de forma más productiva.

OpenCV ayuda a estudiantes y profesionales a implementar de manera rápida y eficiente proyectos de investigación en visión artificial. OpenCV proporciona una infraestructura de algoritmos maduros de visión por ordenador y machine learning del cual partir en el desarrollo de nuevas implementaciones de soluciones basadas en visión por computador. (Bradski and Kaehler, 2008)

En este trabajo se desarrollo un algoritmo capaz de detectar objetos, calcular su posición, e identificar, mediante su color y su forma, el objeto detectado. Este sistema de visión se desarrollo para proveer de visión por computador al robot humanoide HOAP-3.

El documento está estructurado de la forma siguiente, en el apartado 2 se habla de la plataforma utilizada, en el apartado 3 y 4 se explican los algoritmos desarrollados para la detección de objetos y cálculo de la distancia, en el apartado 4 se presenta el sistema implementado en el robot humanoide, en el apartado 6 se muestran los resultados experimentales y por último se presentan las conclusiones

2 Herramientas y plataforma

Las librerías OpenCV fueron desarrolladas por Intel en 1999 y sus siglas provienen del acrónimo inglés Open source Computer Vision Library. Se trata de un conjunto de librerías creadas para el tratamiento de imágenes, proporcionando un alto nivel de funciones para su procesado. Destinadas

principalmente a aplicaciones en tiempo real, como pueden ser: tracking de objetos, reconocimiento de caras, reconstrucción 3D, estabilización de imagen, etcétera. Al tratarse de librerías libres permiten trabajar con ellas desde distintas plataformas y Sistemas Operativos.

La plataforma utilizada para probar los algoritmos ha sido el robot humanoide HOAP-3. Se trata de un robot de fabricación japonesa con 21 grados de libertad, 60 cm de altura y un peso de 9kg. Este robot posee un sistemas de visión estereoscópico que le permite detectar profundidad, además de otros sensores como FSR, infrarrojos, acelerómetros y giróscopos. Esta plataforma está especialmente diseñada para implementar algoritmos de control y es usada en la Universidad Carlos III como plataforma de prueba para Proyectos de Fin de Carrera y Tesis de Máster y Doctorales.

3 Detección de objetos

Para la detección de objetos tendremos que definir bien las características que definen de manera unívoca a nuestro objeto de interés (Ballard and Brown, 1982). A la que vamos a dar más importancia será al color.

Aquí surge una pregunta importante: ¿qué es color? La definición de la Real Academia Española (RAE) dice: "*Sensación producida por los rayos luminosos que impresionan los órganos visuales y que depende de la longitud de onda"*. Sin embargo no debemos olvidar que estamos trabajando con imágenes digitales y lo que nuestros ojos no perciben una cámara sí. Por esa misma razón trabajamos con el modelo HSV cuyas siglas provienen del acrónimo inglés Hue, Saturation, Value o lo que es lo mismo Tono, Saturación y Valor (Figura 1). Define un modelo de color en términos de sus componentes constituyentes en coordenadas cilíndricas (Zelansky y Fisher, 2001):

- Tonalidad: tipo de color representado como un grado de un ángulo (0º-360º).
- Saturación: distancia al eje de brillo negro-blanco. También llamado pureza.
- Valor: brillo del color. Representa la altura en el eje blanco-negro.

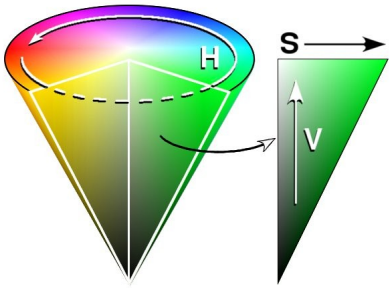

Figura 1: Representación del modelo HSV

El otro modelo de color más extendido es el modelo RGB del acrónimo inglés Red, Green y Blue (Figura 2). La desventaja de este modelo es que presenta menos información. Sin embargo se puede pasar de un modelo a otro aplicando simples fórmulas:

$$H_1 = cos^{-1}\left(\frac{\frac{1}{2}*((R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \quad (1)$$

$$S = \frac{M-m}{M} \quad (2)$$

$$V = \frac{M}{255} \quad (3)$$

Siendo

$H = H_1$ si $B \leq G$

$H = 360° - H_1$, si $B > G$

M = max (R,G,B)

m = min(R,G,B)

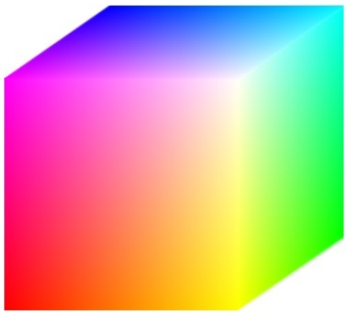

Figura 2: Representación del modelo RGB

Existen muchos más espacios de color en función de cómo modifiquen su representatividad, algunos a destacar son YUV, HSL, etcétera.

Una vez definida la característica principal el problema resulta en cómo distinguir el objeto deseado del resto del entorno. Para ello se utiliza la segmentación que entre otras funciones permite la extracción del fondo. Normalmente se han hecho dos aproximaciones básicas para resolver el problema de la segmentación. Una de ellas basada en la detección de bordes-contornos y otra basada en la detección de regiones.

En los métodos basados en bordes se detectan discontinuidades locales y que más tarde se tratan de unir formando un borde que delimita unos objetos de otros. Y en los métodos basados en regiones, se busca determinar las áreas de una imagen que tienen propiedades homogéneas delimitando los objetos con este borde. Los dos métodos, por lo tanto, son complementarios y dependen en gran medida de la aplicación para la que vayan a ser destinados.

La segmentación empleada en este artículo se basa en la detección de regiones (Somolinos, 2002). Para poder distinguir las regiones se deben aplicar umbralizaciones. Tanto para RGB como para HSV la umbralización se basa en la detección de umbrales del mapa de componentes RGB o HSV. La umbralización de imágenes de objetos en entornos desestructurados muchas veces es compleja por la falta de conocimiento a priori del número de objetos a detectar o cambios en el entorno como por ejemplo la influencia de elementos no deseados como sombras, brillos, la compleji-

dad de los colores, tamaño, posibilidad de solapamientos entre objetos, variaciones en el fondo, etcétera. Esto nos lleva a dos posibles problemas:

- Sub-segmentación: elección de un número bajo de umbrales con lo que se obtienen menos regiones de las deseadas.
- Sobre-segmentación: elección de un número alto por lo que se detectan más regiones de las deseadas.

Para evitar dichos contratiempos se ha decidido realizar una segmentación multinivel en el espacio HSV. La segmentación de una imagen HSV se realiza segmentando cada uno de las componentes de las que consta la imagen: tonalidad, saturación y valor para analizarlos y trabajar con ellos. Otra razón para escoger el espacio HSV frente al RGB es porque puedo diferenciar de manera más precisa mi objeto de interés del resto de la imagen

La detección de objetos es una utilidad muy importante en la robótica. Por ese motivo se ha intentado aumentar la funcionalidad del algoritmo mediante una función que nos haga posible diferenciar un objeto rectangular o cuadrado de uno circular. Esto permite que el humanoide no solo detecte el objeto sino que pueda coger uno u otro con criterio. Para lograr diferenciar la forma del objeto se hace una aproximación a partir del área y perímetro del objeto. Si podemos aproximar el área del objeto a la de un círculo con un margen de error de un 10% entonces el objeto es redondo, sino suponemos que el objeto es rectangular o cuadrado.

4 Estimación de la distancia

La estimación de la distancia se puede realizar de dos formas distintas:

- Partiendo de una sola imagen.
- Con un par de imágenes.

Si partimos de imágenes monoculares hay que tener en cuenta aspectos como: iluminación, sombras, tamaño del objeto, distancia focal de la cámara, etcétera. Este método no es aconsejable ya que la distancia obtenida es una aproximación, lo que se traduce en falta de fiabilidad y precisión.

Partiendo de un par de imágenes es mucho más sencillo y preciso. Esto se debe a que tenemos dos imágenes distintas del mismo entorno. Estas imágenes presentan una característica clave para el cálculo de distancias: la disparidad. La disparidad es la diferencia de la dirección horizontal en ambas imágenes. De una forma más intuitiva, la disparidad es la diferencia de posición de un punto fijo en ambas imágenes. Dicho punto puede ser el centroide de un objeto y la disparidad la distancia de dicho centro al margen izquierdo de las dos imágenes. A parte del centroide se deben conocer dos datos más:

- La distancia entre las cámaras (DIO: distancia intraocular).
- Los ángulos de visión horizontal y vertical de las cámaras.

La DIO suele variar entre 45-75 mm, en el caso del humanoide empleado se sitúa en 60mm. Cuanto mayor sea la DIO, mayor disparidad tendremos y por lo tanto, mejor se podrá calcular la distancia al objeto.

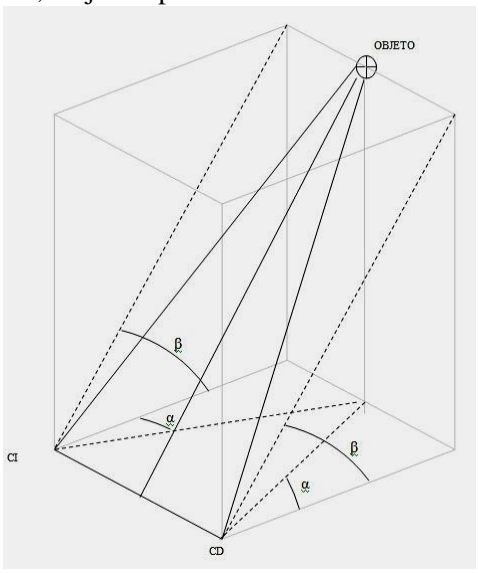

Figura 3: Representación sistema cámaras-objeto.

En la Figura 3 se puede observar un modelo simplificado del sistema formado por las cámaras del robot y el objeto. Basándose en dicho esquema se han aplicado las ecuaciones y fundamentos teóricos para el cálculo del objeto al centro de la línea formada por las cámaras (DIO).

A continuación se explica el desarrollo matemático para el cálculo. Sólo se detallan los cálculos con una cámara pero hay que hacerlos para ambas:

$$\alpha = \left(\left(\frac{l}{2}\right) - xc1\right) * \left(\frac{H_{ANGLE}}{l}\right) \tag{4}$$

donde: l: anchura de la imagen en píxeles.
xc1: posición x del centroide del objeto.
H_ANGLE: ángulo de visión horizontal de la cámara.

$$\beta = \left(\left(\frac{h}{2}\right) - yc1\right) * \left(\frac{V_{ANGLE}}{h}\right) \tag{5}$$

donde: h: altura de la imagen en píxeles.
yc1: posición y del centroide del objeto.
V_ANGLE: ángulo de visión vertical de la cámara.

$$\frac{a}{\sin(\alpha)} = \frac{b}{\sin(\beta)} = \frac{c}{\sin(\gamma)} \tag{6}$$

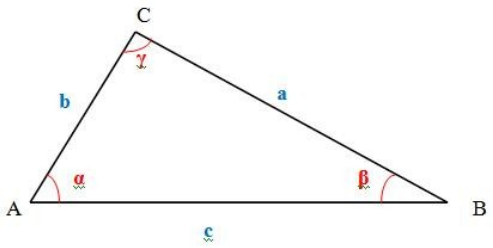

Figura 4: Notación habitual del triángulo.

El teorema del seno (Ecuación 6) establece que: la relación existente entre un lado dividido entre el ángulo opuesto se mantiene para los tres lados y sus respectivos ángulos (Figura 4).

Combinando las ecuaciones (4,5,6) y realizando proyecciones sobre los planos horizontal, vertical y lateral se obtiene la distancia desde el centro de la DIO al centro del objeto (Figura 5 y Figura 6).

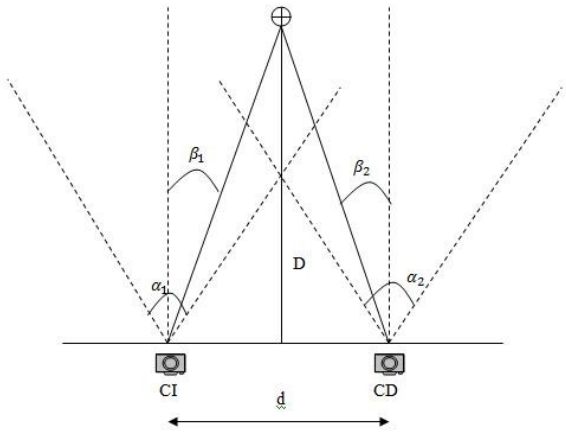

Figura 5: Proyección horizontal del sistema cámaras-objeto.

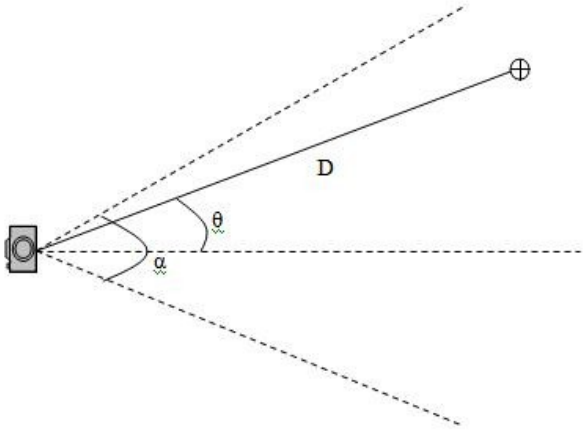

Figura 6: Proyección lateral del sistema cámaras-objeto

5 Sistema de Visión del Robot Humanoide para la Detección e Identificación de objetos

5.1 Comunicación con el robot

El sistema de visión del robot humanoide se implemento con una arquitectura sencilla de cliente servidor. En la Figura 7 se muestra la arquitectura del sistema implementado.

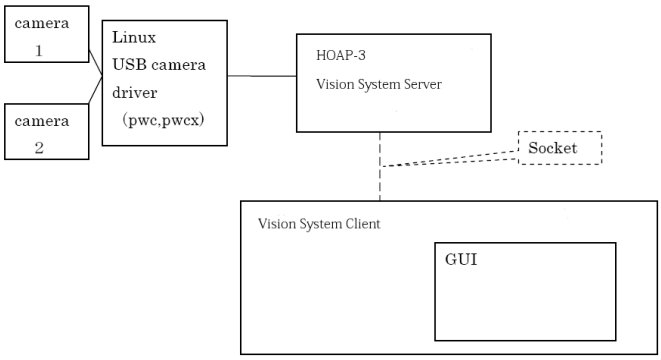

Figura 7: Arquitectura del sistema

El robot HOAP-3 cuenta con un sistema de dos cámaras que nos permite conseguir una visión estero. El servidor de visión, PC interno del HOAP-3, se encarga de capturar y envía la captura de imagen de las cámaras como un array de bytes (char). El cliente recibe la información de la imagen del robot en el formato YUV240p y luego convierte de este formato a la estructura IplImage de openCV, requerida por los algoritmos de detección e identificación descritos en la próxima sección.

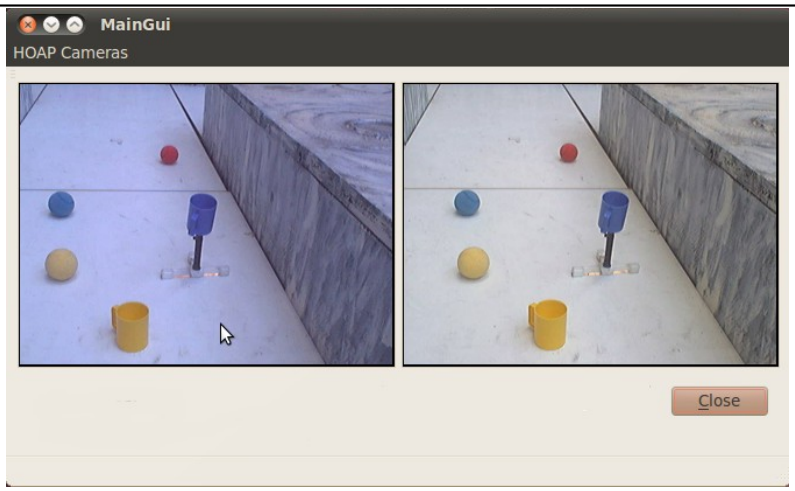

Figura 8: Interfaz de visualización de las imágenes

La imagen procesada se muestra al usuario a través de una interfaz creada con las librerías QT. Las librerías QT y OpenCV se pueden integrar muy fácilmente y se puede cambiar de un formato de imagen a otro mediante las funciones QImage2IplImage y IplImage2QImage.

La Figura 8 muestra la visualización de las imágenes vistas por el robot HAOP-3 en la GUI implementada. La interfaz proporcionara al usuario información sobre los objetos detectados y la distancia al que se encuentra cada objeto.

5.2 Algoritmo de detección

A continuación se muestran y explican los diagramas de flujo de los principales algoritmos desarrollados para la detección de objetos y cálculo de distancias (Figura 9).

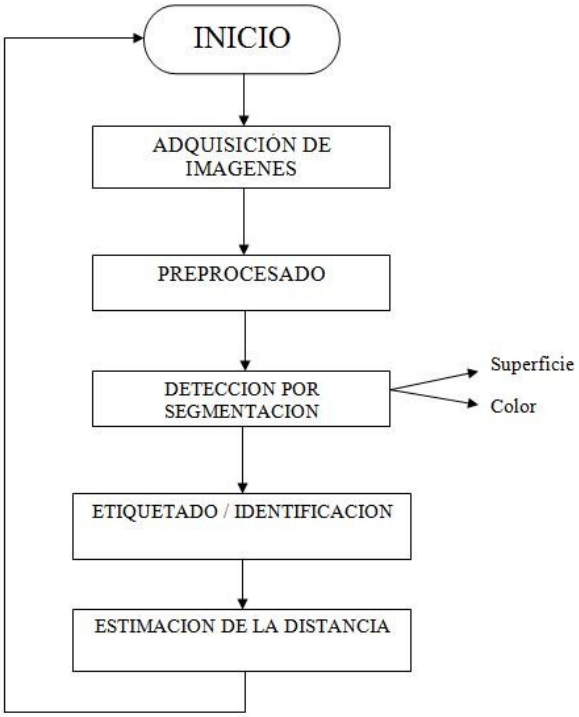

Figura 9: Diagrama de flujo del algoritmo principal

En la adquisición de datos se toma una imagen del vídeo suministrado por las cámaras del humanoide.

En la etapa de pre-procesado se someterá a las imágenes a transformaciones en los espacios de color para poder usarlas posteriormente como máscaras en la etapa de segmentación. En dicha etapa, se realizan funciones lógicas entre las imágenes originales y las máscaras obtenidas anteriormente, para poder separar el fondo de nuestro objeto de interés. Es en este momento cuando se debe determinar si en la imagen obtenida previamente aparece el objeto que deseamos detectar. Para ello se tiene en cuenta el área del objeto en píxeles y las componentes HSV del color.

Posteriormente, en la etapa de Etiquetado / Identificación, se procede a dibujar el contorno del objeto para que cumpla las características descritas anteriormente.

Por último se calcula la distancia a partir de la posición del centro del objeto detectado. Todo este proceso se vuelve a repetir mientras se esté grabando vídeo.

La realización de este algoritmo se ha visto beneficiada por el uso de las librerías OpenCV. Las funciones explicadas a continuación son las básicas de OpenCV así como algunas de las más importantes para el funcionamiento del algoritmo:

-cvLoadImage(char filename, int iscolor);

Esta función me carga la imagen definida por el nombre introducido en filename, en la variable que le especifique. El parámetro iscolor me permite cargarla en el formato de color original o en blanco y negro. Ejemplo:

IplImage* color = cvLoadImage("fotot.jpg", CV_LOAD_IMAGE_COLOR);

Esta función me permite cargar imágenes desde diversos formatos: BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF y TIF.

- cvDrawContours(CvArr *image, CvSeq* contour, double external_color, double hole_color, int max_level, int thickness=1, int connectivity=8);

  Función que permite dibujar contornos en image, dichos contornos vienen definidos por la secuencia contour.Ejemplo:

  cvDrawContours (output, result, CV_RGB(255,0,0), CV_RGB(0,0,255), 0, 2, 8);

- cvThreshold (const CvArr* src, CvArr* dst, double threshold, double maxValue, int thresholdType);

  Función que permite aplicar unos umbrales a una imagen original src y guardarla como imagen dst. Con un umbral mínimo threshold y un valor máximo maxValue, según un tipo de umbralización.

```
cvThreshold (canales[canal], mascara2, umbral2, 255,
CV_THRESH_BINARY_INV );
```

5.2.1 Detección por segmentación

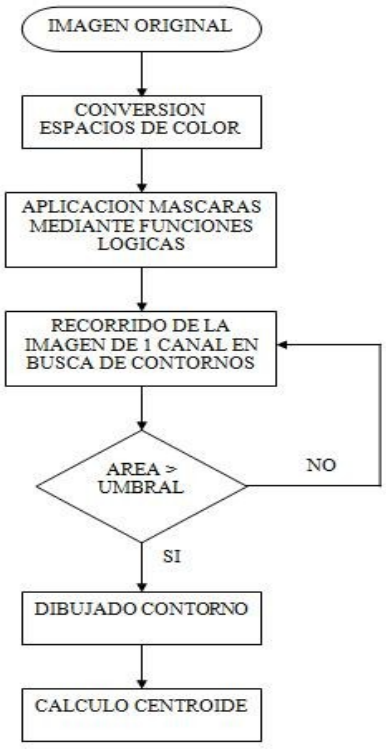

Figura 10: Diagrama de flujo del algoritmo de detección

El proceso seguido en la segmentación ha sido el siguiente (Figura 10):

1. Transformo la imagen original de RGB a HSV.
   Esta operación se realiza de forma inmediata con una función de las librerías OpenCV: cvCvtColor(input,output,conversión_type). Sin embargo lo que se desea no es una sola imagen en el espacio HSV sino tres imágenes en cada una de las cuales tendré uno de los canales del modelo HSV, esto se consigue con la función cvSplit(input,output0,output1,output2,output3) siendo outpu3 cero en este caso.
2. Filtro la imagen original con la máscara de matiz.
3. Filtro la imagen original con la máscara de saturación.

4. Filtro la imagen original con la máscara de valor.
   Como se ha explicado anteriormente para poder detectar un objeto es necesario realizar una segmentación, que en este caso se hará mediante segmentación por regiones. En los pasos 2-4 lo que se hace es crear unas máscaras, mediante umbralización basada en color, de cada uno de los canales del modelo HSV (Figura 11).

Figura 11: Máscaras de los tres canales del modelo HSV

Cuando se trabaja en el sistema HSV, el color de cada uno de los píxeles de una imagen está definido por tres componentes: Tono, Saturación y Brillo o Valor. Para identificar los píxeles de un determinado color se comprueba que los niveles de sus tres componentes corresponden a los del color buscado. Por ello una umbralización por color tendrá la forma de:

$$g(x,y) = \begin{cases} 1\ si & \begin{array}{l} H_a \leq f_H(x,y) \leq H_b \\ S_a \leq f_S(x,y) \leq S_b \\ V_a \leq f_V(x,y) \leq V_b \end{array} \\ 0 & en\ cualquier\ caso \end{cases}$$

Donde $f_H(x,y)$, $f_S(x,y)$, $f_V(x,y)$ son, respectivamente, las funciones que dan los niveles de tono, saturación y brillo de cada uno de los píxeles de la imagen.
A partir de cada una de las máscaras obtenidas se aplican funciones lógicas. La función lógica empleada es una AND. OpenCV incluye una función para implementar esta operación: cvAnd(input1, input2, output). Al aplicar esta

función lógica obtengo una imagen filtrada como puede observarse en la Figura 12.

Figura 12: Imagen original filtrada

5. Recorro la imagen obtenida en busca de contornos que cumplan las especificaciones.
6. Dibujar contorno.
7. Cálculo centro del objeto para su uso en el cálculo de distancias

5.2.2 Cálculo de la distancia

En la Figura 13 puede observarse el procedimiento empleado para el cálculo de la distancia del objeto respecto al humanoide.

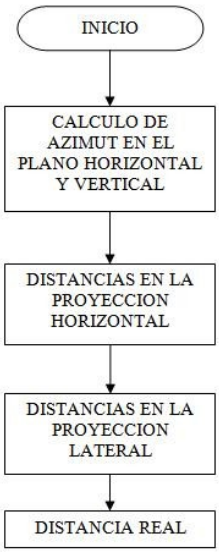

Figura 13: Diagrama de flujo para el algoritmo de cálculo de la distancia

Aplicando las ecuaciones mencionadas anteriormente (1,2,3) se procede al cálculo de la distancia del centro de la línea que une las cámaras al objeto detectado. Con cada paso dado estamos obteniendo la distancia al objeto proyectada en un plano. El resultado obtenido al finalizar este algoritmo es el observado en la Figura 14.

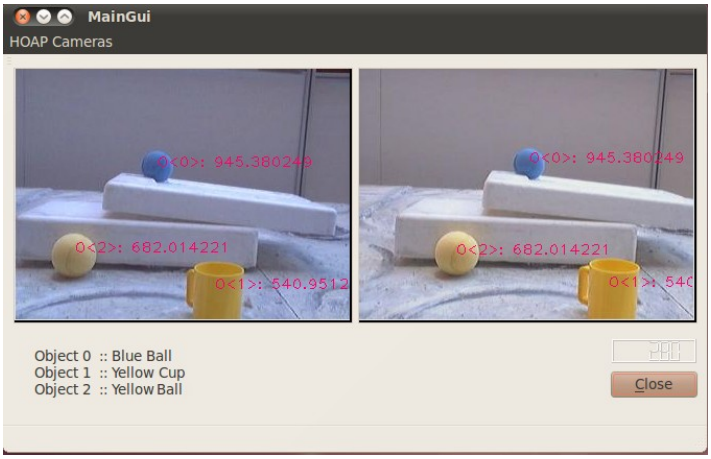

Figura 14: Resultado del algoritmo completo

7 Conclusiones

En este artículo se ha presentado un sistema de identificación de objetos y obtención de las distancias entre el objeto y el robot. Este sistema es capaz de detectar objetos con varias formas y colores y sacar la distancia de todos ellos a la vez. Esto es un primer paso para conseguir dotar al robot humanoide HOAP-3 de mayor autonomía y capacidad de manipulación. Se han implementado los algoritmos mediante las librerías OpenCV, las cuales permiten gran versatilidad a la hora de trabajar con sistemas de visión. También se ha diseñado una interfaz mediante las librerías Qt para mostrar los resultados al usuario.

Referencias

D. H. Ballard, C. M. Brown. "Computer Vision". Ed. Prentice-Hall. 1982.

A. Blower. "Development of a visión System for a Humanoid Robot". Bachelor Thesis. University of Queensland. October 2001.

D. Herrero-Pérez, P. Pierro, C. Balaguer. "Object Recognition and Guided-Grasping for Humanoid Robot in Unstructures Scenario". Universidad Carlos III Madrid, 2009.

P. Gil, F. Torres y F. G.Ortiz "Detección de objetos por segmentación multinivel combinada de espacios de color". XXV Jornadas de Automática, Ciudad Real. 2004.

T. González. "Artificial Vision in the Nao Humanoid Robot". Máster Thesis. Universidad Rovira i Virgill. Septiembre 2009.

P.Pierro; D.Hernandez; M.G.Fierro; C.Balaguer. Humanoid teleoperation system for space environments. 14th International Conference on Advanced Robotics (ICAR '09), pp.1-6. Munich. Germany. Jun, 2009.

J. A. Somolinos. "Avances en robótica y visión por computador". Ediciones de la Universidad de Castilla la Mancha. 2002.

P. Zelanski, M. P. Fisher "Color". Editorial Tursen S.A. Edición Hermann Blume. 2001.