Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Interfaz gestual para el control de un robot humanoide con una cámara RGB-d

PROYECTO FIN DE CARRERA

Autor: Luis Parrilla Bel

Director: Ana Cristina Murillo Arnal

Ingeniería en Informática
Curso 2012-2013

Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

Noviembre de 2012

Interfaz gestual para el control de un robot humanoide mediante cámaras RGB-d

RESUMEN

La divulgación científica, cuya finalidad es hacer accesible la ciencia al público en general, es una tarea que cada vez está tomando mayor relevancia. Este proyecto trata el estudio de dos campos de investigación atractivos para el público en general y con muchos resultados interesantes en los últimos años, como son la robótica y la visión por computador. En particular, el proyecto se centra en el uso del dispositivo Kinect como interfaz para la interacción con el robot RoboNova-1, y en el diseño de una actividad basada en ello para utilizar como taller de divulgación científica.

La aparición del sensor Kinect ha supuesto una revolución en el campo de la visión por computador y de la robótica, por las posibilidades que ofrece el mapa de profundidad capturado (imagen aumentada con información de distancia a la cámara del elemento representado en cada píxel) y por su bajo coste. Gracias al mapa de profundidad que Kinect aporta en tiempo real, se facilita mucho el trabajo de reconocimiento y segmentación de objetos en 3D.

Esta característica facilita la segmentación de las distintas partes de una persona enfrente de la cámara, convirtiendo la kinect en un dispositivo muy apto para crear interfaces con gestos. El objetivo general del proyecto es la implementación de un interfaz gestual, mediante cámaras RGB-d, con un robot humanoide y diseñar con ello una actividad orientada a la divulgación de la robótica y la inteligencia artificial para niños y jóvenes.

En el proceso de desarrollo podemos distinguir dos partes importantes, que son el reconocimiento de los gestos y la comunicación con el robot humanoide. Para el reconocimiento de los gestos se usa la imagen con información 3D captada por la cámara RGB-d para identificar y segmentar donde esta el usuario de la aplicación. La figura de la persona nos sirve para crear una estructura de esqueleto que captará los movimientos de la persona y en los brazos se seleccionan las zonas que determinarán las manos en 3D. Posteriormente este subconjunto de puntos en 3D se proyecta en blanco sobre una imagen negra, obteniendo la mano en 2D. Tras el filtrado del ruido, estas imágenes serán utilizadas por los métodos de clasificación para determinar a que gesto pertenece cada captura.

El robot RoboNova-1 dispone de un software propio para su programación desde Windows. Esta aplicación nos permitirá introducir programas y secuencias de movimiento en la memoria interna del robot utilizando el programa RoboBasic. La comunicación con el robot se realizará mediante un módulo diseñado para otro robot, por lo que hubo que adaptarlo a las especificaciones de protocolos de comunicación utilizados por el chip MR-C3024 del RoboNova-1.

La aplicación desarrollada se divide en dos bloques. El primero contiene las aplicaciones que nos servirán para capturar los datos necesarios en el entrenamiento de los métodos de clasificación de gestos. En el segundo bloque encontramos la aplicación principal que hará uso del interfaz gestual y con la que se evaluará el rendimiento de los métodos de clasificación.

Una vez desarrollado el interfaz se preparó un taller que se llevó a cabo durante la celebración de la "V Semana de la ingeniería y arquitectura". En él participaron estudiantes desde 3º de ESO hasta 2º de Bachiller. Durante una demostración se les explicó el funcionamiento del sistema y posteriormente fue probado por los estudiantes. Se obtuvo un buen resultado en el funcionamiento y buena aceptación entre los asistentes.

Índice general

Capítulo 1

Introducción

1. Motivación

La divulgación científica, cuya finalidad es hacer accesible la ciencia al público en general, es una tarea que cada vez está tomando mayor relevancia. Una manera de hacer llegar estos temas y trabajos al mayor número posible de personas, y de manera más comprensible y amena, es presentándolo mediante talleres interactivos y con herramientas atractivas para ellos.

Este proyecto trata el estudio y divulgación de dos campos de investigación atractivos para el público en general y con muchos resultados interesantes en los últimos años, como son la robótica y la visión por computador. Recientemente, el campo de los videojuegos a acercado al público en general muchos sensores que tradicionalmente se utilizan en robótica, y ahora son más comunes, y también más asequibles debido a la mayor producción. Un ejemplo son las cámaras RGB-d (cámaras de visión y profundidad), como la utilizada en el sensor Kinect.

Este proyecto se centra en el uso del dispositivo Kinect como interfaz para interacción con un robot (ver Figura 1.1). El sensor Kinect[1] es un producto de Microsoft comercializado para su uso con la Xbox 360 (Anexo B). Los sensores RGB-d están generando grandes avances en el campo del reconocimiento visual automático [1, 7], ofreciendo un complemento muy valioso a los algoritmos de visión por computador que utilizan solamente sensores de visión [2, 10]. Por ejemplo, en el caso concreto que se estudia en este proyecto, reconocimiento de gestos, permiten una captura y segmentación de las zonas de interés de la imagen (las manos) mucho más eficaz que si usáramos solamente una cámara convencional [3, 5, 9].

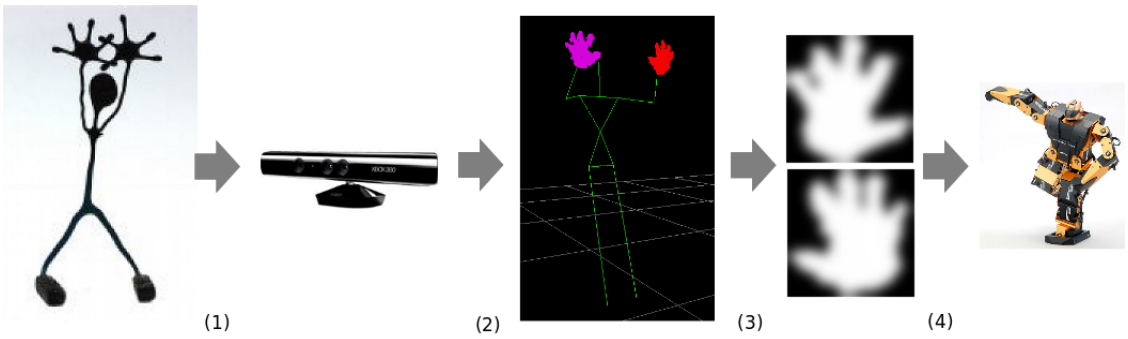


Figura 1.1: (1) El sensor Kinect captura al usuario. (2) Se reconoce el cuerpo del usuario y sus manos. (3) Las manos son analizadas. (4) Se transmite la orden al robot.

[1] http://www.xbox.com/es-ES/kinect

Aprovechando por tanto las ventajas de este sensor para crear interfaces con gestos [4, 5, 6, 8], se propone diseñar un interfaz para comunicar con un robot y diseñar una actividad orientada a la divulgación de la robótica y la inteligencia artificial para estudiantes tanto de primaria, como de secundaria y bachiller. La idea de este proyecto es formar parte de una serie de actividades de divulgación realizadas periódicamente por el grupo de Robótica, Percepción y Tiempo Real del DIIS.

2. Objetivos y alcance del proyecto

El objetivo general del proyecto es la implementación de un interfaz gestual mediante cámaras RGB-d con un robot humanoide y diseñar con ello una actividad orientada a la divulgación de robótica e inteligencia artificial para niños y jóvenes. Las tareas más en detalle realizadas en este proyecto han sido las siguientes:

- Estudio, instalación y familiarización con el entorno ROS [12], incluyendo drivers Openni para comunicación con sensores Kinect y librerías PCL [11] y OpenCv para facilitar las operaciones con imágenes 3D y 2D respectivamente.
- Puesta en marcha de la plataforma robótica a utilizar. Estudio de las especificaciones, utilidades y posibilidades de comunicación remota del robot humanoide disponible para el proyecto, el robot Robonova-1.
- Estudio de algoritmos sencillos para detección de gestos mediante sensores RGB-d.
- Implementación o adaptación/mejora de los métodos estudiados en la literatura, para que funcionen como interfaz del sensor Kinect con el robot disponible (a través del sistema ROS instalado en un ordenador portátil, donde se realizarán todos los cálculos relacionados con el reconocimiento de gestos).
- Diseño de una actividad/taller que haga uso de el interfaz/sistema de comunicación diseñado.
- Realización de pruebas y documentación de las actividades a realizar, para permitir el uso de los resultados de este proyecto en actividades próximas de divulgación científica.

3. Herramientas y entorno

La ejecución del interfaz se ha realizado sobre ROS [12], un pseudo sistema operativo utilizado para gestión de plataformas robóticas y sensores relacionados. ROS provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Se ha utilizado la versión "ROS Electric"[2], la calificada como estable. El proyecto se ha desarrollado bajo un sistema operativo Ubuntu 10.04 "Lucid Lynx"[3], siendo éste sobre el que "Willow Garage", principal desarrollador del sistema ROS, nos ofrece soporte.

El manejo del sensor Kinect se ha realizado mediante el driver Openni [22] y con las librerías PCL[11] y OpenCv [23] tenemos lo necesario para el tratamiento de imágenes en 3D y 2D. Estas librerías las podemos encontrar dentro de "ros-pkg", un repositorio de paquetes aportados por la contribución de usuarios. Para la comunicación con el robot RoboNova-1 se utiliza "Robobuilder-ROS-package"[4]. Un script de Python para integrar las plataformas Robobuilder en ROS.

Las modificaciones en el código del RoboNova-1 se han realizado en el capítulo 2 con el software que el propio robot nos proporcionaba. Este software es un entorno de programación de Windows basado en el lenguaje RoboBasic, especializado y orientado a robots.

[2]Robot Operating System: http://www.ros.org/wiki/
[3]http://www.ubuntu.com/
[4]https://code.google.com/p/robobuilder-ros-pkg/#Robobuilder_ROS_package_written_in_Python

4. Estructura del documento

En esta memoria se describe el proceso para la creación de un interfaz gestual mediante cámaras RGB-d con un robot humanoide. En el capítulo 2 se explicará el método utilizado para el reconocimiento y segmentación de las manos, así como el tratamiento de las imágenes y los métodos de clasificación probados. En el capítulo 3 se explicará el sistema de comunicación con el robot RoboNova-1 y su programación. En el capítulo 4 se detallará la aplicación realizada. En el capítulo 5, las conclusiones recopilan el resultado del proyecto, líneas de trabajos futuros y una valoración personal del proyecto fin de carrera. Finalmente encontraremos varios anexos con información de interés sobre el sistema, dispositivos utilizados y los resultados completos de las pruebas realizadas.

5. Planificación

En el diagrama de Gantt que se muestra en la Figura 1.2 queda reflejada la gestión del tiempo utilizada para la realización de este proyecto. En el se detallan las tareas que se han realizado y el tiempo invertido en ellas. Entre las tareas realizadas hay tareas de estudio, documentación, realización de tutoriales, prueba de programas de reconocimiento, implementación del interfaz y presentación de talleres durante la celebración de la "V semana de la ingeniería y arquitectura"[5].

[5] http://www.semanaingenieriayarquitectura.com/

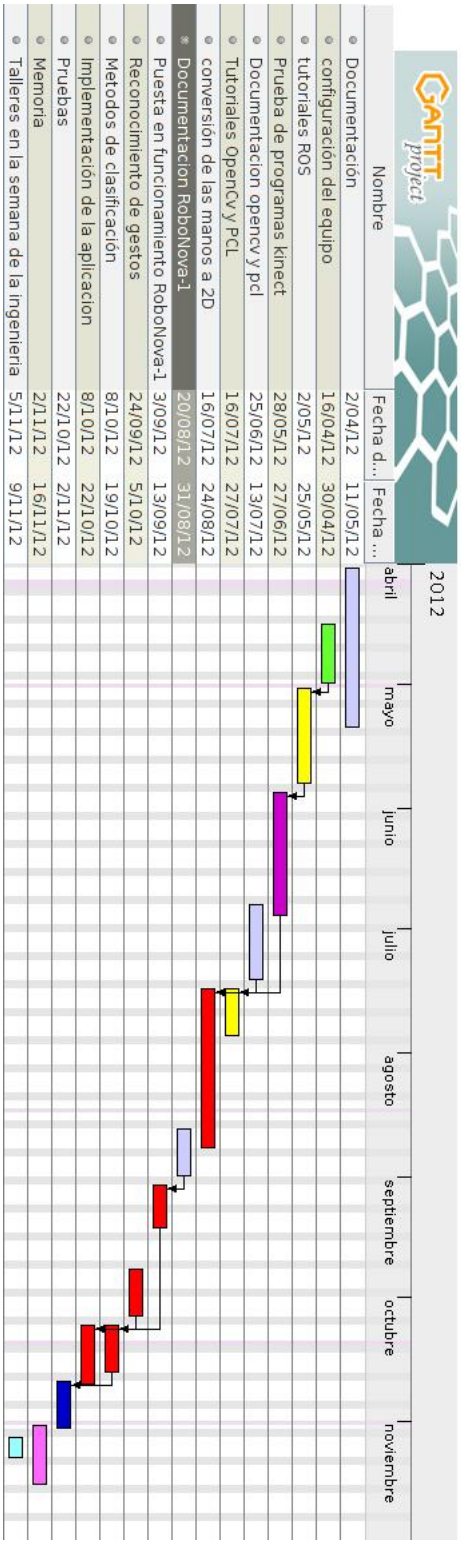


Figura 1.2: Diagrama de Gantt, que resume la distribución temporal de tareas del proyecto.

Capítulo 2

Reconocimiento de Gestos

Tal y como se ha descrito en los objetivos, se requería implementar un interfaz gestual mediante cámaras RGB-d con un robot humanoide.

En este proceso, podemos diferenciar dos partes importantes, el reconocimiento de los gestos y la comunicación con el robot humanoide, que se describirán en éste y el siguiente capítulo respectivamente.

En el proceso para reconocer un gesto, debemos pasar por varias etapas como muestra el diagrama de la Figura 2.1. En las diferentes secciones de este capítulo se explicará los diferentes pasos seguidos en el proceso de reconocimiento. En primer lugar deberemos reconocer y segmentar las manos (Sección 1). A continuación, utilizar filtros que procesen la imagen de las manos para facilitar el reconocimiento (Sección 2). Finalmente se realizará la clasificación del gesto capturado (Sección 3).

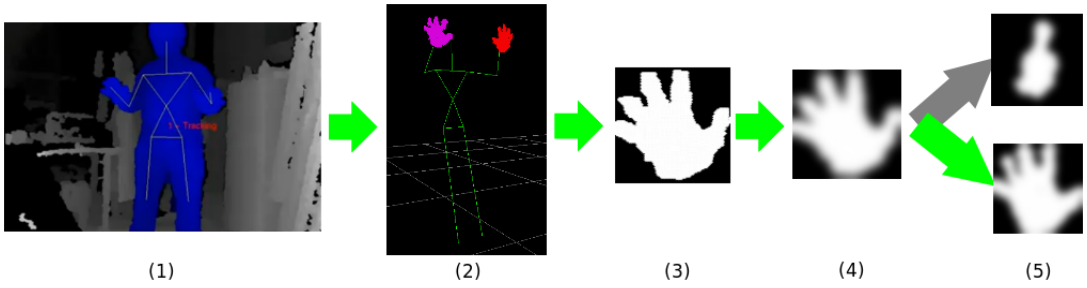


Figura 2.1: Proceso de reconocimiento de gestos. (1)(2)(3) Reconocimiento y segmentación de las manos. (4) Tratamiento de la imagen. (5) Clasificación del gesto .

1. Reconocimiento y segmentación de las manos en datos 3D

Los sensores RGB-d nos proporcionan una secuencia de imágenes en el formato típico RGB sincronizado con la correspondiente imagen de profundidad en tiempo real, como se ve en la Figura 2.2. Esta segunda parte, las imágenes de profundidad, son la parte que se ha utilizado para el reconocimiento y segmentación de las manos en el desarrollo del interfaz.

En el proceso de reconocimiento de gestos, debemos pasar por varias fases hasta llegar a poder comparar diferentes muestras de manos, como se ha podido ver en la Figura 2.1.

Figura 2.2: Imagenes capturadas por el sensor kinect. En la derecha podemos ver la imagen de la cámara RGB y en la izquierda la imagen captada por el sensor de profundidad.

Detección de la persona y las manos desde los datos 3D

Como se puede observar en el diagrama anterior (Figura 2.1), la primera parte de este proceso es reconocer a la persona que va a utilizar el interfaz. Para ello, existen diferentes programas, implementados con métodos diferentes, para el reconocimiento del cuerpo usando cámaras RGB-d. Después de realizar un estudio entre varios, se decidió utilizar el siguiente por ser el más sencillo de integrar con el entorno de programación requerido.

Se ha partido del programa "Hand Detection"[1] del paquete *KinectDemos* de las librerías de ROS. Este programa se encarga de detectar, dentro de la imagen de profundidad, la figura de personas. Esta selección la realiza mediante la búsqueda de zonas de la imagen con variaciones de posición o de profundidad en la secuencia de vídeo (Figura 2.3). Con este proceso se seleccionan cuerpos móviles y se excluyen objetos que puedan aparecer dentro del área de visión de la Kinect.

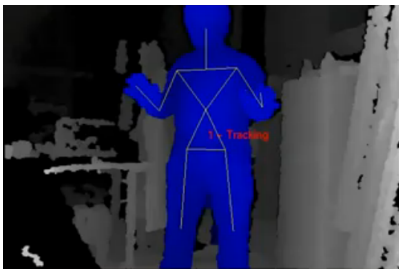


Figura 2.3: Reconocimiento del usuario.

Con la persona en cuestión seleccionada, procede a buscar una silueta determinada de ésta, con las manos a ambos lados de la cabeza, la cual debes imitar para que sea capaz de crear un "esqueleto" (Figura 2.4) que determinará la posición del cuerpo y los movimientos que se realicen. A partir de la estructura del esqueleto se localiza la posición final del brazo, las muñecas, y busca la nube de puntos situados alrededor de esa posición creando las manos.

Finalmente, las nubes de puntos son "publicadas" en ROS como "/hand0_fullcloud" mano izquierda y "/hand1_fullcloud" mano derecha para su posterior utilización, como podemos ver en la Figura 2.5. más adelante en el anexo A se detalla en que consiste el proceso de publicación de información en ROS, básicamente es una manera para compartir información entre procesos ejecutados en paralelo.

[1] http://www.ros.org/wiki/mit-ros-pkg/KinectDemos

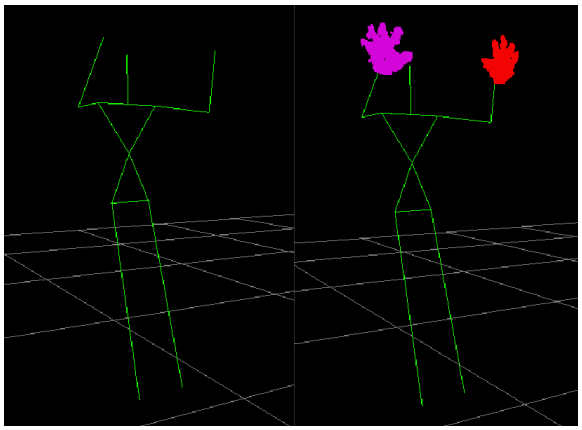


Figura 2.4: En primer lugar se realiza el reconocimiento del cuerpo y creación de un esqueleto, como podemos ver en la imagen de la izquierda. Posteriormente se seleccionan los puntos que componen la mano, como muestra la imagen de la derecha.

Segmentación de las manos y conversión a 2D

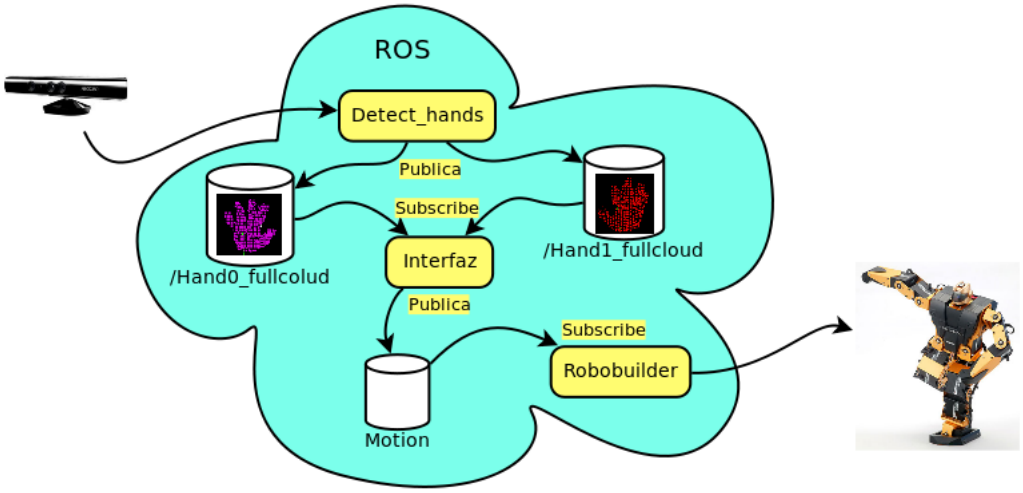


Figura 2.5: Diagrama de comunicación de procesos en ROS. Vemos como "interfaz" se subscribe a */Hand0_fullcloud* y */Hand1_fullcloud* que publica "Detect_Hands". Interfaz convierte las manos a 2D, clasifica las imágenes y posteriormente realiza el mismo proceso para comunicar al robot la acción a realizar.

Cuando la aplicación principal comience su ejecución se subscribirá a los "/hand0_fullcloud" y "/hand1_fullcloud" que *detect_hands* estará publicando, y así comenzará recibir las nubes de puntos encontradas correspondientes a ambas manos. En este momento poseemos unas nubes de puntos en 3D que representan las manos (Figura 2.6), pero es difícil y costoso el trabajar con los datos en este formato, por lo que el siguiente paso será la transformación de éstas a imágenes en 2D con las que realizaremos la clasificación de las diferentes posturas de la mano. Para el análisis de las imágenes se utilizan las bibliotecas de OpenCv.

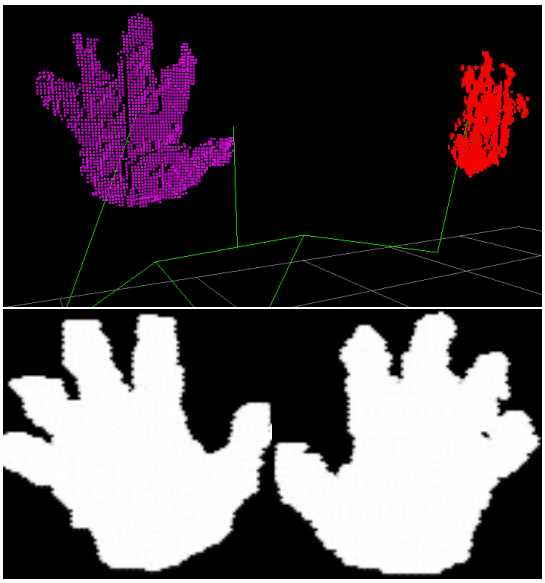


Figura 2.6: La imagen superior muestra las nubes de puntos en 3D de las manos. La inferior, imágenes en 2D de las nubes de puntos.

Debido a la versión de la librería *perception_pcl-1.0.2*, librería de PCL soportada por ROS Electric, no disponemos de muchas funciones incorporadas en versiones posteriores como conversión de tipos o el guardar una nube de puntos como imagen. Por otro lado, ROS dispone también de librerías para la conversión de tipos, como *cv_bridge*, pero en el caso de estas nubes de puntos no ordenadas el resultado no era el deseado. Por lo tanto, se ha implementado un algoritmo propio que convierte la nube de puntos en el tipo de datos requerido en opencv, en particular crea imágenes (2D) proyectando cada punto de la nube en blanco sobre una imagen negra.

2. Preprocesado de las imágenes de las manos 2D

Al convertir las manos a imágenes en 2D, el primer problema encontrado es que dependiendo de la posición de la mano, p.e. palma abierta o palma cerrada, tienen diferente tamaño, pero también lo tienen dos manos a diferente distancia del sensor o las manos de un adulto y un niño. Eso creaba un problema a la hora del reconocimiento del gesto. Una posible solución era insertar muestras de diferentes tamaños de cada uno de los gestos, pero para llevar a cabo esta solución era necesario insertar una cantidad de muestras muy superior y la necesidad de realizar las pruebas con más gente, con diferentes tamaños de mano, y a diferentes distancias de la Kinect. Finalmente, la solución a este problema fue el redimensionado de las capturas hasta igualarlas a las de las muestras (Figura 2.7). Para ello, al arranque de la aplicación, se toma el tamaño actual de la palma de la mano y se usa con las siguientes imágenes como referencia.

La creación de las imágenes se realiza dibujando, en color blanco, cada uno de los puntos que componen las manos en 3D, sobre un fondo negro. Para la elección del tamaño a utilizar tuve que tener en cuenta varios parámetros, de los cuales se seleccionaron los de mejores resultados tras las pruebas.

Figura 2.7: Imagen palma (izquierda). Imagen puño desproporcionada (centro). Imagen puño manteniendo las proporciones (derecha).

Tamaño de la imagen y de la proyección de los puntos

Al haber una cantidad variable de puntos en la nube de cada mano, hubo que realizar pruebas con el tamaño de la imagen y el tamaño del punto a utilizar. En la Figura 2.8 podemos ver varios ejemplos.

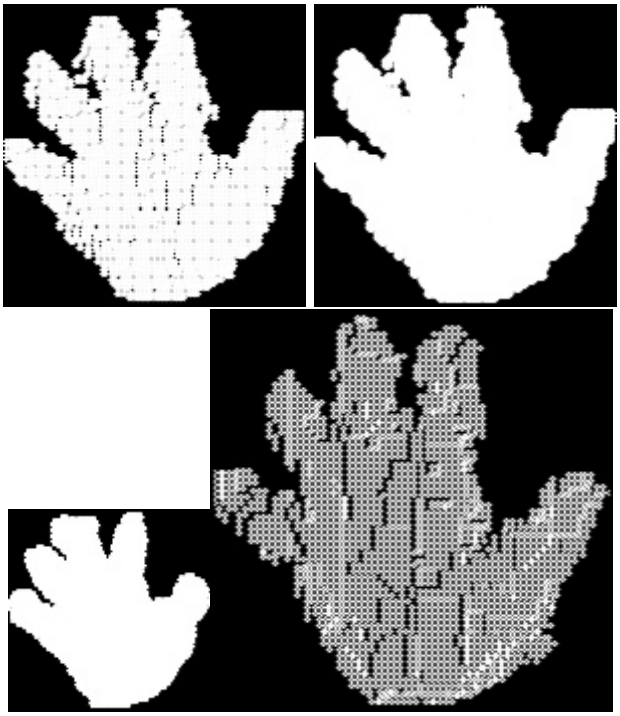


Figura 2.8: muestras de diferentes configuraciones de la imagen

Tras las pruebas se pudo comprobar que se conseguían rendimientos similares con imágenes de diferentes tamaños, utilizando el grosor del punto adecuado. Tras finalizar las pruebas y analizar los resultados se seleccionó las siguientes configuraciones:

- Tamaño en píxeles del lado de la imagen de la mano = 100
- Propiedades del punto:
  - Radio del círculo en píxeles = 0

- Grosor de la línea en píxeles = 2

Este resultado coincide con la imagen inferior izquierda de la Figura 2.8.

Filtrado del ruido

Tras una preselección de varias combinaciones de tamaño de imagen y de punto, se repitieron las pruebas incorporando pasos de filtrado a la imagen para suavizarla y evitar ruido, con la intención de que mejorase el resultado del reconocimiento como se muestra en muchos ejemplos de la literatura [13]. Las siguientes pruebas se realizaron con filtro Gaussiano [14] y filtro de mediana [15].

El índice utilizado para los filtros indicará el tamaño del recuadro a tener en cuenta, centrado en el píxel que se va a modificar, utilizado para obtener el nuevo valor.

Filtro Gaussiano

Un desenfoque Gaussiano es el resultado de una imagen difuminada por una función Gaussiana. Es un efecto ampliamente utilizado para reducir el ruido de la imagen y reducir los detalles. El valor de cada píxel sera el resultado del producto de la función Gaussiana aplicada en cada una de las dimensiones [16]:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

En un filtro Gaussiano, el índice utilizado nos determinara el valor de σ, la desviación de la distribución Gaussiana (Figura 2.9). El valor resultado será una media ponderada, realizada mediante círculos concéntricos, de los valores del recuadro seleccionado.

Figura 2.9: Imagen original (izquierda). Filtrado con Gaussian Blur con indice 7 (centro). Filtrado con Gaussian Blur con ndice 23 (derecha).

Filtro de mediana

El filtro de mediana es una técnica de filtrado digital no lineal, a menudo utilizado para eliminar el ruido. Tal reducción de ruido es un típico paso de tratamiento previo para mejorar los resultados del procesamiento posterior, como la detección de bordes en una imagen. El Filtrado de mediana es ampliamente utilizado en el procesamiento digital de la imagen, ya que, en determinadas condiciones, conserva los bordes mientras elimina ruido de la imagen [17].

Este filtro reemplaza cada píxel con la mediana de sus píxeles vecinos localizados en el cuadrado que rodea al píxel en cuestión, como muestra el ejemplo de la Figura 2.10.

En este caso, el índice utilizado para los filtros indicará el tamaño del recuadro a tener en cuenta, centrado en el píxel que se va a modificar, utilizado para obtener el nuevo valor.

3	35	12
6	25	45
15	17	22

3	6	12	15	17	22	25	35	45

Figura 2.10: Ejemplo para la búsqueda del valor que se asignará (en verde, abajo) al píxel central (en verde, arriba) utilizando un filtro de mediana.

Figura 2.11: Imagen original (izquierda). Filtrado con Median Blur con indice 7 (centro). Filtrado con Median Blur con ndice 23 (derecha).

3. Algoritmos de clasificación de las imágenes 2D para reconocer las distintas posturas de la mano

Una vez obtenidas las imagenes binarizadas de la mano detectada, se realiza el último paso de decidir a que gesto se asemeja más. En una primera versión se realizó la comparación respecto a unas muestras de cada gesto comparando la cantidad de píxeles que se diferenciaban respecto a cada muestra. El resultado fue bueno, pero con un número reducido de gestos y muestras por gesto. Este sistema implica el comparar la nueva imagen capturada con todas las muestras y eso es un proceso costoso si utilizamos un número elevado de muestras que nos permitan el reconocimiento variando el ángulo o posición de la mano. Por el contrario, si se utilizan pocas muestras por gesto, supone el tener que hacer el gesto con una posición muy semejante a la de las muestras, ya que en caso contrario el resultado tenia una alta probabilidad de ser erróneo.

Para una segunda versión, se procedió a analizar los resultados obtenidos con otros dos métodos de clasificación más complejos y eficientes. De los diferentes métodos de clasificación descritos en la literatura [19] se decidió utilizar uno de los más populares de cada gran familia de clasificadores: un método de tipo generativo, *K Nearest Neighbors,* y otro de tipo discriminativo, *Support Vector Machine.*

K Nearest Neighbors

El método de clasificación basado en *K Nearest Neighbors*[18], KNN, es un método de clasificación "lazy learning" de tipo generativo, que estima, a partir de la información proporcionada por un conjunto de vectores de entrenamiento, el valor de la función de densidad de probabilidad de que un elemento pertenezca a una clase determinada. Para el reconocimiento de patrones, el algoritmo KNN es un método de clasificación de objetos basado en los ejemplos del entrenamiento más cercanos en el espacio de los elementos.

El entrenamiento se realiza con vectores en un espacio multidimensional, asignándole a cada uno una etiqueta de la clase a la que pertenece. Para el proceso de clasificación se seleccionan los

k ejemplos más cercanos y el nuevo vector es clasificado como la clase que más se repita entre los ejemplos seleccionados. Podemos ver un ejemplo en la Figura 2.12.

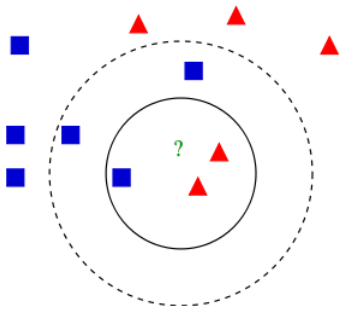


Figura 2.12: Ejemplo del algoritmo KNN. Deseamos clasificar el interrogante verde como triángulo o cuadrado. Con k = 3, éste es clasificado como la clase triángulo, ya que hay un cuadrado y 2 triángulos, dentro del círculo que los contiene. Si k = 5, éste es clasificado como la clase cuadrado, ya que hay 2 triángulos y 3 cuadrados, dentro del círculo externo.

Generalmente se utiliza la distancia euclídea para determinar quienes son los vecinos más cercanos:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p}(x_{ir} - x_{jr})^2} \tag{2.2}$$

Support Vector Machines

El método de clasificación basado en *Support Vector Machines* [20], al contrario del anterior, es un método de tipo discriminativo, que modela la dependencia de una variable no observada en una variable observada. Dentro de un marco estadístico, esto se hace mediante el modelado de la distribución de probabilidad condicional[21] $P\,(y|x)$, que se puede utilizar para predecir y a partir de x. Éste método realiza la clasificación mediante la construcción de uno o varios hiperplanos N-dimensionales que separan los datos de manera óptima en dos categorías. En la Figura 2.13 vemos un ejemplo en 2D .

4. Pruebas y análisis de resultados

Para buscar los parámetros en los que se obtuviesen los mejores resultados se ha realizado una batería de pruebas evaluando los diferentes parámetros y almacenando su mejor o peor comportamiento en la clasificación deseada. Para la evaluación se han utilizado las medidas estándar utilizadas en problemas de clasificación, *precision-recall*[2], que resumen lo obtenido en cada prueba para su posterior comparación.

Las pruebas se han realizado mediante secuencias grabadas anteriormente para poder realizar una evaluación exhaustiva sobre los mismos datos con distintas opciones. La grabación de estos datos se realizaba siguiendo unos patrones para obtener una cantidad lo más similar posible de muestras por cada gesto, y así realizar comparaciones más equitativas. Mediante el programa *analiza* se obtienen unas tablas con los resultados de las clasificaciones de ambas manos para los dos sistemas de clasificación a comparar.

[2] http://en.wikipedia.org/wiki/Precision_and_recall

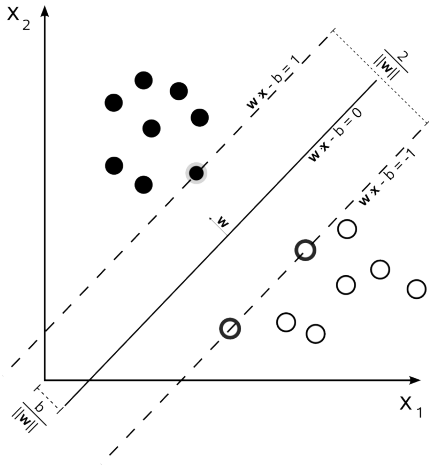


Figura 2.13: Ejemplo en 2D de la separación entre dos clases con el máximo margen posible.

El programa *analiza* se invoca de la siguiente manera:

> rosrun interfaz analiza *nombre_calibracion resolucion radio_punto grosor_linea tipo_linea filtro indice_filtro video*

- *nombre_calibracion*: nombre del directorio donde se encuentra las imágenes utilizadas para la calibración
- *resolucion*: dimensión del lado de la imagen en píxeles
- *radio_punto*: radio del círculo utilizado para cada punto al proyectarlos en la imagen
- *grosor_linea:* grosor de la línea utilizada en los círculos de cada línea
- *tipo_linea*: tipo de línea utilizado para el dibujo de cada punto
  - 1= 8-connected line
  - 2 = 4-connected line
  - 3 = CV_AA: antialiased line
- *filtro*:
  - 0 = sin filtro
  - 2 = Gaussian Blur
  - 3 = Median Blur
- *indice_filtro:* índice para el filtro que se utilizará. Este valor debe ser número impar.
- *video*: Nombre del vídeo del cual se desean obtener los resultados del análisis

Para cada uno de los experimentos se muestran dos tablas con los resultados obtenidos con cada uno de los métodos de clasificación, como podemos observar en la Tabla 2.1. En cada una de ellas la parte izquierda y derecha de la tabla corresponde a los resultados de reconocimiento de la mano izquierda y derecha respectivamente.

Cada tabla muestra el resultado de reconocer los gestos 1 y 2 (ejemplos en Figura 2.14), y el 0 se utiliza como caso "nulo" cuando el clasificador decide que la imagen no corresponde con

ninguno de los gestos. Las columnas muestran cuantos tests correspondientes a esa clase han sido clasificados como cada una de las clases posibles. Es decir, el resultado óptimo seria obteniendo una diagonal, con todos los tests en las casillas 1-1 y 2-2. El criterio que buscamos es el clasificador con mayor porcentaje de aciertos obtenido.

Observando los resultados obtenidos de las pruebas, los cuales podemos encontrar en el anexo D, se observa que dependiendo del método de búsqueda que se utilice los resultados varían. En ambos casos coincide la configuración de la imagen que da resultados mejores, es decir, el tamaño de la imagen, radio y grosor del círculo. Sin embargo, la mejor elección para el filtro utilizado varía según el método de clasificación. Si realizamos la búsqueda por **k nearest neighbor** el mejor resultado lo obtenemos si utilizamos un **filtro Gaussiano con índice 19,** como vemos en la tabla 2.1, y, en cambio, si utilizamos **Support Vector Machine** el mejor resultado lo obtenemos con un **filtro Median Blur con índice 5**, que se encuentran en la tabla 2.2.

Los resultados han sido los siguientes:

- Para la clasificación con KNN:
  - rosrun interfaz analiza dos2-100-0-2-2-19 100 0 2 3 2 19 luisPalmOk2
    - tamaño en píxeles del lado = 100
    - radio del círculo = 0
    - grosor de la línea = 2
    - tipo de línea = CV_AA
    - filtro = Gaussian Blur
    - índice para el filtro = 19

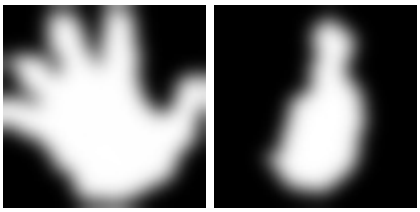


Figura 2.14: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

- Para la clasificación con SVM:
  - rosrun interfaz analiza dos2-100-0-2-3-5 100 0 2 3 3 5 luisPalmOk2
    - tamaño en píxeles del lado = 100
    - radio del círculo = 0
    - grosor de la línea = 2
    - tipo de línea = CV_AA
    - filtro = Median Blur
    - índice para el filtro = 5

Tras realizar las pruebas y seleccionar el formato de imagen con el que se obtiene mejores resultados, se ha realizado un nuevo análisis tras la inserción de más muestras para la calibración del sistema. En las pruebas anteriores se analizaba partiendo de 50-60 muestras por gesto. En esta ocasión se realizará el análisis con aproximadamente 300 muestras por gesto. Con los datos de la

LEFT	0	1	2	RIGHT	0	1	2
0	0	3	0	0	0	3	0
1	0	54	10	1	0	52	11
2	0	12	59	2	0	15	57

Precisión obtenida mediante KNN = 0.78

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	0	3
1	0	59	5	1	0	13	50
2	0	56	15	2	0	56	16

Precisión obtenida mediante SVM = 0.37

Tabla 2.1: Resultados de las pruebas

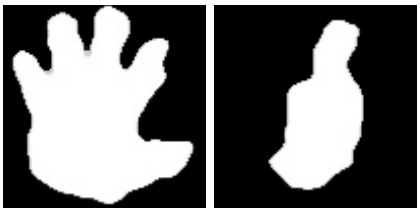


Figura 2.15: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	2	1
1	1	40	22	1	0	28	35
2	0	10	62	2	0	6	66

Precisión obtenida mediante KNN = 0.71

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	0	3
1	1	53	9	1	0	39	24
2	0	11	61	2	9	46	26

Precisión obtenida mediante SVM = 0.65

Tabla 2.2: Resultados de las pruebas utilizando un filtro de mediana

tabla 2.3 podemos observar que en el caso de KNN el resultado no varía, en cambio en el de SVM mejora notablemente.

Estos resultados muestran que si se disponen de pocas muestras para el entrenamiento el uso del método de clasificación “k-nearest neighbor” es más eficiente que con “Suport Vector Machine”. En cambio, si aumentamos el número de muestras el resultado es similar con ambos métodos.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	0	4	0
1	0	48	8	1	0	46	9
2	0	8	34	2	0	8	34

Precisión obtenida mediante KNN = 0.79

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	0	4	0
1	0	47	9	1	0	46	9
2	0	6	36	2	0	7	35

Precisión obtenida mediante SVM = 0.8

Tabla 2.3: Resultados de las pruebas utilizando un filtro de mediana

Capítulo 3

Comunicación con robot humanoide RoboNova-1

Como ya se ha indicado anteriormente, para el desarrollo de este proyecto se ha utilizado el robot RoboNova-1(figura 3.1). El RoboNova-1 es un robot humanoide que lleva incorporados 16 servos digitales. A pesar de que también cuenta con un sensor de infrarrojos para su control desde un mando a distancia, la comunicación se realizará por medio de una conexión línea serie con el ordenador. Si nos dirigimos al anexo C encontraremos información más detallada sobre el robot RoboNova-1.

Figura 3.1: Robot humanoide RoboNova-1

1. Robobuilder-ros-pkg

Para el control del robot he utilizado el paquete Robobuilder-ros-pkg incorporado en el repositorio de ROS por RoboSavvy[24]. En él encontramos un controlador para la plataforma "Robobuilder", la cual lleva un sistema de comunicación, por línea serie, con algunas similitudes al que utiliza el "RoboNova-1". El controlador es un script Python[1]. Éste realiza una subscripción a "robobuilder_motion", por donde se le mandarán las ordenes de las acciones a realizar entre las que tiene programadas en el programa interno.

[1]Se necesita una versión de Python igual o superior a la 2.7

Debido a las diferencias entre los protocolos de comunicación entre estos dos modelos, hubo que realizar algunas modificaciones en este script:

- Modificar la velocidad de conexión a 9600 bits/seg.
- Desactivar el modo de control directo.
- Realizar comprobación del puerto al que esta conectado el robot.

Robobuilder utiliza una conexión a 11520 bits/seg y para el control directo utiliza otros protocolos de comunicación diferentes a los de RoboNova-1. Por otro lado, el script está preparado para realizar la conexión en el puerto USB0. Pero utilizándolo de manera simultanea con el sensor Kinect, éste puede ser reconocido fácilmente en algún otro puerto y se debe comprobar el puerto al que se encuentra conectado nuestro robot.

Para la comunicación con la memoria interna del robot, donde se encuentran las secuencias de movimiento preprogramados, éste script es válido para este proyecto. En el modo de control directo, comunicación inmediata con los motores del robot, debemos realizarla de manera independiente a este script siguiendo las especificaciones de protocolo del chip "Hitec MR-C3024"[2][3].

Para realizar una acción de la memoria interna del robot deberemos incluir el fichero *robobuilder/Motion.h* en el que tenemos definida la variable que utiliza la comunicación con el robot y publicarlo ROS para que el robot reciba la instrucción:

```
ros::Publisher motion_;
motion_ = n.advertise<robobuilder::Motion> ("robobuilder_motion", 1);
robobuilder::Motion m;
m.motion = (int8_t)7;
motion_.publish(m);
```

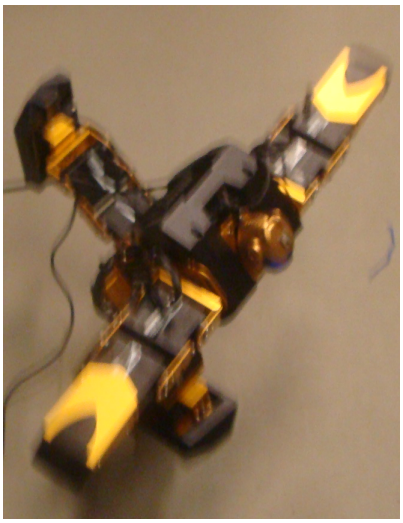


Figura 3.2: Robot humanoide RoboNova-1 ejecutando la acción 7 programada en su memoria interna.

[2]http://www.ceautomatica.es/sites/default/files/upload/10/CEABOT/recursos/C3024_Serial_Protocol.pdf
[3]http://www.ceautomatica.es/sites/default/files/upload/10/CEABOT/recursos/controller%20serial%20interface.pdf

En el caso de control directo nos comunicaremos mandando las instrucciones directamente al robot por medio de línea serie:

```
std::vector<unsigned char> command;
command.push_back(0xE6);
command.push_back(0x07);
command.push_back(0xB4);
sendCommand(command);
```

El valor "0xE6"indica la instrucción de cambiar la posición de un servo, el byte segundo indica el servo a controlar y por ultimo la posición a la que debe trasladarse. En este caso el resultado sería el que muestra la Figura 3.3.

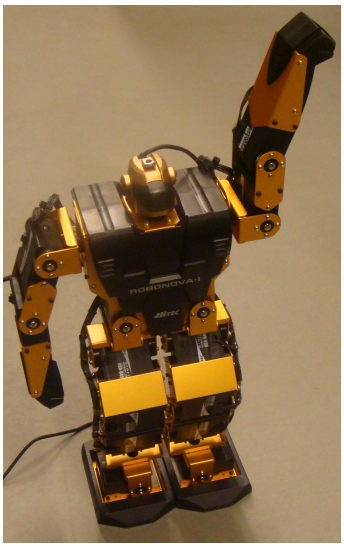


Figura 3.3: Robot humanoide RoboNova-1

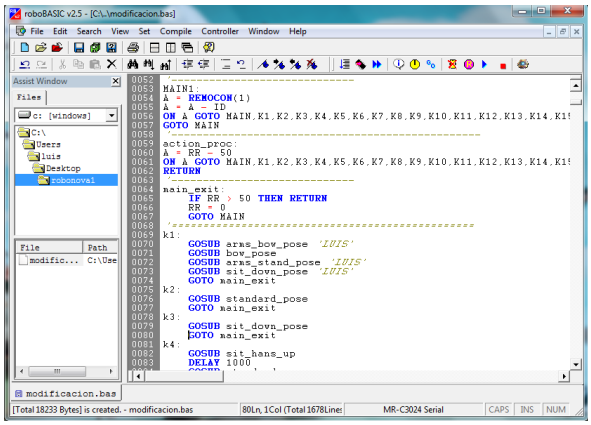
2. Programación del RoboNova-1

El robot posee un software propio basado en el lenguaje RoboBasic, que como su nombre indica es un lenguaje del tipo Basic pero especializado y orientado a robots(Figura 3.4).

Con la siguiente rutina como ejemplo, el robot procedería a sentarse y, tras esperar un segundo, volvería a levantarse a la velocidad indicada, en este caso 8:

```
SPEED 8
GOSUB sit_down_pose
DELAY 1000
GOSUB standard_pose
GOTO main_exit
```

sit_down_pose y *standard_pose* son rutinas en las que se especifican los diferentes movimientos que deben realizar cada uno de los motores del robot:




```
'-------------------------------
MAIN1:
A = REMOCON(1)
A = A - ID
ON A GOTO MAIN,K1,K2,K3,K4,K5,K6,K7,K8,K9,K10,K11,K12,K13,K14,K15
GOTO MAIN
'-------------------------------------------------
action_proc:
A = RR - 50
ON A GOTO MAIN,K1,K2,K3,K4,K5,K6,K7,K8,K9,K10,K11,K12,K13,K14,K15
RETURN
'-------------------------------
main_exit:
    IF RR > 50 THEN RETURN
    RR = 0
    GOTO MAIN
'-------------------------------------------
k1:
    GOSUB arms_bow_pose 'LUIS'
    GOSUB bow_pose
    GOSUB arms_stand_pose 'LUIS'
    GOSUB sit_down_pose 'LUIS'
    GOTO main_exit
k2:
    GOSUB standard_pose
    GOTO main_exit
k3:
    GOSUB sit_down_pose
    GOTO main_exit
k4:
    GOSUB sit_hans_up
    DELAY 1000
```

Figura 3.4: Entorno de programación roboBASIC

standard_pose:

MOVE G6A, 100, 76, 145, 93, 100, 100
MOVE G6D, 100, 76, 145, 93, 100, 100
WAIT
MOVE G6B, 100, 30, 80, 100, 100, 100
MOVE G6C, 100, 30, 80, 100, 100, 100
WAIT
RETURN

Mediante el parámetro "G6?" indicamos la extremidad en la que debemos realizar el movimiento, y los valores siguientes son la posición de cada uno de los servos. *G6A* y *G6D* se refieren a las piernas del robot y *G6B* y *G6C* a los brazos.

Éste software nos permite el controlar en tiempo real los motores del robot y capturar la posición de los motores para crear nuevos movimientos sin la necesidad de calcular los valores necesarios para cada articulación (Figura 3.5).

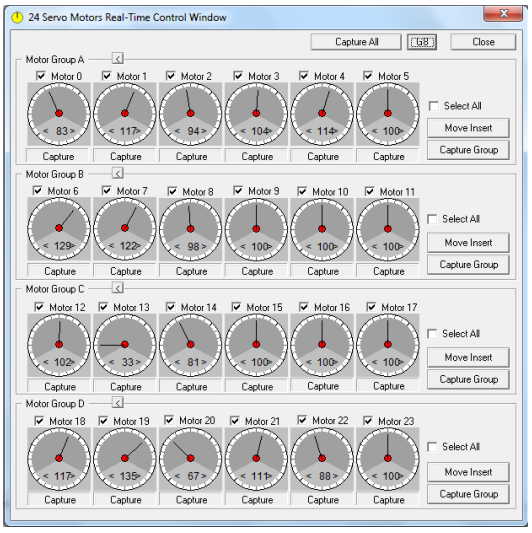


Figura 3.5: Panel para el control de los servos del robot en tiempo real.

Capítulo 4

Aplicación realizada

Como se ha ido describiendo a lo largo de la memoria, se ha creado un interfaz gestual para el control de un robot humanoide.

Para ello se han implementado varias aplicaciones que realizarán las diferentes tareas necesarias para el funcionamiento del interfaz. La aplicación se divide en dos bloques, tal como muestra la Figura 4.1. El primero nos servirá para capturar las muestras necesarias en el entrenamiento de los métodos de clasificación. El segundo hará uso del sistema de reconocimiento y del interfaz gestual.

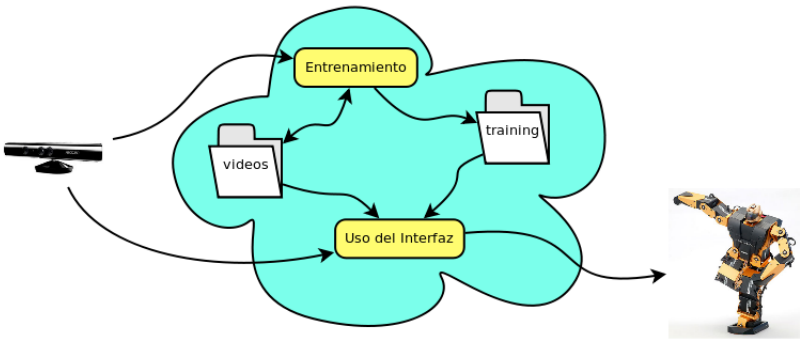


Figura 4.1: Esquema de uso de recursos de la aplicación

1. Entrenamiento del sistema

Para el funcionamiento del interfaz gestual es necesario disponer de los archivos de muestras con los que entrenar los métodos de clasificación. Si no se disponen de ellos, se podrán crear con las aplicaciones implementadas para tal fin (Figura 4.2). El entrenamiento lo podemos realizar de dos maneras diferentes, situándonos ante la Kinect directamente o mediante vídeos realizados para tal fin.

Grabar

El módulo "Grabar" realizará grabaciones que posteriormente nos servirán para la captura de muestras necesarias para el entrenamiento de los sistemas de búsqueda y para realizar los análisis de resultados para las pruebas.

Esta aplicación tiene dos modos de funcionamiento según los parámetros añadidos al lanzarla.

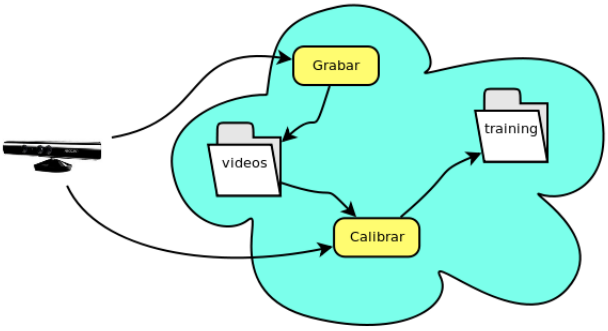


Figura 4.2: Esquema de uso de recursos del bloque de entrenamiento del sistema

```
rosrun interfaz grabar 'nombag' [label]
```

En el caso de no introducir "label" la aplicación lanzará mensajes por pantalla con el gesto que se debe realizar. Mientras tanto realizará publicaciones de las etiquetas correspondientes a lo indicado en el mensaje de pantalla.

Si por el contrario introducimos un valor en el campo "label", el gesto a realizar será el introducido en "label" y los mensajes de pantalla nos indicarán el tiempo restante hasta concluir la grabación y publicará constantemente el valor de "label".

Al comenzar su ejecución y seleccionado el modo de funcionamiento, la aplicación solicitará al usuario que suba y baje los brazos para comenzar. En este momento comenzará la grabación del vídeo "nombag", que almacenará las publicaciones de "manos" y "esqueleto" realizadas por "Detect_hands"[1] y las publicaciones "label" que ella misma realiza.

Hasta que se solicite la introducción de posturas de la mano determinadas, el valor publicado del topic "label" para la captura del vídeo será 0.

Calibrar

Calibrar tomará las muestras necesarias para el entrenamiento de los sistemas de clasificación. Para lanzar calibrar utilizaremos el siguiente comando:

```
rosrun interfaz calibrar 'calibracion' 'resolucion' 'radio' 'grosor' 'tipo' 'filtro' 'indiceFiltro' ['nombag']
```

- *calibracion*: nombre del directorio donde se encuentra las imágenes utilizadas para la calibración
- *resolucion*: dimensión del lado de la imagen en píxeles
- *radio*: radio del círculo utilizado para cada punto al proyectarlos en la imagen
- *grosor:* grosor de la línea utilizada en los círculos de cada línea
- *tipo*: tipo de línea utilizado para el dibujo de cada punto
  - 1= 8-connected line
  - 2 = 4-connected line
  - 3 = CV_AA: antialiased line

[1] demo de los repositorios ROS-pkg

- *filtro*:
  - 0 = sin filtro
  - 2 = Gaussian Blur
  - 3 = Median Blur
- *indiceFiltro:* índice para el filtro que se utilizará. Este valor debe ser número impar.
- nombag: Nombre del vídeo del cual se desean obtener los resultados del análisis

Esta aplicación utilizará 'nomCalibracion' para crear, en el caso de que no exista, las carpetas usadas para almacenar las muestras en función de la mano y posteriormente en función del gesto, tal y como indica la figura 4.3.

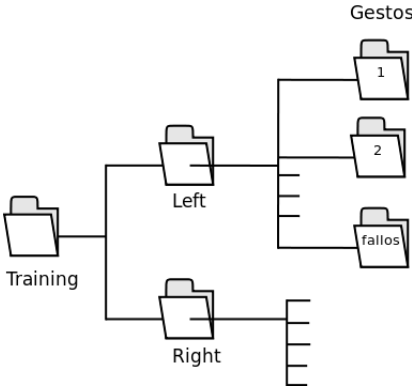


Figura 4.3: Esquema del almacenamiento de los datos de entrenamiento

Si hemos rellenado el campo 'nombag' en este momento comenzará la reproducción de la grabación la cual utilizará para su ejecución. Si no hemos introducido nada, la aplicación procederá como "grabar" indicando por pantalla el gesto a realizar para la captura de muestras. El programa insertará las muestras en la carpeta correspondiente según la mano y el gesto, indicado por el mensaje en pantalla o por las publicaciones de "label" proporcionadas por el vídeo.

2. Uso del clasificador y el interfaz gestual.

Con muestras almacenadas en los ficheros de entrenamiento ya se puede hacer uso del sistema. En este bloque disponemos de otras dos aplicaciones, como podemos ver en la Figura 4.4. La primera de ellas, *analizar*, nos servirá para analizar la eficiencia del sistema con cada uno de los métodos de clasificación utilizados, KNN y SVM. La segunda, *interfaz*, hará uso del interfaz gestual para interaccionar con el robot humanoide.

Analizar

Con esta aplicación obtendremos un fichero con los resultados de las clasificaciones obtenidas mediante los métodos KNN y SVM. En ellas se mostrará, para cada gesto los aciertos y fallos obtenidos, enumerando los errores en función del gesto a reconocer y el reconocido. La tabla tendrá una fila y una columna por cada gesto. Cada fila corresponde a cada uno de los gestos. Cada columna nos indicará la cantidad de reconocimientos obtenidos por cada gesto.

Para lanzar esta aplicación lanzaremos el siguiente comando:

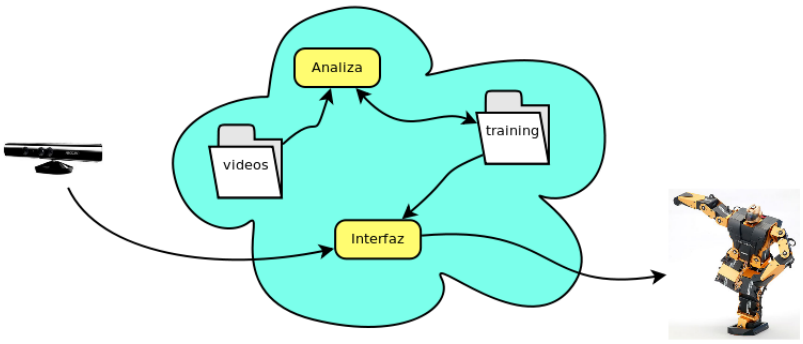


Figura 4.4: Esquema de uso de recursos del interfaz

```
rosrun interfaz analiza 'calibracion' 'resolucion' 'radio' 'grosor' 'tipo' 'filtro' 'indice_filtro' 'nombag'
```

- *calibracion*: nombre del directorio donde se encuentra las imágenes utilizadas para la calibración
- *resolucion*: dimensión del lado de la imagen en píxeles
- *radio*: radio del círculo utilizado para cada punto al proyectarlos en la imagen
- *grosor:* grosor de la línea utilizada en los círculos de cada línea
- *tipo*: tipo de línea utilizado para el dibujo de cada punto
  - 1= 8-connected line
  - 2 = 4-connected line
  - 3 = CV_AA: antialiased line
- *filtro*:
  - 0 = sin filtro
  - 2 = Gaussian Blur
  - 3 = Median Blur
- *indice_filtro:* índice para el filtro que se utilizará. Este valor debe ser número impar.
- nombag: Nombre del vídeo del cual se desean obtener los resultados del análisis

Al igual que en las anteriores, los campos 'resolucion', 'radio', 'grosor', 'tipo' y 'filtro' son los parámetros utilizados para el tratamiento de las imágenes. En el campo 'calibracion' indicaremos las muestras que usaremos para el entrenamiento de los métodos de clasificación. Con 'nombag' indicaremos el vídeo utilizado para el calculo de resultados.

Al finalizar la aplicación encontraremos en la carpeta 'calibracion' el fichero de resultados, *'calibracion'.result,* con dos tablas, una para cada mano para cada método de clasificación, KNN y SVM. En el anexo D se encuentran las tablas obtenidas en los análisis realizados.

Por cada gesto reconocido de manera equivocada, se guardará una imagen de la captura para poder analizar el motivo del fallo. Estas imágenes serán nombradas con el siguiente patrón:

'metodo'-'gesto'-'resultado'-'numResultado'.jpg

Por ejemplo, el archivo “knn-1-2-4.jpg” nos indicaría que es la cuarta captura que el método KNN ha reconocido como gesto de tipo 2 y se trata de gesto de tipo 1.

El gesto ’0’ nos servirá para la clasificación de posturas de la mano que no deben ser utilizadas. En ellas encontraremos las muestras capturadas al comienzo del vídeo hasta el comienzo de muestras claras.

Interfaz

Mediante el comando:

```
rosrun interfaz interfaz 'nomCalibracion'
```

lanzaremos la aplicación que cumple con el objetivo principal de este proyecto. Al inicio de la ejecución, se realizará el entrenamiento para el reconocimiento de los gestos. Para ello será necesario el haber creado previamente una calibración, mediante el programa “calibar”, con el nombre ’nomCalibracion’. Tras el entrenamiento comenzará la comunicación gestual con el robot.

Funcionamiento de la aplicación

Al comienzo de la aplicación, el robot se encontrará en estado “espera”. Esto evitará el reconocimiento de gestos indeseados al inicio de la ejecución, tras el reconocimiento de las manos.

Disponemos de varios comandos con los que nos comunicaremos con el robot:

- Activar/Desactivar: *Levantar ambos brazos*. El robot dispone de dos estados, espera y activo. Solo realizará acciones cuando se encuentre activo. Durante el tiempo de “espera” el robot estará sentado sin obedecer ninguna orden. Al levantar ambos brazos, pasará a estado “activo” y comenzará a realizar las acciones indicadas hasta su desactivación. Al cambiar a su estado inicial, el robot realizará una reverencia y procederá a sentarse hasta ser activado de nuevo.
- Acciones:
  - Palma Izquierda: caminar hacia delante
  - Palma Derecha: caminar hacia atrás
  - Ok Izquierdo: girar izquierda
  - Ok Derecho: girar derecha

Requisitos del sistema

La aplicación esta implementada y probado su uso con el software siguiente:

- Ubuntu 10.04 LTS “Lucid Lynx”
- ROS Electric
- Python 2.7
- Paquete kinect-demos del repositorio “ros-pkg”

Instalar y lanzar la aplicación

En una terminal aparte arrancaremos el sistema ROS con:

```
roscore
```

Para el uso de la aplicación deberemos tener instalado el programa “Hand_interaction” del paquete kinect-demos y posteriormente compilar el interfaz gestual. Para ello en una nueva terminal usaremos los comandos:

```
rosmake hand_interaction
rosmake interfaz
```

Si es el primer uso, será necesario realizar la captura de gestos para el entrenamiento del sistema. para ello ejecuta el módulo "calibrar" introduciendo tu nombre en el campo nombreCalibración:

```
rosrun interfaz calibrar 'nombreCalibracion' 100 0 2 3 2 19
```

Para lanzar el interfaz utilizaremos dos terminales, en la primera escribiremos:

```
roscd interfaz
python nodes/connector.py
```

y en la segunda:

```
rosrun interfaz interfaz 'nombreCalibracion'
```

Una vez lanzada la aplicación:

- Sitúese delante del sensor Kinect y muévase para ser reconocido.
- Levante los brazos situando las palmas abiertas a los lados de la cabeza hasta que sus manos sean reconocidas
- ¡¡¡Comience a mandar instrucciones al robot!!!

3. Taller realizado

Durante la celebración de la "V semana de la ingeniería y arquitectura" celebrada en la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza, se realizó un taller durante las visitas realizadas por distintos grupos de secundaria y bachiller al laboratorio del grupo de Robótica, Percepción y Tiempo Real del DIIS.

En el taller se les explicó el funcionamiento del interfaz durante una demostración y posteriormente fueron los alumnos los que probaron la aplicación (Figura 4.5).

Los resultados obtenidos fueron satisfactorios. La aplicación reconoció correctamente las instrucciones realizadas por los alumnos que probaron el interfaz. Además tuvo muy buena aceptación entre los asistentes.

Figura 4.5: Talleres realizados durante la “V semana de la ingeniería y arquitectura”.

Capítulo 5

Conclusiones y trabajos futuros

1. Conclusiones

La aparición del sensor Kinect ha supuesto una revolución en el campo de la visión por computador y de la robótica por las posibilidades que ofrece el sensor de profundidad y su bajo coste. Gracias al mapa de profundidad que Kinect aporta en tiempo real, se facilita mucho el trabajo de reconocimiento y segmentación de objetos.

Los objetivos de este proyecto fin de carrera eran la creación de un interfaz gestual para el control de un robot humanoide mediante cámaras RGB-d y su uso en la creación de un taller de divulgación orientado a estudiantes de secundaria y bachiller, objetivos que se han cumplido de manera satisfactoria. Este interfaz se ha implementado utilizando el pseudo sistema operativo ROS. Su diseño para el desarrollo de aplicaciones para robots fue muy útil en la implementación del interfaz. Aunque el uso de ROS no es complejo fue necesario documentarse sobre éste y realizar los tutoriales que se ofrecen en su pagina web para conocer las posibilidades que nos ofrece.

Para el desarrollo del interfaz, se ha utilizado el sensor Kinect en el reconocimiento de personas y la segmentación de sus manos. Ha sido necesaria una intensa búsqueda de información y aplicaciones desarrolladas para tal fin, ya que hay muchos ejemplos en Internet, pero no siempre la documentación es exacta o resulta fácil ponerlos en marcha. En muchos de los ejemplos encontrados, no fue posible su instalación por incompatibilidades de software o por falta de documentación.

Respecto a los métodos estudiados para la clasificación de las imágenes de las manos, uno de tipo generativo, *K Nearest Neighbors,* y otro de tipo discriminativo, *Support Vector Machine,* los resultados muestran que con pocos gestos y pocas muestras es más efectivo el modelo generativo, pero si aumentamos el número de muestras, el modelo discriminativo consigue unos resultados semejantes.

El robot utilizado para el desarrollo del proyecto ha sido RoboNova-1. Al utilizar una plataforma de hardware real (frente a un simulador), permite resultados más interesantes, pero también problemas prácticos. Por ejemplo, surgieron problemas a la hora de realizar la comunicación con el robot por línea serie desde las aplicaciones, ya que RoboNova-1 no posee drivers en ROS. Partiendo del driver creado para el robot Robobuilder y tras realizar algunas modificaciones se consiguió la conexión con nuestro robot, pero fue necesario el estudio de los protocolos de comunicación, a través de línea serie, utilizados por RoboNova-1 y el estudio del lenguaje Python, lenguaje en el que está implementado el driver utilizado.

Con el interfaz gestual en funcionamiento se realizó un taller, en el laboratorio de robótica del I3A, durante la semana de la ingeniería. En ellos, participaron estudiantes de diversos grupos, desde 3º de la ESO hasta 2º de Bachiller. El resultado fue muy positivo ya que se comprobó el correcto funcionamiento de la aplicación y obtuvo buena aceptación entre los estudiantes.

2. Trabajo futuro

Tras la realización de este proyecto, se proponen algunas líneas de trabajo futuro:

- Insertar más gestos a reconocer por el sistema. Se podrían insertar más gestos que implicarían más instrucciones para la comunicación con el robot. Para ello se debe hacer un estudio de diferentes posturas de mano que los sistemas de clasificación puedan separar claramente en clases diferenciadas e insertar muestras suficientes para su correcta clasificación.
- Durante la calibración existen muestras que son etiquetadas en un determinado gesto pero, por error en la captura de puntos de la Kinect o por error humano, son muestras que pueden perjudicar en el posterior reconocimiento de manos. Para ello se puede mejorar la calibración retirando todas aquellas muestras no deseadas.
- Completar el driver para RoboNova-1 mediante la adaptación completa del script de Robobuilder utilizado.
- Incorporar sensores a RoboNova-1. El robot solo dispone de un sensor de infrarrojos que comunica con un mando a distancia. El chip del robot posee varios conectores libres para incorporar sensores que podrían ser muy útiles. Un sensor de posición nos sería de mucha utilidad debido a la poca estabilidad de este robot. Con este sensor instalado el robot podría incorporarse de nuevo al caer. Por otro lado una conexíon inalámbrica para la comunicación con el robot daría más libertad de movimientos a este.
- Utilizar el interfaz gestual para el control de otro tipo de robots o dispositivos. La aplicación se podría adaptar para su uso con brazos de robots, para el control de robots domésticos o como ratón del ordenador.

3. Valoración personal

La elección de este proyecto de fin de carrera fue por el interés personal que tenía en los campos de la robótica y la inteligencia artificial. Mi desconocimiento, tanto en el campo de la robótica como en el de visión por computador, ha supuesto la dedicación de mucho tiempo al estudio de las herramientas a utilizar, sus librerías y sus posibilidades, así como de diferentes métodos de clasificación para el reconocimiento de los gestos.

Tras la realización de los talleres durante la semana de la ingeniería comprobé el resultado de mi proyecto. Considero que es un buen método para que la gente aprenda algunos conocimientos sobre robótica y visión por computador de manera comprensible y amena.

Este proyecto me ha supuesto una gran satisfacción personal, por la cantidad de conocimientos obtenidos sobre temas que me parecen de gran interés, por el uso de software libre en el desarrollo de la aplicación, por la buena aceptación que obtuvo por parte de los estudiantes en los talleres realizados y por la motivación que me ha dado a seguir “trasteando” con la robótica y la inteligencia artificial.

Bibliografía

[1] K. Lai, L. Bo, X. Ren, D. Fox. "RGB-D Object Recognition: Features, Algorithms, and a Large Scale Benchmark". Consumer Depth Cameras for Computer Vision: Research Topics and Applications, 2013. 1

[2] M. Krainin, K. Konolige, D. Fox. "Exploiting Segmentation for Robust 3D Object Matching". ICRA, 2012. 1

[3] L Xia, CC Chen, JK Aggarwal. "Human detection using depth information by Kinect". Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference, 2011. 1

[4] Z Ren, J Meng, J Yuan, Z Zhang. "Robust hand gesture recognition with kinect sensor". Proceeding MM '11 Proceedings of the 19th ACM international conference on Multimedia. Pages 759-760. 2011. 1

[5] Z Ren, J Yuan, Z Zhang "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera". Proceeding MM '11 Proceedings of the 19th ACM international conference on Multimedia. Pages 1093-1096. 2011. 1, 1

[6] JP Wachs, M Kölsch, H Stern, Y Edan."Vision-based hand-gesture applications".Magazine Communications of the ACM CACM Homepage archive Volume 54 Issue 2. Pages 60-71. 2011. 1

[7] P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox. "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments". International Journal of Robotics. Research 31:5, 2012. 1

[8] MNK Boulos, BJ Blanchard, C Walke. "Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation". International Journal of Health Geographics. 2011 1

[9] W. Choi,C. Pantofaru, S. Savarese. "Detecting and Tracking People using an RGB-D Camera via Multiple Detector Fusion". Workshop on Challenges and Opportunities in Robot Perception, at the International Conference on Computer Vision (ICCV). 2011 1

[10] David Bueno Monge , "Reconocimiento de Objetos en 3D Utilizando Sensores de Visión y Profundidad de Bajo Coste", 2012. Proyecto fin de carrera. 1

[11] Radu Bogdan Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)". In IEEE International Conference on Robotics and Automation (ICRA), 2011. 2, 3

[12] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng. "ROS: an open-source Robot Operating System". Retrieved 3 April 2010. 2, 3

[13] González, R.C., Wintz, P. "Procesamiento digital de imágenes". Addison-Wesley. 1996 2

[14] Shapiro, L. G. & Stockman, G. C: "Computer Vision", page 137, 150. Prentence Hall, 2001 2

[15] G.R. Arce, "Nonlinear Signal Processing: A Statistical Approach", Wiley:New Jersey, USA, 2005. 2

[16] N. Hagen and E. L. Dereniak, "Gaussian profile estimation in two dimensions," Appl. Opt. 47:6842-6851, 2008 2

[17] E. Arias-Castro and D.L. Donoho, "Does median filtering truly preserve edges better than linear filtering?", Annals of Statistics, vol. 37, no. 3, pp. 1172–2009. 2

[18] Stuart Russell and Peter Norvig. "Artificial Intelligence: A Modern Approach", second edition, p. 733. Prentice Hall 2003 3

[19] Richard O. Duda, Peter E. Hart, David G. Stork. "Pattern classification ". 2001. 3

[20] Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995 3

[21] Papoulis, A. "Conditional Probabilities and Independent Sets." §2-3 in Probability, Random Variables, and Stochastic Processes, 2nd ed. New York: McGraw-Hill, pp. 33-45, 1984. 3

[22] http://www.ros.org/wiki/openni 3

[23] http://opencv.willowgarage.com/documentation/cpp/index.html 3

[24] Robosavvy: http://robosavvy.com/site/ 1

Anexos

Anexo A

ROS

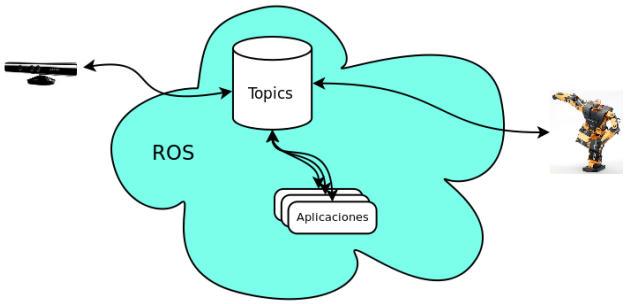


Figura A.1: Diagrama de sistema de comunicación utilizado en ROS.

ROS, cuyas siglas significan "Robot Operating System", es un pseudo sistema operativo que nos proporciona librerías y herramientas para el desarrollo de software para robots. ROS provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos, que pueden recibir, mandar y multiplexar: sensores, control, estado, planificación y otros mensajes.

ROS implementa varios estilos diferentes de comunicación, incluyendo un sistema síncrono de estilo RPC[1] de comunicación a través de los servicios, la transmisión asíncrona de datos a través de "topics", y el almacenamiento de datos en un servidor de parámetros. Los diferentes nodos que se ejecutan en ROS podán publicar los nuevos valores de los "topics" y subscribirse a éstos para consultarlos. A pesar de que ROS no trabaja en un marco de tiempo real, si que es posible el integrar en ROS código en tiempo real. Podríamos describir el grafo del tiempo de ejecución como una red peer-to-peer de procesos que están débilmente acoplados utilizando la infraestructura de comunicación ROS.

La librería está orientada para un sistema UNIX, siendo Ubuntu el sistema en el que ofrecen soporte. Para el desarrollo de este proyecto se utiliza ROS Electric por ser la versión con soporte más estable del momento. Dicha versión funciona bajo el sistema operativo "Ubuntu 10.04 Lucid Lynx[2]".

[1] Remote Procedure Call
[2] http://releases.ubuntu.com/lucid/

Anexo B

Sensor Kinect

Figura B.1: Sensor Kinect.

Kinect es un controlador de juego libre y entretenimiento creado por ALEX Kipman y desarrollado por Microsoft. En noviembre de 2010 fue lanzada para su uso con la videoconsola Xbox 360, y a partir de junio de 2011 también para PC, para su uso con Windows 7 y Windows 8.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un micrófono de múltiples matrices y un mecanismo de inclinación motorizado. El sensor de profundidad es un proyector de infrarrojos combinado con un sensor CMOS monocromo que permite a Kinect ver el entorno en 3D en cualquier condición de luz ambiental. El rango de detección de la profundidad del sensor es ajustable gracias al software de Kinect capaz de calibrar automáticamente el sensor, basado en la jugabilidad y en el ambiente físico del jugador, tal como la presencia de sofás.

El hardware de Kinect se ha confirmado que se basará en un diseño de referencia y la tecnología 3D-calor fabricados por la compañía israelí de desarrollo PrimeSense Ltd.

El sensor proporciona vídeo a una frecuencia de 30 Hz, con una resolución de 640x480 píxeles, utilizando una profundidad de 8 bits en el sensor RGB y 11 bits en el sensor de profundidad, ofreciendo hasta 2048 niveles de sensibilidad. El sensor de infrarrojos tiene un rango de funcionamiento que va desde 0.7 hasta 6 metros y tiene un campo de visión angular de 57^{o} horizontalmente y de 43^{o} verticalmente, mientras que el pivote motorizado puede inclinar el sensor hasta 27^{o} verticalmente.

En noviembre de 2010, Industrias Adafruit ofreció una recompensa por un controlador de código abierto para Kinect. El 10 de noviembre, se anunció al español Héctor Martín como el ganador, que usó métodos de ingeniería inversa con Kinect y desarrolló un controlador para GNU/Linux que permite el uso de la cámara RGB y las funciones de profundidad. Desde entonces han salido diferentes controladores de código abierto para su uso con diferentes sistemas operativos. Actualmente, este sensor es una herramienta utilizada en multitud de investigaciones relacionadas con la visión por computador, como la captura de gestos, reconstrucción de entornos 3D, detección y reconocimiento de objetos...

Anexo C

RoboNova-1

Figura C.1: Robot RoboNova-1.

Rovonova-I es un robot humanoide que se compone de 16 servos digitales HSR 8498HB, especialmente desarrollados para este robot y que incluyen características especiales como ”Motión Feedback”, lo que le da posibilidad de leer externamente la posición real del servo y permite que se pueda colocar el robot manualmente en cualquier posición y luego leer y guardar la posición en un programa leyendo los valores de los 16 servos desde el propio controlador.

El robot RoboNova-1 incorpora el controlador Hitec MR-C3024, con el microcontrolador Atmel ATMega 128 capaz de controlar hasta 24 servos. Éste controlador cuenta entre otras cosas con 40 puertos de entrada y salida digitales, puerto serie, bus I2C, 8 entradas analógicas, altavoz y led. También dispone demás de 64 KBytes de memoria interna para almacenar programas.

El robot posee un software propio basado en el lenguaje RoboBasic (Figura C.2), que como su nombre indica es un lenguaje del tipo Basic pero especializado y orientado a robots. Esto significa que por un lado es muy fácil de aprender, de hecho si se sabe programar en basic, ya sabe programar en RoboBasic y por otro lado que incluye gran variedad de comandos específicos para controlar las funciones del robot que facilitan mucho la tarea y simplifican enormemente el proceso de programación. Mediante la función 'Catch & Play' podemos crear movimientos poniendo el robot en la posición deseada manualmente y el programa creará el movimiento del robot.

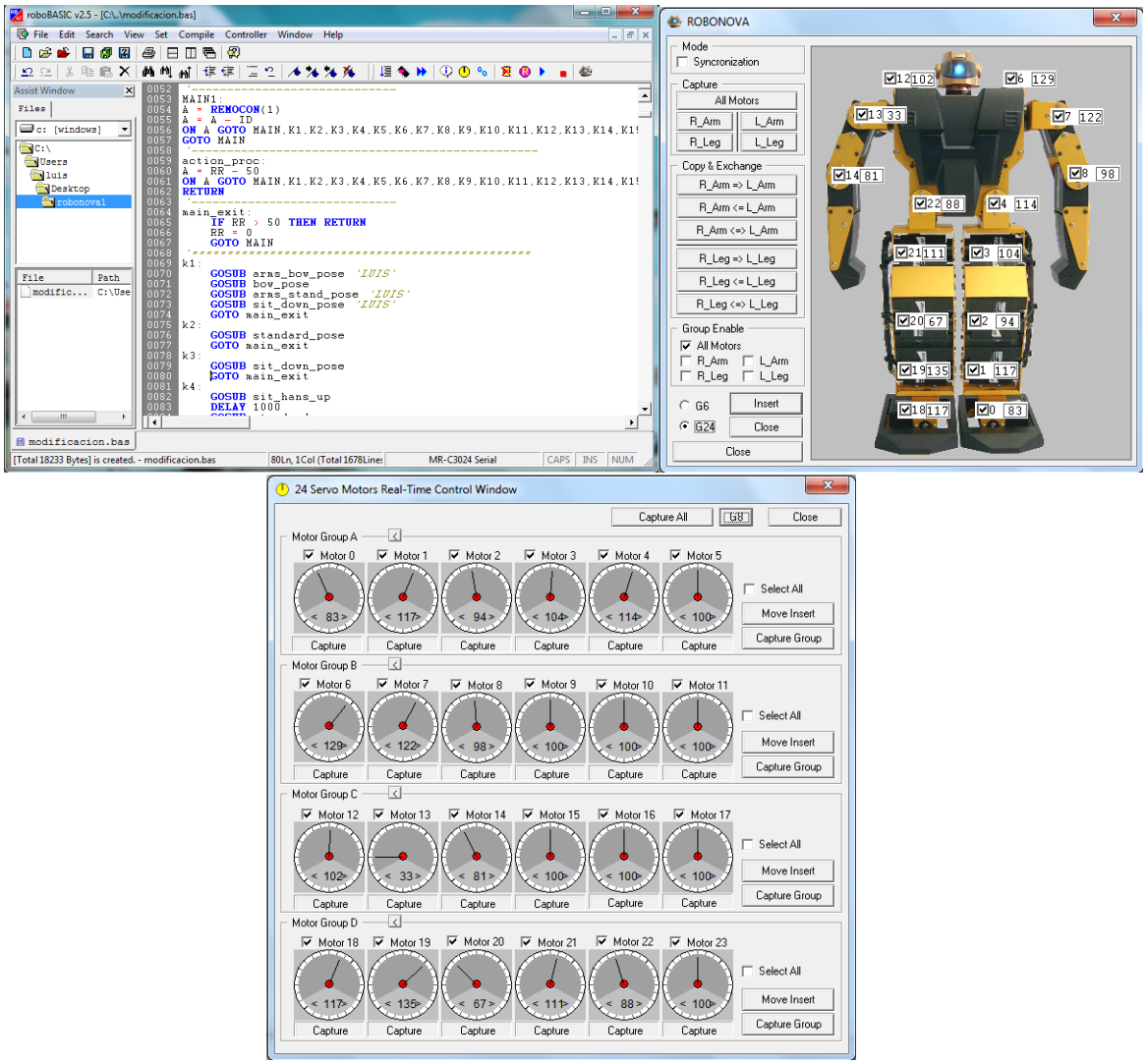


Figura C.2: (Arriba) Entorno de programación roboBasic. (Centro) Panel para el control en tiempo real de los servos. (Abajo) Otro panel para el control del robot de manera más visual.

Existen diferentes sensores para RoboNova-1:

- Dispositivo inalámbrico
- Sensor de infrarrojos
- Sensor de inclinación
- Sensor de ultrasonidos
- Acelerómetro
- Sensor de distancias por infrarrojos
- Micrófono
- Sensor de luz

Anexo D

Pruebas y resultados

En este anexo se adjuntan los resultados de las diferentes pruebas que se han ido realizando para la elección del formato de imagen y sistema de reconocimiento ha utilizar.

Todas las pruebas realizadas en cada una de las fases se han realizado con el mismos vídeo utilizando el programa "analizar". Con esta aplicación obtendremos un fichero, *'calibracion'.result,* con los resultados. En el encontraremos dos tablas, una para cada mano, por método de clasificación, KNN y SVM, como la tabla D.1:

GESTOS	0	1	2
0			
1			
2			

Tabla D.1: Tabla de resultados.

En ellas se mostrará, para cada gesto los aciertos y fallos obtenidos, enumerando los errores en función del gesto a reconocer y el reconocido. La tabla tendrá una fila y una columna por cada gesto. Cada fila corresponde a cada uno de los gestos. Cada columna nos indicará la cantidad de reconocimientos obtenidos por cada gesto. El gesto '0' nos servirá para la clasificación de posturas de la mano que no deben ser utilizadas. En ellas encontraremos las muestras capturadas al comienzo del vídeo hasta el comienzo de muestras claras.

1. Fase 1: Selección de propiedades de las imágenes

Resultados obtenidos en las pruebas para la selección del tamaño de la imagen y el grosor del punto. Las pruebas se han realizado todas mediante la misma grabación "luisPalmOk.bag", para así poder comparar de manera más fiable los resultados obtenidos. A partir de los datos expuestos a continuación, se observa que los mejores resultados se obtienen con las propiedades de las imágenes que se recopilan en la tabla D.2, siendo en todas ellas los resultados muy similares. En la selección realizada podemos ver que todas las pruebas mantienen una proporción entre el tamaño de la imagen y el grosor del punto utilizado en la proyección. Si utilizamos un tamaño de punto muy pequeño no obtenemos una imagen suficientemente nítida y si, por el contrario, utilizamos un grosor muy grande, la imagen pierde detalle.

PRUEBA	RESOLUCIÓN	RADIO	GROSOR LINEA	KNN	SVM
2	100	0	2	0.56	0.07
3	100	0	3	0.57	0.07
4	150	0	2	0.56	0.27
5	150	0	3	0.56	0.21
7	150	1	2	0.57	0.10
8	150	1	3	0.58	0.09
12	200	1	2	0.48	0.10
13	200	1	3	0.57	0.11

Tabla D.2: Pruebas seleccionadas y resultados de la precisión.

Prueba 1:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 1

Figura D.1: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	0	4	0	0	1	3
1	0	13	43	1	0	6	48
2	0	9	33	2	0	2	41

Precisión obtenida mediante KNN = 0.45

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	0	4	0
1	1	51	4	1	0	54	0
2	0	34	8	2	0	42	1

Precisión obtenida mediante SVM = 0.55

Tabla D.3: Resultados de la prueba 1

Prueba 2:

Píxeles de lado: 100
Radio del círculo: 0

Grosor de línea: 2

Figura D.2: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	0	46	9	1	45	1	9
2	0	7	36	2	8	1	33

Precisión obtenida mediante KNN = 0.56

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	47	8	0	1	48	7	0
2	22	21	0	2	23	19	0

Precisión obtenida mediante SVM = 0.07

Tabla D.4: Resultados de la prueba 2

Prueba 3:

Píxeles de lado: 100

Radio del círculo: 0

Grosor de línea: 3

Figura D.3: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	0	46	9	1	45	1	8
2	0	7	36	2	8	1	34

Precisión obtenida mediante KNN = 0.57

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	48	7	0	1	47	7	0
2	22	21	0	2	19	23	1

Precisión obtenida mediante SVM = 0.07

Tabla D.5: Resultados de la prueba 3

Prueba 4:

Píxeles de lado: 150

Radio del círculo: 0

Grosor de línea: 2

Figura D.4: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	1	46	9	1	46	1	8
2	0	7	35	2	7	1	34

Precisión obtenida mediante KNN = 0.56

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	37	19	0	1	52	3	0
2	7	1	34	2	32	10	0

Precisión obtenida mediante SVM = 0.27

Tabla D.6: Resultados de la prueba 4

Prueba 5:

Píxeles de lado: 150

Radio del círculo: 0

Grosor de línea: 3

Figura D.5: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	3	44	9	1	45	1	8
2	0	6	36	2	8	1	34

Precisión obtenida mediante KNN = 0.56

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	47	9	0	1	45	1	8
2	16	24	2	2	12	1	30

Precisión obtenida mediante SVM = 0.21

Tabla D.7: Resultados de la prueba 5

Prueba 6:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 1

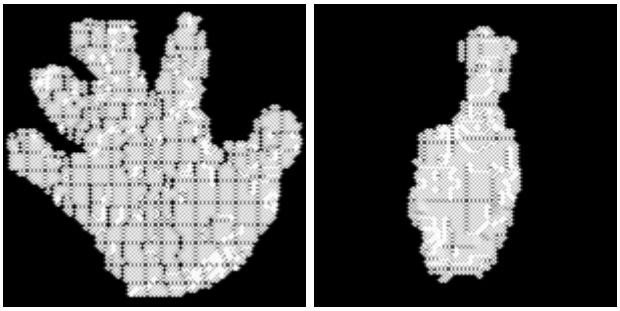


Figura D.6: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	14	11	31	1	41	1	12
2	0	3	39	2	8	1	34

Precisión obtenida mediante KNN = 0.41

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	22	33	1	1	47	6	1
2	2	35	5	2	20	15	8

Precisión obtenida mediante SVM = 0.25

Tabla D.8: Resultados de la prueba 6

Prueba 7:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 2

Figura D.7: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	0	47	9	1	45	1	8
2	0	6	36	2	8	1	34

Precisión obtenida mediante KNN = 0.57

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	47	9	0	1	47	6	1
2	18	24	0	2	23	15	5

Precisión obtenida mediante SVM = 0.10

Tabla D.9: Resultados de la prueba 7

Prueba 8:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 3

Figura D.8: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	3	1	0
1	0	47	9	1	45	1	8
2	0	6	36	2	8	1	34

Precisión obtenida mediante KNN = 0.58

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	45	11	0	1	47	7	0
2	25	17	0	2	21	22	0

Precisión obtenida mediante SVM = 0.09

Tabla D.10: Resultados de la prueba 8

Prueba 9:

Píxeles de lado: 200

Radio del círculo: 0

Grosor de línea: 2

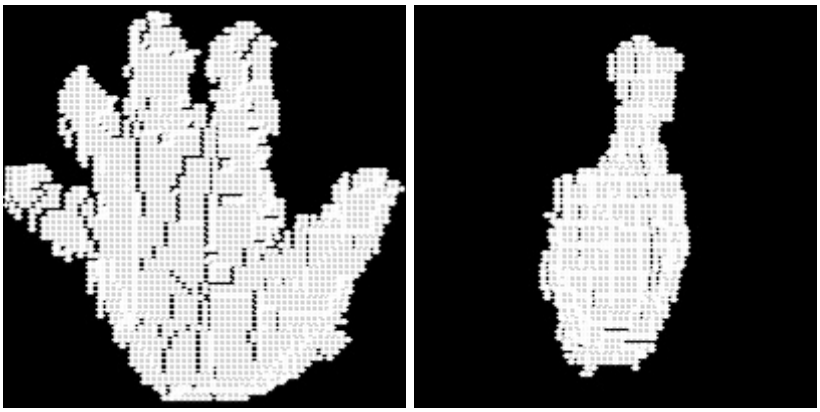


Figura D.9: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	2	1	1
1	14	29	13	1	38	6	10
2	0	7	35	2	8	1	34

Precisión obtenida mediante KNN = 0.51

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	19	37	0	1	45	5	4
2	1	41	0	2	19	14	10

Precisión obtenida mediante SVM = 0.25

Tabla D.11: Resultados de la prueba 9

Prueba 10:

Píxeles de lado: 200

Radio del círculo: 0

Grosor de línea: 3

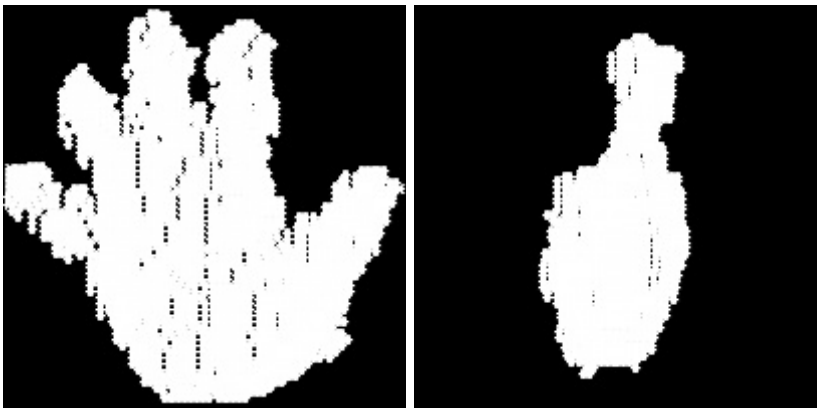


Figura D.10: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	6	38	11	1	45	1	8
2	0	6	37	2	8	1	34

Precisión obtenida mediante KNN = 0.53

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	4	0	0
1	28	27	0	1	46	8	0
2	4	39	0	2	21	22	0

Precisión obtenida mediante SVM = 0.17

Tabla D.12: Resultados de la prueba 10

Prueba 11:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 1

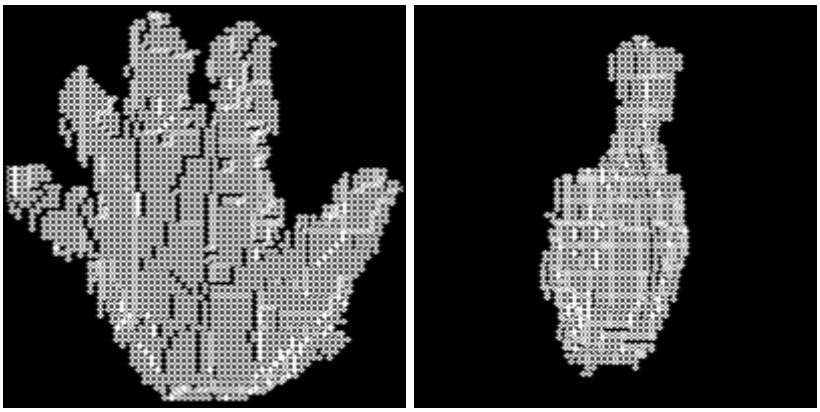


Figura D.11: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	2	1	0	0	2	2
1	0	8	48	1	1	29	25
2	0	3	39	2	0	5	37

Precisión obtenida mediante KNN = 0.55

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	1	3	0
1	1	55	0	1	20	35	0
2	0	41	1	2	5	37	0

Precisión obtenida mediante SVM = 0.44

Tabla D.13: Resultados de la prueba 11

Prueba 12:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 2

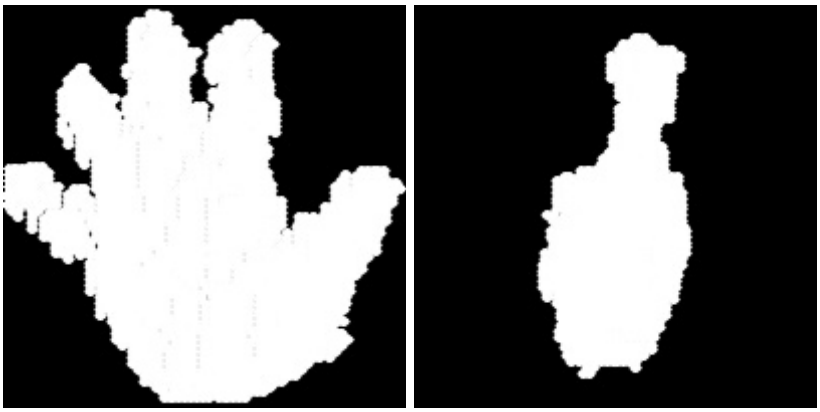


Figura D.12: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	0	47	9	1	40	16	0
2	0	6	36	2	9	33	0

Precisión obtenida mediante KNN = 0.48

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	40	16	0	1	50	4	0
2	9	33	0	2	30	13	0

Precisión obtenida mediante SVM = 0.10

Tabla D.14: Resultados de la prueba 12

Prueba 13:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 3

Figura D.13: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	4	0	0
1	0	46	9	1	45	1	8
2	0	7	36	2	8	1	34

Precisión obtenida mediante KNN = 0.57

LEFT	0	1	2	RIGHT	0	1	2
0	3	1	0	0	3	1	0
1	41	14	0	1	47	7	0
2	14	29	0	2	22	20	1

Precisión obtenida mediante SVM = 0.11

Tabla D.15: Resultados de la prueba 13

2. Fase 2: Repetición de las pruebas con los mejores resultados

En esta segunda fase, se han repetido las pruebas para las características que mejores resultados han obtenido. En esta segunda fase se ha utilizado la grabación "luisPalmOk2.bag".

Tras observar los resultados, que se mostrarán a continuación, se realizó una segunda criba de resultados entre los que se seleccionaron los que muestra la tabla D.16.

PRUEBA	RESOLUCIÓN	RADIO	GROSOR LINEA	KNN	SVM
1	100	0	2	0.73	0.61
2	100	0	3	0.74	0.62
5	150	1	2	0.73	0.58
6	150	1	3	0.76	0.56
8	200	1	3	0.72	0.61

Tabla D.16: Resultados de la prueba

Prueba 1:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 2

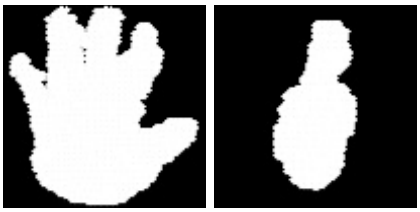


Figura D.14: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo "ok".

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	2	1
1	1	44	18	1	0	30	33
2	0	10	62	2	0	6	66

Precisión obtenida mediante KNN = 0.73

LEFT	0	1	2	RIGHT	0	1	2
0	0	1	2	0	0	0	3
1	0	22	41	1	0	7	56
2	0	3	69	2	0	1	71

Precisión obtenida mediante SVM = 0.61

Tabla D.17: Resultados de la prueba 1

Prueba 2:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 3

Figura D.15: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	1	49	14	1	0	30	34
2	0	11	60	2	0	5	66

Precisión obtenida mediante KNN = 0.74

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	2	52	10	1	0	48	16
2	0	13	58	2	0	59	12

Precisión obtenida mediante SVM = 0.62

Tabla D.18: Resultados de la prueba 2

Prueba 3:

Píxeles de lado: 150

Radio del círculo: 0

Grosor de línea: 2

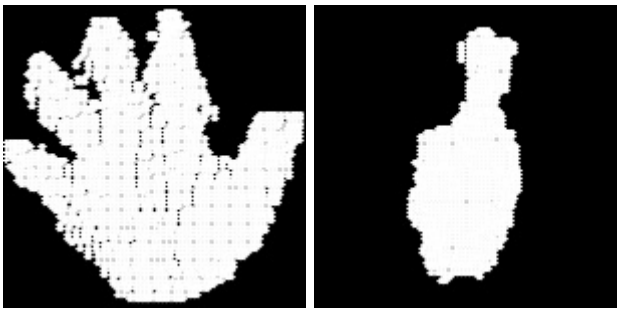


Figura D.16: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo "ok".

LEFT	0	1	2	RIGHT	0	1	2
0	0	1	2	0	0	2	1
1	0	13	50	1	0	25	40
2	0	1	71	2	0	6	64

Precisión obtenida mediante KNN = 0.62

LEFT	0	1	2	RIGHT	0	1	2
0	0	1	2	0	0	3	0
1	0	30	33	1	0	65	0
2	0	4	68	2	0	70	0

Precisión obtenida mediante SVM = 0.59

Tabla D.19: Resultados de la prueba 3

Prueba 4:

Píxeles de lado: 150

Radio del círculo: 0

Grosor de línea: 3

Figura D.17: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	1	37	26	1	0	17	46
2	0	7	64	2	0	5	67

Precisión obtenida mediante KNN = 0.67

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	2	1
1	0	55	9	1	0	61	2
2	0	25	46	2	0	70	2

Precisión obtenida mediante SVM = 0.59

Tabla D.20: Resultados de la prueba 4

Prueba 5:

Píxeles de lado: 150
Radio del círculo: 1
Grosor de línea: 2

Figura D.18: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	3	0
1	1	49	14	1	0	30	33
2	0	11	60	2	0	7	65

Precisión obtenida mediante KNN = 0.73

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	0	50	14	1	0	58	5
2	0	25	46	2	0	67	5

Precisión obtenida mediante SVM = 0.58

Tabla D.21: Resultados de la prueba 5

Prueba 6:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 3

Figura D.19: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	3	0
1	1	52	11	1	0	37	28
2	0	11	60	2	0	9	61

Precisión obtenida mediante KNN = 0.76

LEFT	0	1	2	RIGHT	0	1	2
0	1	0	2	0	0	1	2
1	6	40	18	1	0	42	23
2	1	9	61	2	0	57	13

Precisión obtenida mediante SVM = 0.56

Tabla D.22: Resultados de la prueba 6

rueba 7:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 2

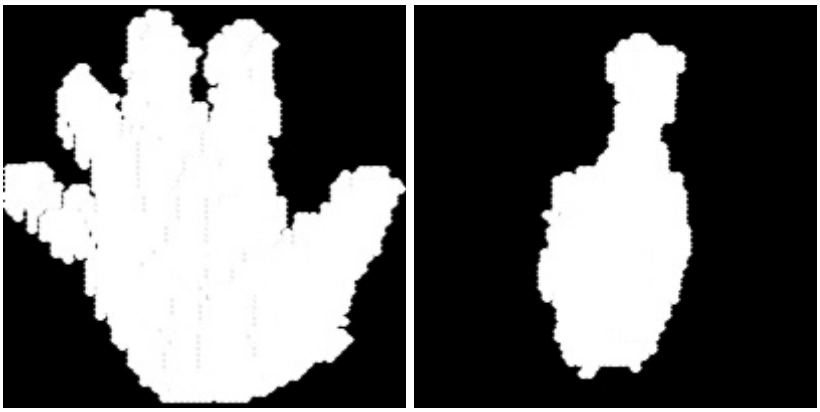


Figura D.20: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	1	2
1	1	31	32	1	0	22	44
2	0	7	64	2	0	4	65

Precisión obtenida mediante KNN = 0.66

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	0	56	8	1	0	43	23
2	0	31	40	2	0	60	9

Precisión obtenida mediante SVM = 0.53

Tabla D.23: Resultados de la prueba 7

Prueba 8:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 3

Figura D.21: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	1	2
1	2	48	14	1	0	23	40
2	0	11	60	2	0	4	68

Precisión obtenida mediante KNN = 0.72

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	3	0
1	0	47	17	1	0	63	0
2	0	12	59	2	0	72	0

Precisión obtenida mediante SVM = 0.61

Tabla D.24: Resultados de la prueba 8

3. Fase 3: Selección de filtro

A partir de los resultados obtenidos, se han realizado de nuevo las pruebas pasando las imágenes por filtros que mejoren la imagen a reconocer. Se han utilizado los filtros "Median Blur" y "Gaussian Blur" de las librerías de OpenCv.

Tras estos experimentos se decidió que la mejor configuración a utilizar en la versión final de la aplicación, para cada método, es:

- En el caso de búsqueda por KNN utilizaremos:
  - Píxeles de lado: 100
  - Radio del círculo:0
  - Grosor de línea: 2
  - Filtro: Gaussian Blur
- Para la búsqueda mediante SVM el mejor resultado lo obtenemos con:
  - Pixeles de lado: 100
  - Radio del círculo:0
  - Grosor de línea: 2
  - Filtro: Median Blur

Prueba 1:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 2
Filtro: ninguno

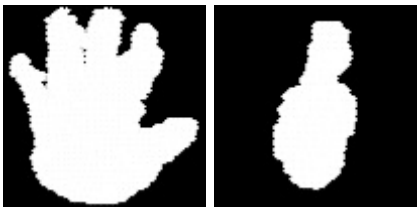


Figura D.22: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	2	1
1	1	44	18	1	0	30	33
2	0	10	62	2	0	6	66

Precisión obtenida mediante KNN = 0.73

LEFT	0	1	2	RIGHT	0	1	2
0	0	1	2	0	0	0	3
1	0	22	41	1	0	7	56
2	0	3	69	2	0	1	71

Precisión obtenida mediante SVM = 0.61

Tabla D.25: Resultados de la prueba 1

Prueba 2:

Píxeles de lado: 100

Radio del círculo: 0

Grosor de línea: 2

Filtro: Gaussian Blur

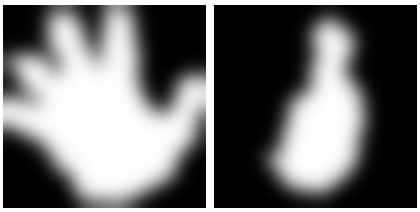


Figura D.23: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	3	0	0	0	3	0
1	0	54	10	1	0	52	11
2	0	12	59	2	0	15	57

Precisión obtenida mediante KNN = 0.80

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	0	3
1	0	59	5	1	0	13	50
2	0	56	15	2	0	56	16

Precisión obtenida mediante SVM = 0.37

Tabla D.26: Resultados de la prueba 2

Prueba 3:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 2
Filtro: Median Blur

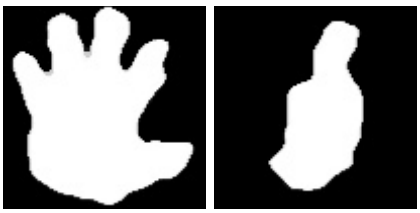


Figura D.24: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	2	1
1	1	40	22	1	0	28	35
2	0	10	62	2	0	6	66

Precisión obtenida mediante KNN = 0.71

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	0	3
1	1	53	9	1	0	39	24
2	0	11	61	2	9	46	26

Precisión obtenida mediante SVM = 0.65

Tabla D.27: Resultados de la prueba 3

Prueba 4:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 3
Filtro: Ninguno

Figura D.25: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo "ok".

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	1	49	14	1	0	30	34
2	0	11	60	2	0	5	66

Precisión obtenida mediante KNN = 0.74

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	2	52	10	1	0	48	16
2	0	13	58	2	0	59	12

Precisión obtenida mediante SVM = 0.61

Tabla D.28: Resultados de la prueba 4

Prueba 5:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 3
Filtro: Gaussian Blur

Figura D.26: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo "ok".

LEFT	0	1	2	RIGHT	0	1	2
0	0	3	0	0	0	3	0
1	1	59	10	1	0	52	11
2	0	12	59	2	0	15	57

Precisión obtenida mediante KNN = 0.82

LEFT	0	1	2	RIGHT	0	1	2
0	0	3	0	0	0	3	0
1	1	63	0	1	0	63	0
2	0	70	1	2	0	72	0

Precisión obtenida mediante SVM = 0.46

Tabla D.29: Resultados de la prueba 5

Prueba 6:

Píxeles de lado: 100
Radio del círculo: 0
Grosor de línea: 3
Filtro: Median Blur

Figura D.27: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	3	0
1	2	35	27	1	0	39	26
2	0	7	64	2	0	9	61

Precisión obtenida mediante KNN = 0.72

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	3	0
1	0	44	20	1	0	65	0
2	0	9	61	2	0	70	0

Precisión obtenida mediante SVM = 0.61

Tabla D.30: Resultados de la prueba 6

Prueba 7:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 2

Filtro: Ninguno

Figura D.28: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo "ok".

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	3	0
1	1	49	14	1	0	30	33
2	0	11	60	2	0	7	65

Precisión obtenida mediante KNN = 0.73

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	0	50	14	1	0	58	5
2	0	25	46	2	0	67	5

Precisión obtenida mediante SVM = 0.57

Tabla D.31: Resultados de la prueba 7

Prueba 8:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 2

Filtro: Gaussian Blur

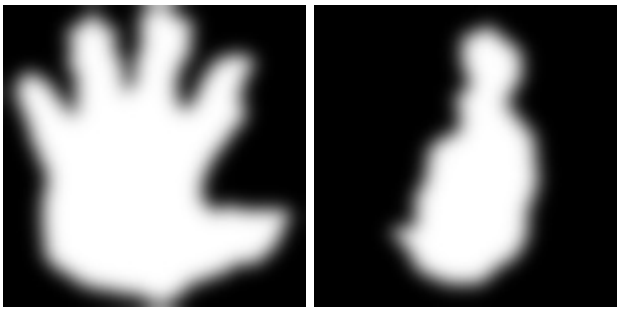


Figura D.29: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	2	0	0	0	3	0
1	2	52	9	1	0	53	11
2	0	11	61	2	0	14	57

Precisión obtenida mediante KNN = 0.81

LEFT	0	1	2	RIGHT	0	1	2
0	0	0	3	0	0	3	0
1	3	0	60	1	0	64	0
2	0	0	72	2	0	71	0

Precisión obtenida mediante SVM = 0.49

Tabla D.32: Resultados de la prueba 8

Prueba 9:

Píxeles de lado: 150
Radio del círculo: 1
Grosor de línea: 2
Filtro: Median Blur

Figura D.30: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	2	1
1	1	50	12	1	0	28	36
2	0	11	61	2	0	5	66

Precisión obtenida mediante KNN = 0.74

LEFT	0	1	2	RIGHT	0	1	2
0	2	0	1	0	0	0	3
1	21	33	9	1	0	5	59
2	1	10	61	2	0	0	71

Precisión obtenida mediante SVM = 0.61

Tabla D.33: Resultados de la prueba 9

Prueba 10:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 3

Filtro: ninguno

Figura D.31: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	3	0
1	1	52	11	1	0	37	28
2	0	11	60	2	0	9	61

Precisión obtenida mediante KNN = 0.76

LEFT	0	1	2	RIGHT	0	1	2
0	1	0	2	0	0	1	2
1	6	40	18	1	0	42	23
2	1	9	61	2	0	57	13

Precisión obtenida mediante SVM = 0.56

Tabla D.34: Resultados de la prueba 10

Prueba 11:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 3

Filtro: Gaussian Blur

Figura D.32: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	2	0	0	0	3	0
1	2	50	10	1	0	52	11
2	0	13	60	2	0	15	57

Precisión obtenida mediante KNN = 0.79

LEFT	0	1	2	RIGHT	0	1	2
0	0	0	3	0	0	2	1
1	0	17	45	1	0	63	0
2	0	60	13	2	0	71	1

Precisión obtenida mediante SVM = 0.34

Tabla D.35: Resultados de la prueba 11

Prueba 12:

Píxeles de lado: 150

Radio del círculo: 1

Grosor de línea: 3

Filtro: Median Blur

Figura D.33: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	3	0
1	2	51	10	1	0	32	32
2	0	11	61	2	0	11	60

Precisión obtenida mediante KNN = 0.74

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	1	2
1	14	38	11	1	0	38	26
2	1	19	52	2	0	57	14

Precisión obtenida mediante SVM = 0.51

Tabla D.36: Resultados de la prueba 12

Prueba 13:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 3

Filtro: Ninguno

Figura D.34: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	1	2
1	2	48	14	1	0	23	40
2	0	11	60	2	0	4	68

Precisión obtenida mediante KNN = 0.72

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	3	0
1	0	47	17	1	0	63	0
2	0	12	59	2	0	72	0

Precisión obtenida mediante SVM = 0.61

Tabla D.37: Resultados de la prueba 13

Prueba 14:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 3

Filtro: Gaussian Blur

Figura D.35: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	2	1	0	0	0	3	0
1	2	52	10	1	0	49	14
2	0	11	60	2	9	15	57

Precisión obtenida mediante KNN = 0.79

LEFT	0	1	2	RIGHT	0	1	2
0	2	0	1	0	0	3	0
1	6	48	10	1	0	63	0
2	0	11	60	2	0	72	0

Precisión obtenida mediante SVM = 0.61

Tabla D.38: Resultados de la prueba 14

Prueba 15:

Píxeles de lado: 200

Radio del círculo: 1

Grosor de línea: 3

Filtro: Median Blur

Figura D.36: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	1	1	1	0	0	2	1
1	2	49	12	1	0	23	41
2	0	11	61	2	0	4	67

Precisión obtenida mediante KNN = 0.72

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	1	2
1	1	54	8	1	0	50	14
2	0	12	60	2	0	64	7

Precisión obtenida mediante SVM = 0.61

Tabla D.39: Resultados de la prueba 15

4. Fase 4: Doble filtrado

En esta prueba he utilizado las propiedades que mejor resultado han dado, es decir:

- Píxeles de lado: 100
- Radio del círculo: 0
- Grosor de línea: 2

y le he pasado en primer lugar un filtro Gaussiano y posteriormente un filtro de mediana obteniendo en este caso los mejores resultados para el caso de clasificación mediante SVM.

Figura D.37: Capturas de los gestos: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo ”ok”.

LEFT	0	1	2	RIGHT	0	1	2
0	0	2	1	0	0	2	1
1	1	43	20	1	0	28	35
2	0	11	60	2	0	5	67

Precisión obtenida mediante KNN = 0.71

LEFT	0	1	2	RIGHT	0	1	2
0	0	1	2	0	0	1	2
1	1	35	28	1	0	42	21
2	0	8	63	2	0	12	60

Precisión obtenida mediante SVM = 0.72

Tabla D.40: Resultados de la prueba de doble filtrado

Con los resultados obtenidos en esta cuarta fase de experimentos, se ha decidido el continuar con la misma configuración para el método KNN:

- Configuración para el método KNN:
  - Píxeles de lado: 100
  - Radio del círculo:0
  - Grosor de línea: 2
  - Filtro: Gaussian Blur

En cambio, para el método SVM se han obtenido mejores resultados, por lo que la nueva configuración será:

- Configuración para el método SVM:
  - Pixeles de lado: 100
  - Radio del círculo:0
  - Grosor de línea: 2
  - Filtro: Gaussian Blur - Median Blur

5. Fase 5: Inserción de más muestras para la calibración

Tras realizar las pruebas anteriores y seleccionar el formato de imagen que nos da mejor resultado, se ha procedido a la inserción de más muestras para la calibración del sistema. En las pruebas anteriores se analizaba partiendo de 50-60 muestras por gesto. En esta ocasión se realizará el análisis con aproximadamente 300 muestras por gesto.

En la tabla D.41 observaremos que en el caso de KNN el resultado no varía, en cambio en el de SVM los resultado mejoran notablemente. Estos resultados muestran que si se disponen de pocas muestras para el entrenamiento el uso del método de clasificación “k-nearest neighbor” es más eficiente que con “Suport Vector Machine”. En cambio, si aumentamos el número de muestras el resultado es similar con ambos métodos.

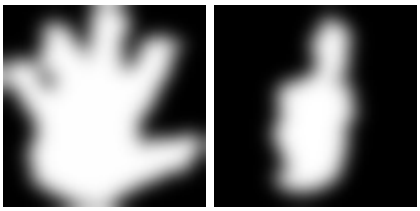


Figura D.38: Gestos reconocidos por el sistema: a la izquierda el gesto 1, Palma abierta, y a la derecha el gesto 2, dedo "ok".

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	0	4	0
1	0	48	8	1	0	46	9
2	0	8	34	2	0	8	34

Precisión obtenida mediante KNN = 0.79

LEFT	0	1	2	RIGHT	0	1	2
0	0	4	0	0	0	4	0
1	0	47	9	1	0	46	9
2	0	6	36	2	0	7	35

Precisión obtenida mediante SVM = 0.8

Tabla D.41: Resultados de las pruebas con 300 muestras por gesto.