Universidad del País Vasco

Facultad de Informática

Euskal Herriko Unibertsitatea

Informatika Fakultatea

TITULACIÓN: Ingeniería informática

Interfaz accesible persona-robot para un brazo articulado portátil

Alumno: D. Eduardo Navas Torres

Director: D. Julio Abascal González

Proyecto Fin de Carrera, Junio de 2013

Resumen

Este proyecto es un proyecto de fin de carrera para la consecución del título de ingeniería informática en la facultad de informática de la UPV/EHU en Donostia. El proyecto consiste en realizar una interfaz accesible para un robot asistencial llamado iArm. Este robot es un brazo robótico articulado con posibilidad de acoplamiento a una silla de ruedas eléctrica alimentado por su batería. De esta manera se pretende manipular objetos próximos facilitando así la vida diaria del usuario. La interfaz por defecto es un pequeño teclado de 16 botones, cuya utilización no es adecuada para la mayoría de las personas con discapacidad motora en los miembros superiores.
La interfaz que se propone desarrollar incluye una webcam para que el usuario tenga la posibilidad de elegir qué acciones quiere realizar mediante sencillos controles.

Índice

Capítulo 1 - Introducción

1.1 - Introducción

El origen del término robot data de 1921 donde Karel Capek lo utilizó por primera vez en 1921 en una obra teatral. Este término se deduce de la palabra polaca robotnik, que en castellano se traduce como obrero. La obra teatral relataba un ejército de máquinas con forma humana que fue creado para reemplazar al hombre en tareas y trabajos convirtiéndolos así en esclavos[1].

La realidad no dista mucho de aquella obra ya que a día de hoy los robots son creados para que trabajen bajo nuestras órdenes o para reemplazarnos en los trabajos más duros y peligrosos. Como por ejemplo la limpieza de una zona de accidente nuclear donde la exposición de los humanos a los gases tóxicos o zonas radioactivas son perjudiciales e incluso mortales. Aquí es donde los robots ganan relevancia.

De la misma manera que utilizamos los robots para que realicen trabajos peligrosos por nosotros, también se utilizan para dotar a personas discapacitadas de asistencia para realizar tareas que de otra manera no podrían llevar a cabo. Puede ser desde proporcionar una mano electromecánica con la que coger cosas o unas piernas biomecánicas para poder sustituir las piernas perdidas y así poder andar.

En ocasiones las personas necesitan de atención constante y ayuda en la realización de tareas diarias como puede ser el comer o moverse de una habitación a otra. En estos casos se disponen de diferentes robots adecuados para resolver la tarea, pero el problema reside en la interfaz de control de estos robots. Estas interfaces pueden no ser lo suficientemente accesibles como para que el usuario sea capaz de utilizarlas por lo que la solución no es válida.

Es necesario el desarrollo de interfaces accesibles adecuadas para la utilización de estos robots. En este sentido se ha realizado este proyecto: disponer de una interfaz adecuada, es decir accesible, para controlar el brazo robótico articulado iArm creado por la empresa holandesa Exact Dynamics[2].

1.2 - Objetivos iniciales del proyecto

El objetivo inicial es crear una interfaz accesible facilitando el uso del robot asistencial iArm creado por la empresa Exact Dynamics.

Para conseguir nuestro objetivo es necesario dividirlo en objetivos específicos, de esta manera permitirá hacer una revisión final a cada objetivo y así poder extraer las conclusiones oportunas. Los objetivos específicos definidos son los siguientes:

- Minimizar los desplazamientos necesarios para el control del brazo articulado.
- Dotar de algún tipo de visión al brazo.
- Disponer de algún dispositivo hardware con el que manipular el robot.

Para realizar los objetivos específicos se definen las siguientes tareas a realizar:

La primera tarea es unificar las operaciones básicas del robot en una misma ventana de tal manera que hagamos más fácil su utilización. Además hay que conseguir una comunicación adecuada mediante sincronización para evitar o solucionar posibles bloqueos del brazo.

A continuación se añadirán diferentes dispositivos de entrada para controlar el robot como por ejemplo un joystick con el que controlar la pinza, un deslizador para controlar la altura o un touchpad para mover la pinza hacia adelante y hacia atrás.

Otra tarea es la inclusión de diferentes webcam con la que el robot pueda distinguir y situar objetos en la escena para poder acercarlos al usuario. Para ello habrá que desarrollar alguna técnica de visión artificial de reconocimiento de objetos.

1.3 - Fases de desarrollo

Para el desarrollo de este proyecto se ha seguido el siguiente proceso:

- Decisión de entorno de desarrollo, lenguaje de programación y librería de visión artificial.
- Familiarización con el brazo articulado utilizando los programas de ejemplo proporcionados por el fabricante. Primeros programas de ejemplo controlando el robot mediante el teclado.
- Instalación de los recursos necesarios para la realización del proyecto como la librería de visión artificial de OpenCV, las librerías de Phidgets y preparación de Visual Studio 2010.
- Control del robot mediante joystick y elementos Phidgets. Posterior decisión de elementos para el uso final en la aplicación.
- Familiarización y primeras pruebas con la webcam y la librería OpenCV.
- Búsqueda de información y documentación de diferentes algoritmos de reconocimiento de objetos.
- Pruebas de los diferentes algoritmo de detección.
- Primera versión donde se comprueba la funcionalidad de todos los subprogramas en conjunto.
- Diseño final de la interfaz, mejoras del programa y limpieza de código.
- Pruebas finales y extracción de conclusiones.

1.4 - Estructura de la memoria

Para facilitar la lectura de este documento, se incluye a continuación una breve descripción de cada capítulo:

Capítulo 1 - Introducción

En este capítulo se detalla los principales objetivos del proyecto, las fases desarrolladas para la consecución del proyecto y un breve resumen de la estructura de la memoria.

Capítulo 2 - Estado del arte

En este apartado se describen las técnicas existentes para la consecución del proyecto. Para ello se ha realizado un estudio de cada campo, como consecuencia de este estudio se han obtenido las diferentes técnicas en el campo de la robótica, visión artificial e interfaces accesibles mostrando diferentes ejemplos de cada uno de ellos.

Capítulo 3 - Herramientas y plataforma

Se detallan cada una de las herramientas utilizadas tanto hardware como software. Mostrando por ejemplo las diferentes opciones que nos brinda la librería OpenCV o diferentes componentes Phidgets.

Capítulo 4 - Diseño

Apartado donde se muestran la descripción, los requisitos, el diagrama de secuencia y los posibles errores de cada función creada del sistema final desarrollado.

Capítulo 5 – Visión artificial

Se detallan las diferentes técnicas de Visión Artificial investigadas y desarrolladas en el transcurso del proyecto y los criterios utilizados para la selección de la más adecuada.

Capítulo 6 - Desarrollo

En este capítulo se expone el funcionamiento general del programa a través de las diferentes fases por las que ha pasado el proyecto.

Capítulo 7 - Gestión de proyecto

Este capítulo muestra las decisiones tomadas para el correcto desarrollo del proyecto en cuestiones de comunicaciones y control de versiones. Además se muestra una comparativa de las horas planificadas y las horas reales.

Capítulo 8 - Conclusiones y líneas futuras

Exposición de las conclusiones obtenidas a partir de los resultados obtenidos. Se detallan diferentes líneas futuras por las que podría derivarse este proyecto.

Apéndice 1: Estructura de datos y funciones de OpenCV

Listado de las estructuras de datos y funciones más utilizadas de la librería de visión artificial de OpenCV.

Apéndice 2: Funciones del robot iARM

Listado de funciones del robot iArm, así como una descripción y método de uso.

Apéndice 3: Herramientas y plataforma

Se exponen las diferentes herramientas utilizadas en el proyecto tanto Software como Hardware.

Capítulo 2 - Estado del arte

2.1 - Robótica

La Robótica hace unión de diferentes disciplinas para desarrollar robots capaces de realizar muchas labores humanas o simplemente facilitarlas. Dichas disciplinas incluyen la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física. Dentro del conjunto de robots existen un sinfín de formas de agrupar los robots, en este caso se explicarán los diferentes tipos existentes en base a la función que ejercen. Los diferentes tipos son según[3]:

- Robots industriales: Este tipo es en el que más robots se han desarrollado gracias a su utilidad dentro de la industria para realizar los trabajos más peligrosos, el primer robot industrial transportaba piezas fundidas en un molde hasta la cadena de montaje y soldaba estas partes sobre el chasis del vehículo.
- Robots para aplicaciones médicas y biológicas: Se utilizan para realizar operaciones quirúrgicas. Donde se destacan la telecirugía y la cirugía mínimamente invasiva.
- Robots domésticos: Están destinados a facilitar las labores domesticas como el robot Roomba que sirve para limpiar el suelo.
- Robots educativos: Sirven para acercar a los estudiantes las diferentes áreas de la robótica, un ejemplo puede ser el robot Mindstorm.
- Robots espaciales: Tienen como fin explorar superficies no accesibles para el humano, como por ejemplo los planetas o la luna. Para ello han de ser capaces de tomar decisiones para superar obstáculos y reponerse de posibles fallos. Como ejemplo se nombrará el robot Curiosity enviado por la NASA a Marte.
- Robots asistenciales: Están destinados a mejorar la calidad de vida de personas con alguna discapacidad. Este grupo incluye robots prostéticos, tales como brazos y piernas sustitutas, robots ortésicos o exoesqueletos, robots móviles que pueden ser sillas de ruedas inteligentes o robots manipuladores tal como el iArm.

2.1.1 - Robots asistenciales

Como ya se ha mencionado estos robots sirven para mejorar la calidad de vida de personas con alguna discapacidad, dentro de este conjunto de robots se pueden diferenciar los siguientes tipos[4]:

- Robots especializados en una aplicación concreta. Un ejemplo claro de la funcionalidad de este tipo de robots es la tarea de dar de comer. Por ejemplo el robot Handy que consiste en un brazo robótico muy sencillo que dispone de una cuchara como elemento terminal, el soporte para colocar la bandeja de comida y el soporte del vaso, que también se activa para acercarse a la boca del usuario. Una vez dispuesta la comida en la bandeja se dirige la cuchara a ella, después se acciona al robot para que acerque la comida a una posición cercana a la boca antes prefijada. Los intervalos de actuación son controlables por el usuario. Otra orden de accionamiento, desplaza el vaso hacia la boca y lo inclina para que el usuario pueda beber. El objetivo de este tipo de robots no es la de suplantar al cuidador, sino la de aumentar la autonomía del usuario dándole la oportunidad de comer y beber a su ritmo. El robot iArm posee la función de beber, inclinando el vaso lentamente para que pueda beber sin problemas.

- Robots manipuladores montados sobre sillas de ruedas. La posibilidad de montar el robot sobre la propia silla de ruedas confiere al usuario la capacidad de manipular los objetos de su entorno, además de desplazarse libremente en él. Un ejemplo de este tipo de robots es el brazo llamado Jaco, fabricado por la empresa canadiense Kinova[17]. Jaco es un brazo articulado resistente al agua que se acopla a cualquier tipo de silla de ruedas eléctrica. Cuenta con una mano con tres dedos para un mejor agarre, diseñado para ayudar a los usuarios en sillas de ruedas lograr una mayor autonomía. Además posee una longitud de un metro por lo que permite recoger cosas del suelo, del frigorífico o por ejemplo abrir una puerta.

- Robots con base móvil. El objetivo de este tipo de robots asistenciales es la de facilitar la vida laboral de personas discapacitadas aumentando el campo de actuación de dichas personas sin aumentar las dimensiones del robot utilizando guías para desplazarlo en un determinado entorno. En esta línea puede mencionarse el sistema *Raid[18]*, una estación

de trabajo basada en un computador y que cuenta con el soporte de un robot para acercar objetos al usuario o realizar operaciones del tipo, archivar documentos, pasar hojas de un libro, cargar un disquete en el computador... El robot se desplaza sobre unas guías vertical y horizontal para alcanzar todos los puntos de interés. El sistema es controlado por el usuario a través de un joystick, que además es utilizado para operar con el computador y para guiar la silla de ruedas. A través de un menú en pantalla, el usuario puede controlar un gran número de elementos y efectuar distintas operaciones en la estación de trabajo. Actualmente se esta empezando a disponer de este tipo de robots en entornos domésticos, pudiendo así transportar objetos de una habitación a otra. El campo de aplicación de estos robots es por tanto mucho mayor.

2.1.2 – iArm

El robot iArm creado por la empresa holandesa Exact Dynamics[2] es un brazo robótico articulado que posee 6 grados de libertad por lo que con la pinza que dispone puede alcanzar cualquier posición en el espacio y adoptar la orientación necesaria. La pinza es capaz de soportar hasta 1,5Kg y su velocidad máxima es de 9,9cm/s. Estas características están limitadas en comparación con otros brazos manipuladores ligeros. Estas limitaciones se deben a las aplicaciones especiales para las que ha sido creado que obligan a tener altos niveles de seguridad y control en el movimiento del brazo. Esta concebido para manipular objetos durante el desplazamiento del usuario con su silla y realizar otras operaciones, tales como abrir puertas.

Ilustración 1: Robot iArm

El iArm tiene un gran número de características técnicas de seguridad. A continuación se destacan algunas.

En caso de emergencia, por ejemplo ansiedad, el robot puede ser empujado sin miedo a dañarlo ya que está equipado con acoplamientos de deslizamiento. Se puede empujar cualquier parte del robot. Además, los acoplamientos de deslizamiento también protegen contra el daño en una colisión, por ejemplo contra una puerta. Después de una colisión o un empujón, se debe reiniciar el iArm para corregir los desajustes.

La mano se puede abrir manualmente, sin dañarla. Esto puede ser necesario para coger un objeto. Después de sacar el objeto de entre los dedos la pinza se cierra automáticamente.

2.2 - Visión por computador

La visión por computador o visión artificial analiza y extrae información útil sobre el mundo o entorno a partir de una imagen, un conjunto de imágenes o una secuencia de imágenes [5]. Dentro de la visión por computador se consideran siete etapas, aunque este número puede variar dependiendo del objetivo perseguido.

Las etapas a seguir en visión artificial son las siguientes:

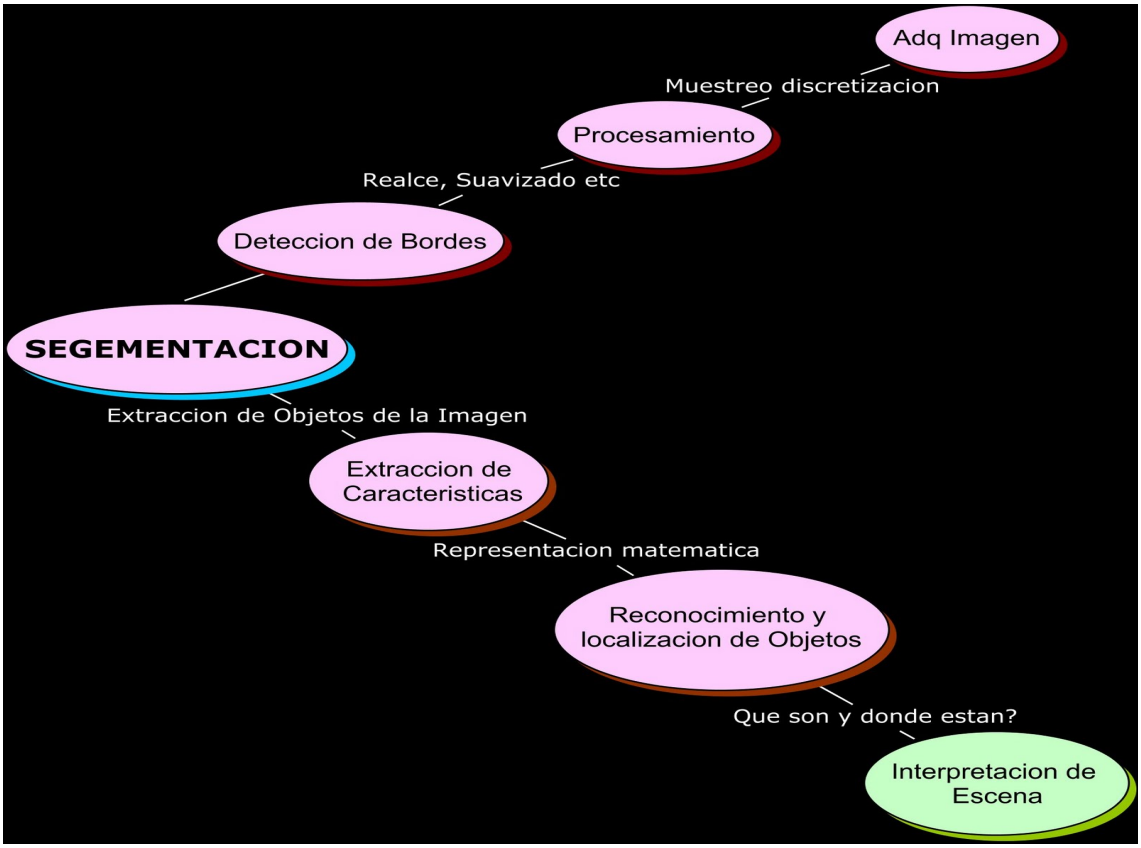


Ilustración 2: Etapas de un sistema de detección[6]

1. **Adquisición de la imagen**: En esta etapa se obtiene una imagen digital a partir del mundo real tridimensional mediante algún dispositivo de captura de imágenes digital.
2. **Pre-procesamiento**: Conjunto de técnicas que facilitan el procesamiento posterior, como son la eliminación de ruido y el realce.
3. **Detección de bordes**: En esta etapa se detectan los bordes de los objetos para poder diferenciarlos del fondo.

4. **Segmentación**: El objetivo de esta etapa es determinar en la imagen regiones cuyos píxeles tienen algún tipo de relación entre sí.
5. **Extracción de características**: Obtención de una representación matemática de los objetos previamente segmentados.
6. **Reconocimiento y localización de objetos**: En esta etapa se clasifican los objetos como pertenecientes a aquella clase cuyas características mas se asemejan. Después se procede a localizar el objeto usando técnicas de triangulación.
7. **Interpretación de la escena**: Con la información obtenida en las etapas anteriores se decide la acción a realizar.

Esta definición de etapas no significa que cualquier sistema de visión tenga necesariamente que pasar por cada una de ellas. Esta división se corresponde más con lo que podría denominarse un problema de reconocimiento e interpretación de la escena.

Además, dentro de la visión artificial podemos diferenciar tres entornos fácilmente distinguibles en los que debe trabajar un robot como son **entornos estructurados**, **entornos semi-estructurados** y **entornos desestructurados.** Los **entornos estructurados** son aquellos en los que el robot posee una descripción formal cuantitativa y exenta de ambigüedad como puede ser un mapa detallado con los posibles obstáculos. Los **entornos semi-estructurados** son aquellos en los que el robot posee una descripción formal pero las variables son incontrolables, como ejemplo puede ser un mapa detallado en el que no aparecen los posibles obstáculos ya que se desconocen a priori y el robot ha de detectarlos en tiempo real y evitarlos. Los **entornos desestructurados** son aquellos en los que se desconoce el entorno de trabajo, tratándose de un entorno incontrolable. Este tipo de entorno suele darse en navegación para exploración en las que el entorno es inaccesible para el ser humano como navegación espacial o en catástrofes en los que es peligrosa la exposición del ser humano.

Aunque se está evolucionando muy rápidamente el campo de visión artificial actualmente no provee una solución suficiente para **entornos desestructurados** ya que en ocasiones hay que tener en cuenta un número muy alto de factores. Además sería necesario una velocidad de

cómputo mayor que la actual para poder actuar en tiempo real.

En **entornos estructurados** la visión por computador es eficaz ya que los robots tienen una gran facilidad para realizar sus tareas y mejorar la calidad del producto final. Un ejemplo claro es en la industria en la que el robot tiene un entorno fijo y una tarea precisa que realizar, por tanto su espacio de trabajo está limitado y es inviolable.

Los **entornos semi-estructurados** generalmente se refiere a entornos domésticos en los que el robot ha de realizar diferentes tareas del hogar como por ejemplo aplicaciones de limpieza en los que se conoce el mapa pero no los posibles obstáculos por lo que las aplicaciones de visión por computador en este tipo de entornos pueden ser complejas, dependiendo de la tarea y de la naturaleza del entorno. Generalmente la visión artificial proporciona buenos resultados en estos entornos.

2.2.1 -Librerías de Visión por computador

OpenCV

OpenCV(Open Source Computer Vision)[7] es una librería de visión por computador desarrollada por Intel en 1999. Actualmente, esta librería se sigue desarrollando bajo la responsabilidad de la empresa Willow Garage. Esta librería aporta un amplio abanico de funciones para procesamiento de imágenes en tiempo real y una de sus principales ventajas es que puede ejecutarse en ordenadores sin grandes prestaciones, pudiendo llegar a conseguir aplicaciones sofisticadas para un amplio conjunto de usuarios.

Ilustración 3: Logotipo de la librería de OpenCV

Esta librería fue inicialmente escrita en C, pero a lo largo de los años se ha desarrollado en diferentes lenguajes como C++, Python, Ruby y Java para conseguir un público más amplio.

Una de las principales ventajas de OpenCV es su versatilidad y portabilidad de unos sistemas a otros, ademas posee licencia de código abierto BSD y su uso es gratuito. Es compatible con la mayoría de sistemas operativos como Windows, MacOS, Linux, FreeBSD, Maemo, OpenBSD e incluso de telefonía como Android e iOS.

En el momento de la realización de esta memoria la versión más reciente es la 2.4.4, publicada el doce de febrero de dos mil trece.

Librería IVT

La librería IVT(Integrating Vision Tollkit)[8] es una potente y rápida librería dedicada a la visión por computador escrita en C++, fácil de usar y su arquitectura es orientada a objetos. Además cuenta con su propia herramienta multiplataforma GUI. Fue desarrollada por la anteriormente llamada Universidad de Karlsruhe, ahora denominada Instituto de Tecnología de Karlsruhe. La primera versión de esta librería fue liberada el 22 de diciembre de 2005.

Esta librería esta disponible como software libre bajo una licencia BSD de tres clausulas. Es multiplataforma y funciona en prácticamente cualquier plataforma que posea un compilador de C++ como Windows, MacOS o Linux.

El paquete de software incluye clases para la captura de imágenes, creación de hilos y algoritmos de procesamiento de imagen. Los formatos internos de imagen son RGB y escala de grises.

AFORGENET

Aforgenet[9] es una librería de visión artificial originalmente desarrollada por Andrew Kirillov para el marco .NET. El código fuente y los binarios están disponibles bajo licencia Lesser GPL. Esta programada en C++ y es compatible exclusivamente con Windows.

Esta librería incluye soporte para procesamiento de imágenes y de video, redes neuronales, programación genética, lógica difusa, machine learning. Además posee bibliotecas para un selecto grupo de robots como Lego Mindstorms, RCX o Surveyor.

2.2.2 Comparativa

Para poder decidir qué librería elegir entre las presentadas anteriormente se ha de estudiar cuál de ellas permite realizar las técnicas de detección que se exponen en el Capítulo 4: Visión Artificial. Para visualizarlo más fácil se define la siguiente tabla que muestra la relación de técnicas y librerías compatibles.

	Segmentación	Transformada de Hough	Sift & SURF	Haar	Blobs
OpenCV	Si	Si	SI	Si	Si
IVT	Si	Si	Si	No	No
AForgeNET	Si	Si	No	No	Si

En la tabla se aprecia que la única librería que es compatible con todas las técnicas que se van a desarrollar es OpenCV. Por lo que será la librería elegida en este proyecto. Además OpenCV tiene una comunidad de usuarios muy extensa, lo que será útil en la búsqueda de tutoriales, ejemplos o soluciones de problemas.

2.3 - Interfaces accesibles

Existen diferentes maneras para interactuar con un ordenador pero el mundo de las interfaces accesibles es más complejo ya que cada persona discapacitada tiene diferentes limitaciones dependiendo de la capacidad cognitiva, características sensoriales o destreza motora. Por lo que se puede decir que para cada persona discapacitada requiere de una interfaz única, en consecuencia no se puede generalizar.

Podemos encontrar en el estudio de Katherine Tsui realizado en 2008[10] sobre interfaces accesibles los resultados de diferentes pruebas realizadas con el usuario final. En el estudio se muestran diferentes interfaces desarrolladas y los resultados de cada una teniendo como participantes a un número amplio de personas con diferentes discapacidades.

2.3.1 - Interfaz del iArm

El brazo robótico articulado iArm tiene como dispositivo de entrada un teclado alfanumérico de 16 botones. Mediante este teclado el usuario puede interactuar con robot navegando por los distintos menús que dispone. Para conocer en que menú nos encontramos dispone de un display alfanumérico en el hombro del robot, esta pantalla muestra un carácter identificando el menú. Dependiendo que menú estemos los botones tienen una funcionalidad u otra. Por lo que se intuye que el grado de accesibilidad de esta interfaz no es muy alta, ya que la capacidad cognitiva debe ser alta, además la pulsación de cada botón no es accesible para personas con discapacidades motoras. Para navegar por los menús es necesario memorizar las posibilidades que nos brinda la aplicación o llevar el mapa de menús con el que el fabricante nos proporciona como parte del manual de instrucciones.

2.3.2 - Phidgets

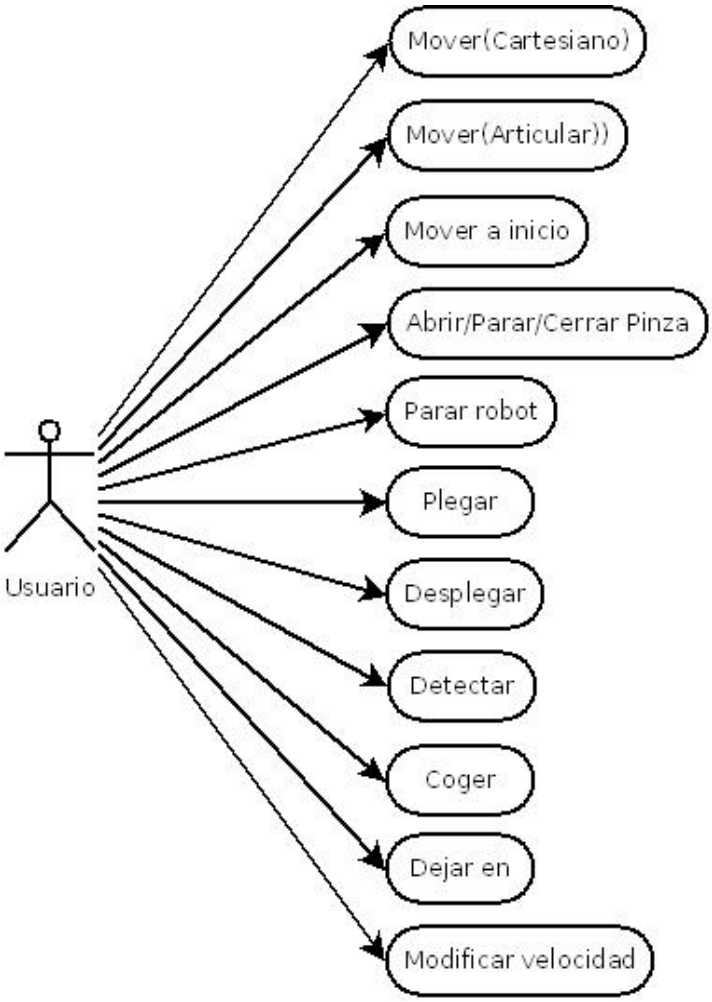
Los Phidgets[11] son dispositivos hardware de bajo coste, por ejemplo minijoysticks, deslizadores, sensores de distancia, sensores de fuerza, sensores ambientales, servos, servomotores, etc. que nos brindan una amplísima gama de productos para interacción. Además cuenta con una API muy extensa con lo que da la posibilidad de adaptarlos a nuestras aplicaciones fácilmente.

Su uso se centra principalmente en la exploración de alternativas de interfaces hardware para facilitar la interacción con cualquier sistema. Gracias a su facilidad de conexión con el ordenador y a la gama de productos tan extensa que posee podemos adaptarlos para crear interfaces accesibles haciendo uso de los diferentes joysticks, deslizadores y demás.

Capítulo 3 - Diseño

En este capítulo mostramos el diseño realizado para las distintas funciones del robot implementadas.

Modelo de casos de uso



Vemos en la ilustración 13 la relación de funciones desarrolladas en la aplicación. Cada función es un caso de uso y cada una es detallada a continuación.

Mover(Cartesiano)

Descripción

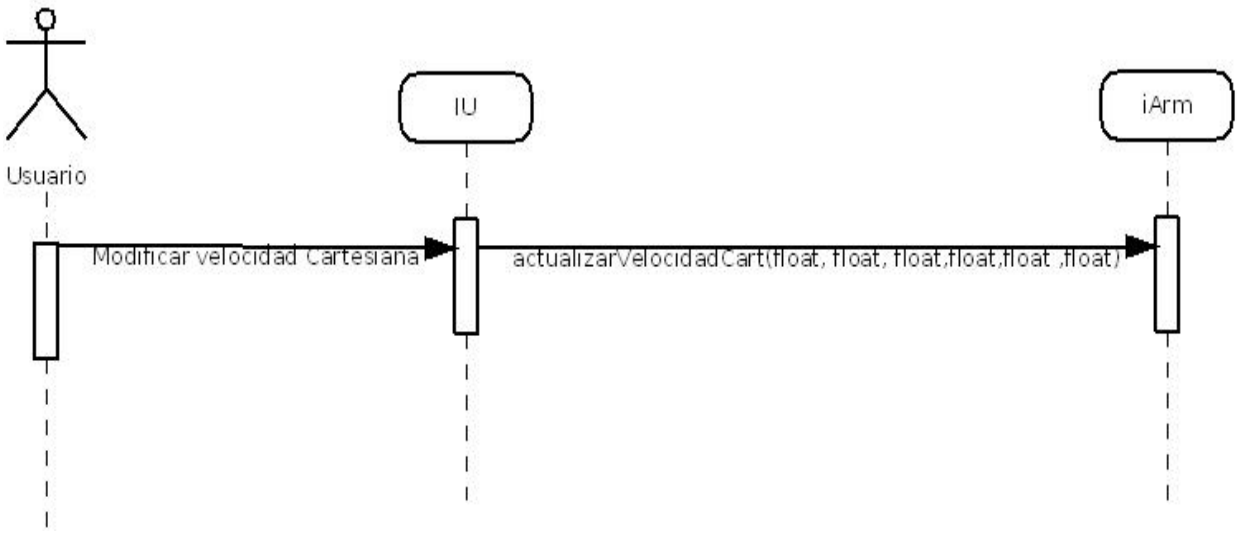
El usuario modifica la posición del manipulador en alguno de sus ejes mediante Phidgets o la interfaz gráfica, el programa del iArm actualiza estos cambios de posición en el robot.

Requisitos

✔ El robot tiene que estar encendido y conectado.

✔ El robot no puede estar plegado.

Diagrama de secuencia



Posibles errores

- El robot alcanza su mayor extensión y emite un sonido.

  Solución: Detener todo movimiento desde el botón de parar. Si el movimiento viene desde los elementos Phidgets, volver a la posición de parada en los elementos Phidgets. Si tras hecho esto no permite ningún movimiento cartesiano mover a posición de inicio.

- La pinza del robot se ha atascado.

  Solución: Detener todo movimiento desde el botón de parar. Si el movimiento viene

desde los elementos Phidgets, volver a la posición de parada en los elementos Phidgets. Si tras hecho esto no permite ningún movimiento cartesiano mover a posición de inicio.

- El robot se choca consigo mismo.

  Solución: Parar por completo todo movimiento del robot cuanto antes.

Mover(Articular)

Descripción

El usuario la posición de alguna articulación mediante la interfaz gráfica.

Requisitos

✔ El robot tiene que estar encendido y conectado.

✔ El robot no puede estar plegado.

Diagrama de secuencia

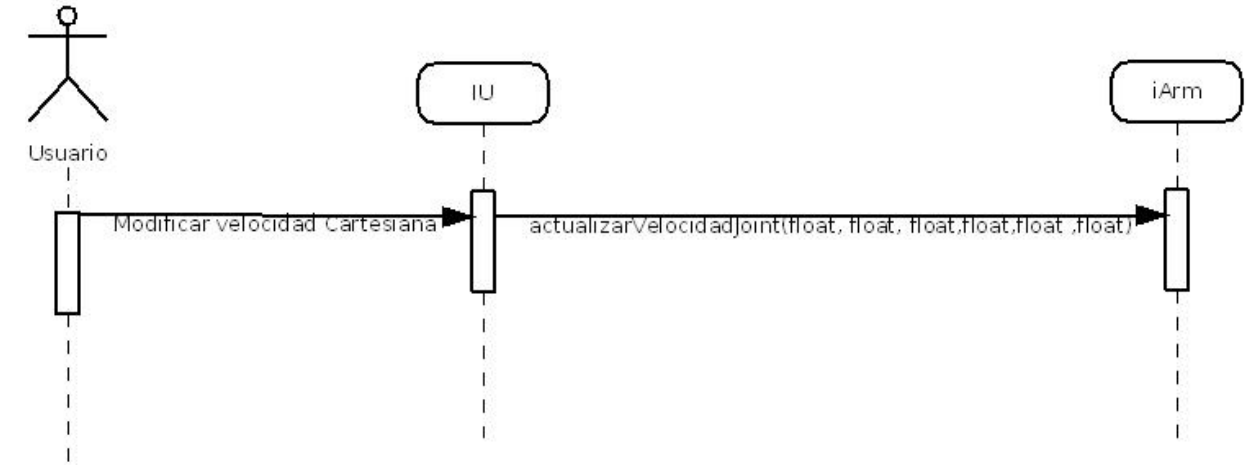


Ilustración 10: Caso de uso: Mover(Articular)

Posibles errores

- El robot alcanza su mayor extensión o la pinza se atasca.

  Solución: Realizar el movimiento contrario al que ha provocado el error.

- El robot se choca consigo mismo.

  Solución: Parar por completo todo movimiento del robot cuanto antes.

Mover a inicio

Descripción

El usuario indica al robot que sitúe la pinza en la posición de inicio, ya sea para volver al estado inicial o para solventar algún tipo de error en alguna articulación como atascamiento de articulación o extensión del robot.

Requisitos

✔ El robot tiene que estar encendido y conectado.

✔ El robot no puede estar plegado.

Diagrama de secuencia

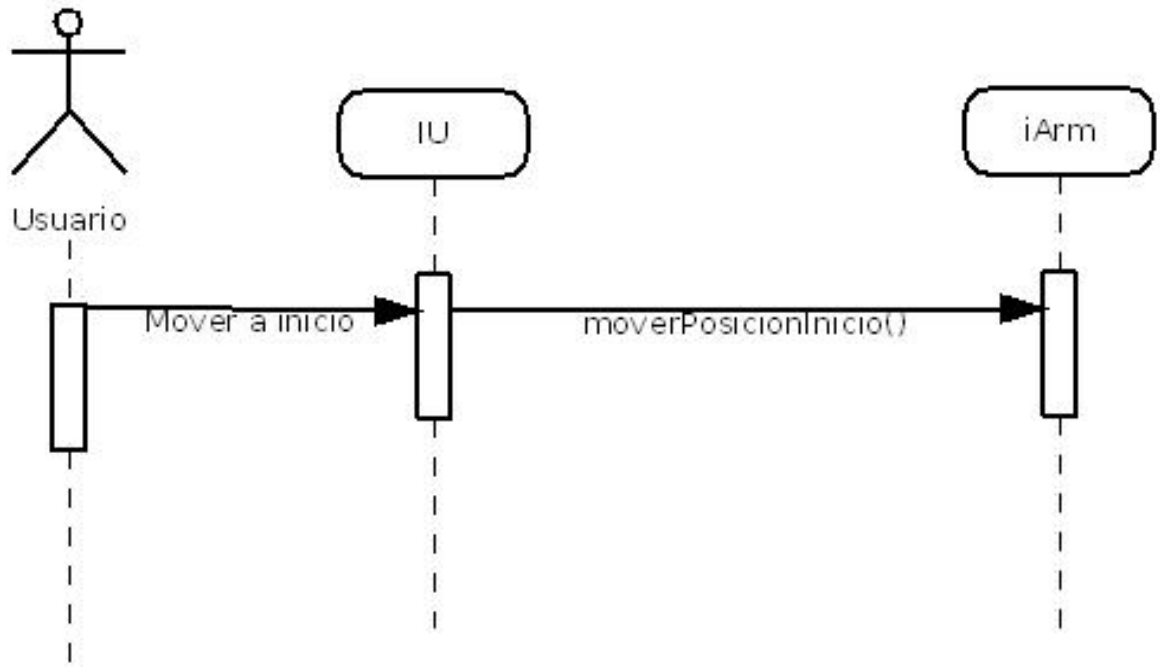


Ilustración 11: Caso de uso: Mover a inicio

Posibles errores

- No se contemplan errores.

Abrir/Parar/Cerrar Pinza

Descripción

El usuario abre, cierra la pinza o para el movimiento de la pinza.

Requisitos

✔ El robot tiene que estar encendido y conectado.

✔ El robot no puede estar plegado.

Diagrama de secuencia

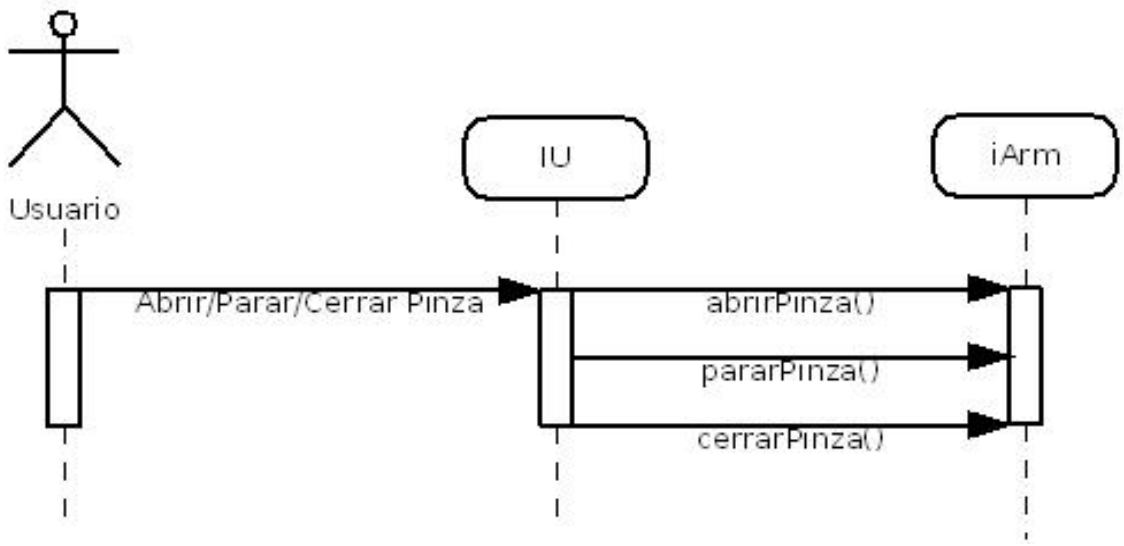


Ilustración 12: Caso de uso: Abrir/Parar/Cerrar Pinza

Posibles errores

- No se contemplan errores.

Parar robot

Descripción

El usuario para por completo todo tipo de movimientos.

Requisitos

✔ El robot tiene que estar encendido y conectado.

✔ El robot no puede estar plegado.

Diagrama de secuencia

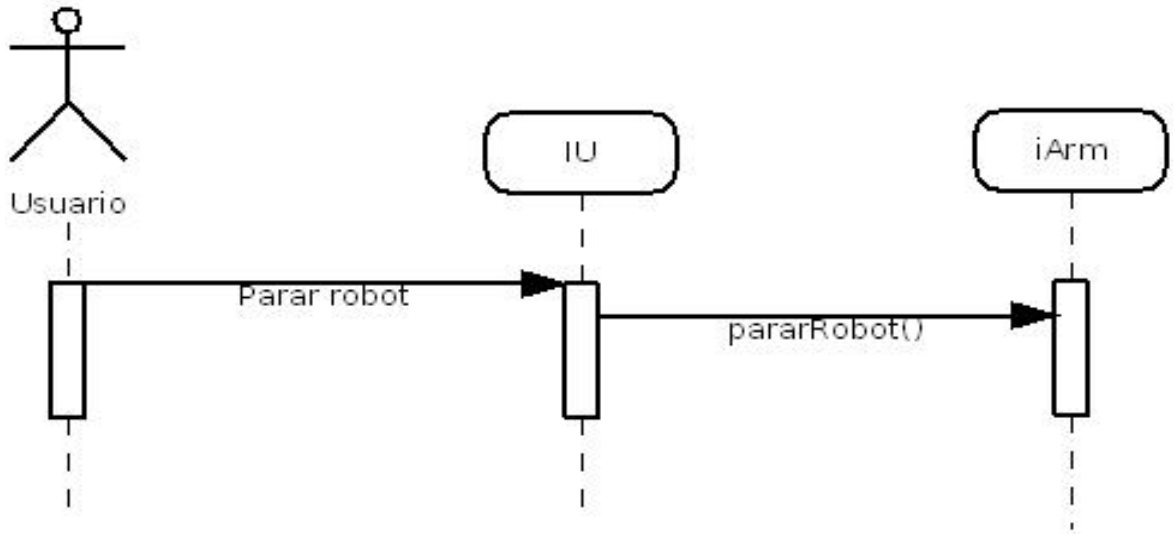


Ilustración 13: Caso de uso: Parar robot

Posibles errores

No se contemplan errores.

Plegar

Descripción

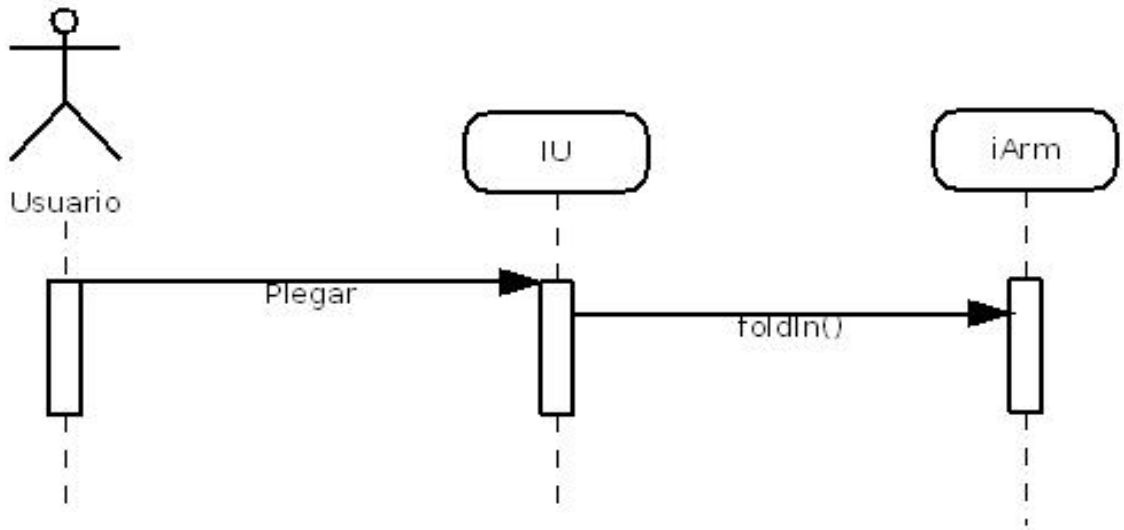
El usuario quiere terminar la utilización del robot por lo que pliega el robot.

Requisitos

✔ El robot tiene que estar encendido y conectado.

✔ El robot no puede estar plegado.

Diagrama de secuencia



Posibles errores

- El robot no está en una posición desde la que pueda plegarse.

  Solución: Mover el robot a la posición de inicio y luego volver a intentar plegar el robot.

Desplegar

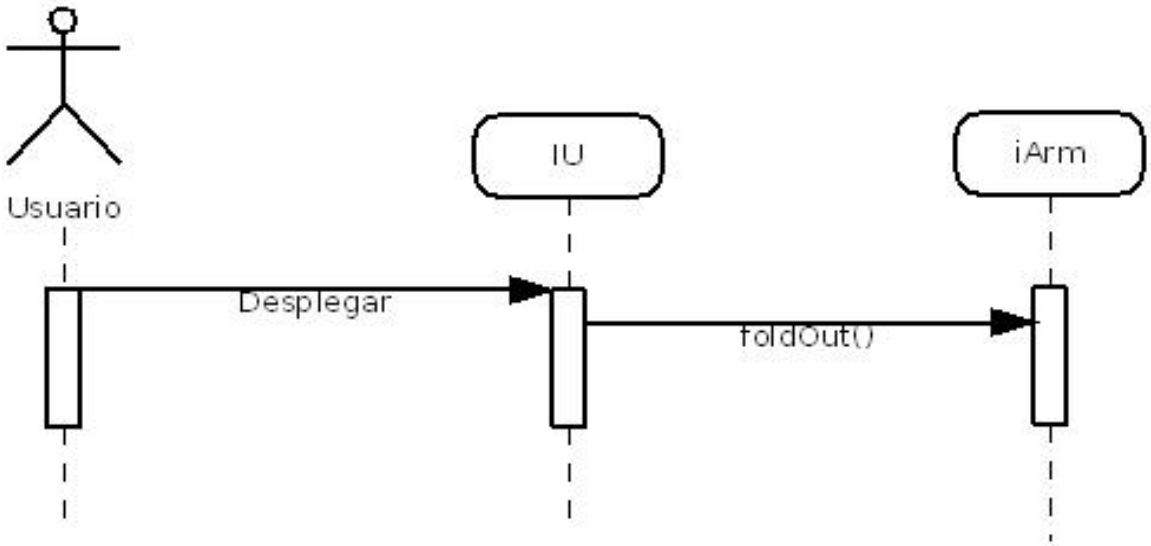
Descripción

El usuario quiere comenzar a utilizar el robot por lo que indica el despliegue.

Requisitos

✔ El robot tiene que estar encendido y conectado.

Diagrama de secuencia



Posibles errores

- No se contemplan errores.

Detectar

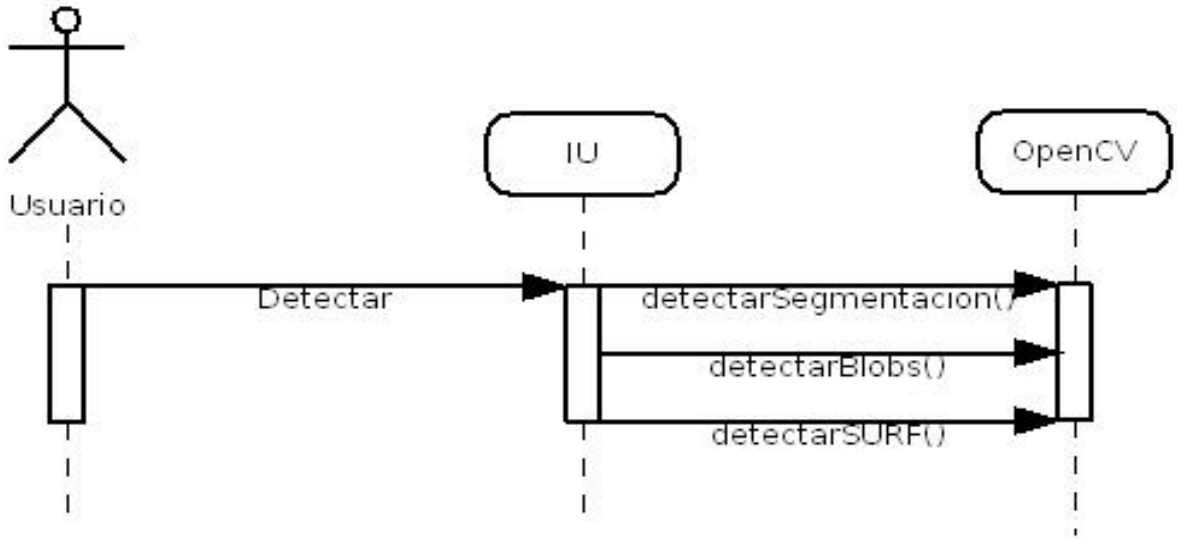
Descripción

El usuario activa la detección de objetos de la interfaz.

Requisitos

-

Diagrama de secuencia



Posibles errores

- La webcam no detecta ningún objeto.

  Solución: Cambiar las condiciones de iluminación.

- El reconocimiento mediante Blobs provoca demasiados falsos positivos.

  Solución 1: Cambiar las condiciones de iluminación.

  Solución 2: Renovar la imagen de fondo.

Coger

Descripción

El usuario coge el objeto detectado por la webcam.

Requisitos

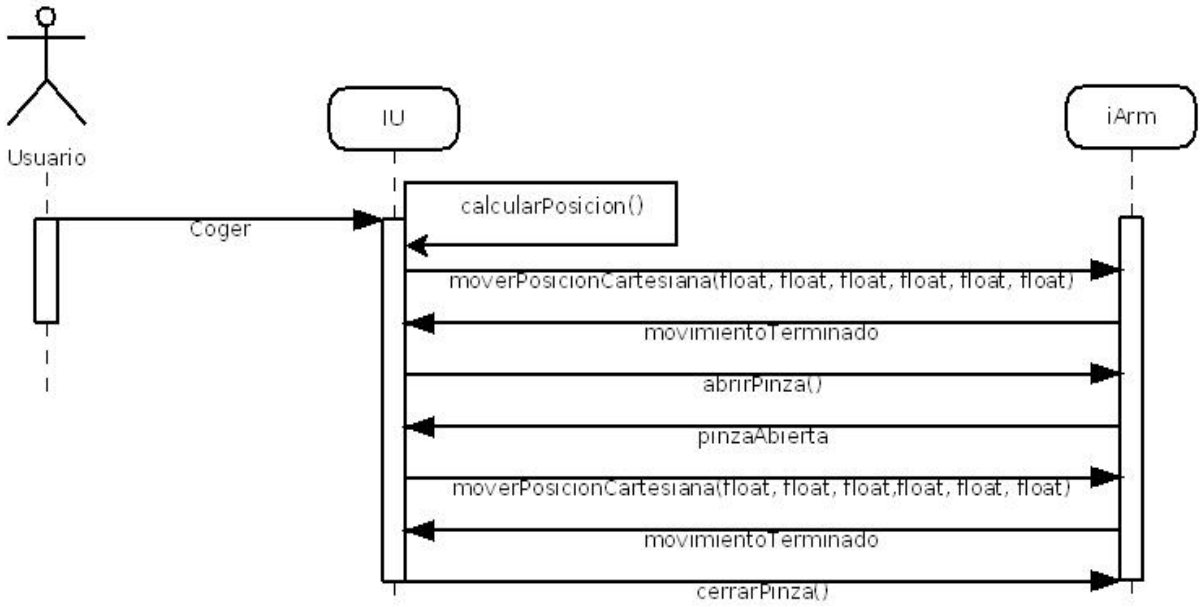
✔ El robot tiene que estar encendido y conectado.

✔ El robot no puede estar plegado.

✔ La detección ha de estar activada.

✔ El objeto tiene que estar detectado.

Diagrama de secuencia



Posibles errores

- La webcam no detecta la pelota suficiente tiempo como para calcular la posición.

  Solución: Modificar las condiciones de iluminación.

Dejar en

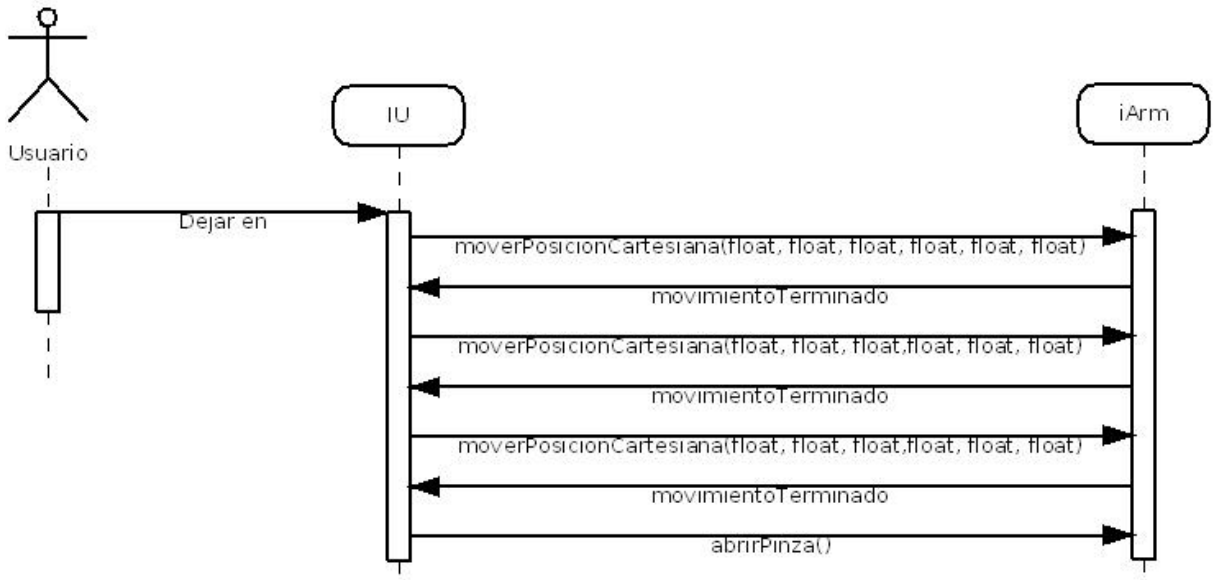
Descripción

El usuario quiere dejar el objeto previamente cogido en una posición de la cuadrícula.

Requisitos

- ✔ El robot tiene que estar encendido y conectado.
- ✔ El robot no puede estar plegado.
- ✔ El robot ha de tener el objeto en la pinza.

Diagrama de secuencia



Posibles errores

- No se contemplan errores.

Modificar Velocidad

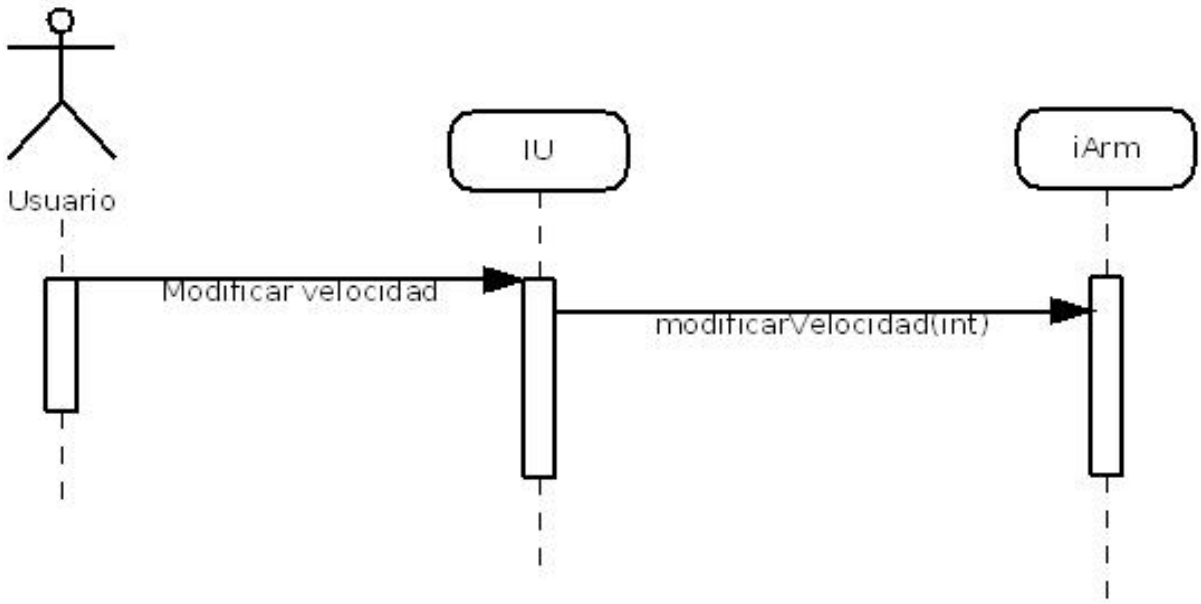
Descripción

El usuario aumenta o disminuye la velocidad de los movimientos cartesianos.

Requisitos

- ✔ El robot tiene que estar encendido y conectado.
- ✔ El robot no puede estar plegado.

Diagrama de secuencia



Posibles errores

- No se contemplan errores.

Capítulo 4 - Visión artificial

4.1 - Introducción

La visión artificial está muy ligada a la robótica dado a que es una buena forma de que un robot pueda interactuar con el entorno autónomamente. En nuestro proyecto necesitamos de esta interacción autónoma por lo que nos vemos en la necesidad de integrar algún tipo de sistema de detección.

Nuestro proyecto necesita un sistema capaz de detectar diferentes objetos y, tras seleccionar por parte del usuario qué objeto desea alcanzar, calcular su posición transmitiéndola al robot para que este proceda a su recogida.

En este proyecto se han desarrollado diferentes programas de visión artificial para comprobar la viabilidad de diferentes técnicas. Para ello se ha utilizado la librería OpenCV, ya mencionada en el capítulo de software utilizado. A continuación se detalla cada aproximación realizada.

4.2 - Clasificador Haar

El clasificador Haar es un método desarrollado por Viola y Jones y es una versión del algoritmo “Adaboost”[12]. Es un clasificador basado en árboles de decisión con entrenamiento supervisado. El entrenamiento se realiza determinando una serie de características basadas en sumas y restas de los niveles de intensidad de la imagen. Basándose en estas características locales se puede obtener un detector de objetos robusto. También se denominan estos clasificadores mediante el nombre de cascada, ya que el resultado del clasificador es el fruto de varios clasificadores más simples o etapas. El candidato a objeto dentro de la imagen a procesar debe superar todas las etapas para ser aceptado.

AdaBoost se emplea para encontrar reglas de clasificación altamente precisas (fuertes) combinando muchas reglas de baja precisión (débiles), es decir que se trata de una combinación lineal de clasificadores débiles para obtener clasificadores fuertes.

Para obtener el clasificador se utilizan cientos o incluso miles de imágenes de un objeto en particular. Se generan los casos positivos y éstos se escalan al mismo tamaño. Además se necesitan

imágenes arbitrarias del mismo tamaño donde no aparezca el objeto a clasificar, estas imágenes se utilizaran como casos negativos.

OpenCV nos da la posibilidad de utilizar este método de detección. Para ello propone diferentes clasificadores ya entrenados, entre ellos un clasificador de caras o un clasificador de ojos. En nuestro caso necesitamos clasificadores propios para poder detectar diferentes objetos por lo que utilizamos una herramienta proporcionada por OpenCV llamada opencv_haartraining para desarrollar diferentes clasificadores usando nuestras imágenes. Para probar estos clasificadores también propone una aplicación que evalúa el clasificador denominada opencv_performance, esta aplicación evalúa la calidad del clasificador utilizando diferentes imágenes positivas.

En este caso se ha procedido a clasificar una taza de desayuno. En total se han entrenado doce clasificadores para conseguir el mejor resultado posible ya que existen muchas variables a tener en cuenta a la hora de hacer el clasificador. A continuación se detallan los diferentes clasificadores creados aclarando qué atributos permite modificar la herramienta de entrenamiento, los valores que hemos definido y los resultados obtenidos.

En primer lugar necesitamos diferentes imágenes tanto positivas como negativas para poder clasificar la taza de desayuno. Para conseguir las imágenes negativas se ha creado un programa que dado un vídeo cualquiera obtiene un número muy alto de imágenes que necesitemos, para que las imágenes sean suficientemente diferenciables se obtiene un fotograma cada diez o veinte fotogramas. En mi caso he utilizado un vídeo casero y otro archivo de vídeo de una película.

Para las imágenes positivas tenemos dos opciones. La primera es hacer nosotros mismos las imágenes del objeto a clasificar teniendo en cuenta que es necesario variar la intensidad de la luz, el ángulo del objeto en la imagen y el tamaño del objeto. La segunda opción es usar la herramienta de OpenCV llamada opencv_createsamples.

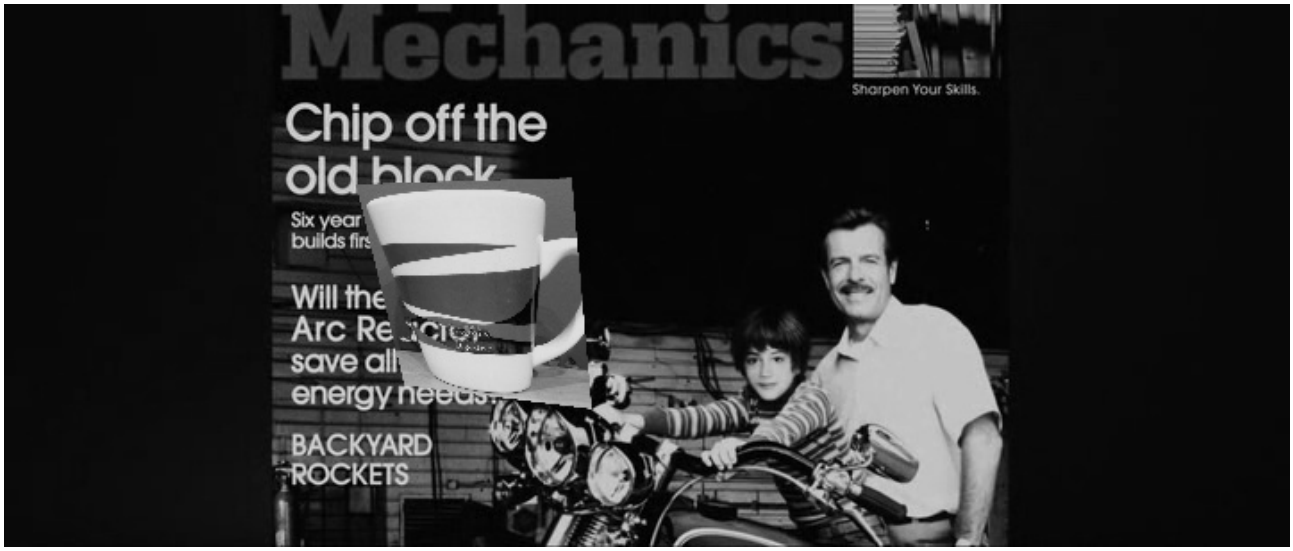
CREATESAMPLES

Esta herramienta genera imágenes positivas a partir de una sola, creando diferentes variaciones de ángulos para obtener diferentes tomas del objeto, variaciones de intensidad en el color del objeto y después las incrusta en la imágenes negativas. Por lo que el numero máximo de imágenes positivas que podemos crear viene dado por el tamaño de nuestra base de datos de imágenes negativas. Las características de cada atributo de esta herramienta son:

- bgcolor: Este atributo nos permite declarar si queremos que las muestras sean en color, el valor equivalente es uno, o no, el valor equivalente es 0. Según la documentación la elección de imágenes en color solo introduce ruido en las muestras, por lo que definimos a cero en todos los casos.
- Background color threshold: Define el valor umbral de fondo. Los valores comprendidos entre bgcolor y Bg Color threshold serán considerados transparentes.
- Max Intensity Deviation: El valor de la desviación máxima de la intensidad de los píxeles del primer plano.
- Max X rotation angle: Valor del ángulo máximo de la rotación en el eje X.
- Max Y rotation angle: Valor del ángulo máximo de la rotación en el eje Y.

- Max Z rotation angle: Valor del ángulo máximo de la rotación en el eje Z.
- Ancho: Ancho de las muestras de salida expresado en píxeles.
- Alto: Alto de las muestras de salida expresado en píxeles.
- Vec: Nombre del archivo de salida.
- Img: Nombre del archivo de imagen que contiene la imagen de muestra positiva.

Dado que es necesario obtener un número alto de imágenes positivas, la obtención manual de éstas se hace un trabajo demasiado extenso por lo que se decide utilizar la herramienta de OpenCV.



En este caso hemos hecho 3 fotos desde distintos ángulos a la taza y a partir de estas tres vistas hemos desarrollado las imágenes positivas. De esta manera podemos detectar la taza en distintas posiciones. En los seis primeros entrenamientos no se han modificado los atributos de createsamples, de tal manera que se hace un entrenamiento por defecto, los cambios han sido en la cantidad de imágenes utilizadas.

A continuación se muestra una relación de los atributos con los valores que se han utilizado en los seis primeros casos.

	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
Imagenes positivas	600	600	150	6000	6000	1500
Imagenes negativas	600	100	600	6000	1000	6000
Bg color	0	0	0	0	0	0
Background color threshold	80	80	80	80	80	80
Max intensity deviation	40	40	40	40	40	40
Max x rotation angle	1,1	1,1	1,1	1,1	1,1	1,1
Max y rotation angle	1,1	1,1	1,1	1,1	1,1	1,1
Max z rotation angle	0,5	0,5	0,5	0,5	0,5	0,5
ancho	24	24	24	24	24	24
alto	24	24	24	24	24	24

En los seis siguientes entrenamientos se han definido todos los atributos a cero. Lo que hacemos es incrustar la muestra positiva directamente en las imágenes negativas creando así la imágenes positivas. Al igual que en los seis primeros entrenamientos los cambios entre entrenamientos reside en la cantidad de imágenes que se utilizan.

La siguiente tabla muestra los valores mencionados:

	Caso 7	Caso 8	Caso 9	Caso 10	Caso 11	Caso 12
Imagenes positivas	600	600	150	6000	6000	1500
Imagenes negativas	600	100	600	6000	1000	6000
Bg color	0	0	0	0	0	0
Background color threshold	0	0	0	0	0	0
Max intensity deviation	40	40	40	40	40	40
Max x rotation angle	0	0	0	0	0	0
Max y rotation angle	0	0	0	0	0	0
Max z rotation angle	0	0	0	0	0	0
ancho	24	24	24	24	24	24
alto	24	24	24	24	24	24

Una vez terminada la aplicación tenemos todas las imágenes disponibles y preparadas para

el entrenamiento.

Haar-Training

Una vez obtenidas las imágenes positivas y negativas procedemos a entrenar el clasificador. Para ello utilizamos la herramienta opencv_haartraining. Los atributos modificables de esta herramienta son los siguientes.

- **data**: Nombre del directorio en que se almacena el clasificador entrenado.
- **Vec**: Nombre del archivo generado en opencv_createsamples que contiene las muestras positivas.
- **Bg**: Archivo de texto que contiene la relación de imágenes negativas.
- **Npos**: Número de imágenes positivas a utilizar en el entrenamiento. Valor recomendado: 5000.
- **Nneg**: Número de imágenes negativas a utilizar en el entrenamiento. Valor recomendado: 6000.
- **Nstages**: Número de etapas de entrenamiento.
- **Nsplits**: Determina la debilidad del clasificador. Se aconseja definirlo a dos para conseguir un clasificador más potente.
- **Mem**: Memoria disponible en MB para realizar cálculos previos, cuanta más memoria más rápido obtendremos nuestro clasificador.
- **Nonsym**: Especifica si el objeto de la imagen a clasificar tiene simetría vertical o no.
- **Minhitrate**: Valor mínimo de éxito deseado para cada etapa del clasificador .
- **Maxfalsealarm**: Valor máximo deseado de la tasa de falsos positivos para cada etapa del clasificador.
- **Boosting algoritm**: Algoritmo de creación de clasificadores.
- **Error**: Algoritmo para detectar errores.
- **Required false alarm**: Valor generado a partir de los valores anteriores. Define el valor requerido de falsos positivos.
- **Ancho**: Ancho de las muestras, este valor ha de ser el mismo que el definido en createsamples.
- **Alto**: Alto de las muestras, este valor ha de ser el mismo que el definido en createsamples.

En el caso del entrenamiento se ha realizado con los parámetros por defecto. La siguiente tabla muestra la relación de los valores por cada atributo de los 6 primeros casos.

	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
Stages	14	14	14	14	14	14
Splits	1	1	1	1	1	1
Imagenes positivas	500	500	100	5000	5000	1000
Imagenes negativas	600	100	600	6000	1000	6000
Min hit rate	0,995	0,995	0,995	0,995	0,995	0,995
Maxfalsealarm	0,5	0,5	0,5	0,5	0,5	0,5
Trimming	0,95	0,95	0,95	0,95	0,95	0,95
Boosting algoritm	GAB	GAB	GAB	GAB	GAB	GAB
Error	Misclass	Misclass	Misclass	Misclass	Misclass	Misclass
Required false alarm	6,1035e-05	6,1035e-05	6,1035e-05	6,1035e-05	6,1035e-05	6,1035e-05

La siguiente tabla muestra la relación de los valores de cada atributo utilizado, en estos casos se ha utilizado de nuevo los valores por defecto exceptuando 'minhitrate', ya que hemos aumentado este valor para obtener un clasificador mejor.

	Caso 7	Caso 8	Caso 9	Caso 10	Caso 11	Caso 12
Stages	14	14	14	14	14	14
Splits	1	1	1	1	1	1
Imagenes positivas	500	500	100	5000	5000	1000
Imagenes negativas	600	100	600	6000	1000	6000
Min hit rate	0,999	0,999	0,999	0,999	0,999	0,999
Maxfalsealarm	0,5	0,5	0,5	0,5	0,5	0,5
Trimming	0,95	0,95	0,95	0,95	0,95	0,95
Boosting algoritm	GAB	GAB	GAB	GAB	GAB	GAB
Error	Misclass	Misclass	Misclass	Misclass	Misclass	Misclass
Required false alarm	6,1035e-05	6,1035e-05	6,1035e-05	6,1035e-05	6,1035e-05	6,1035e-05
False rate	2,34309e-06	6,1035e-05	No conseguid	3,6644e-05	6,47781e-07	3,95126e-6

			o			

Ya hemos conseguido los diferentes clasificadores con los diferentes atributos, es hora de evaluar su eficacia.

Performance

Para probar el clasificador obtenido utilizamos la herramienta propuesta por OpenCV llamada opencv_performance. Esta herramienta prueba el clasificador con imágenes de prueba. En este caso se han realizado 50 imágenes de test de cada ángulo de la taza, por lo que se dispone de 150 imágenes para testear nuestro clasificador.

A continuación se muestran los resultados, las columnas vacías indican que no se consiguió obtener un clasificador válido:

		Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6	Caso 7	Caso 8	Caso 9	Caso 10	Caso 11	Caso 12
Aciertos	Muestra 1	40	34	29	37		34	0	0		0	0	0
	Muestra 2	35	40	24	40		26	0	0		3	0	3
	Muestra 3	42	42	20	41		22	0	1		7	3	3
Fallos	Muestra 1	10	16	21	13		16	50	50		50	50	50
	Muestra 2	15	10	26	10		24	50	50		47	50	47
	Muestra 3	8	8	30	9		28	50	49		43	47	47
Falsos	Muestra 1	85	1019	126	53		120	152	1		62	270	31
	Muestra 2	103	1568	128	62		111	137	3		128	203	35
	Muestra 3	72	1239	140	70		99	120	1		67	241	25
Totales	Aciertos	117	116	73	118		82	0	1		10	3	6
	Fallos	33	34	77	32		68	150	149		140	147	144
	Falsos	260	3826	394	185		330	409	5		257	714	91

Tal y como se refleja en la tabla, el mejor de los casos el clasificador ha detectado la taza de desayuno en 118 imágenes de test de 150 del total, por lo que conseguimos una efectividad del 78,6%. El porcentaje es elevado pero también ha detectado un número demasiado elevado de falsos positivos, en total 185. Por lo que no es una opción válida para nuestro proyecto, ya que la detección ha de ser precisa.

Dado que este tipo de clasificadores mejora con la cantidad de imágenes utilizadas se decide realizar dos clasificadores más aumentando significativamente la cantidad de imágenes tanto positivas como negativas. Las diferencias en estos dos últimos casos están en la utilización de valores adecuados para los atributos del entrenamiento y en el aumento de imágenes tanto positivas como negativas.

A continuación se detallan los datos de createsamples:

	Caso 1	Caso 2
Imagenes positivas	3700*3	1850*3
Imagenes negativas	16000	8000
Bg color	0	0
Background color threshold	0	0
Max intensity deviation	100	100
Max x rotation angle	0,8	0,8
Max y rotation angle	0,8	0,8
Max z rotation angle	0,3	0,3
ancho	24	24
alto	24	24

A continuación se detallan los datos de haartraining:

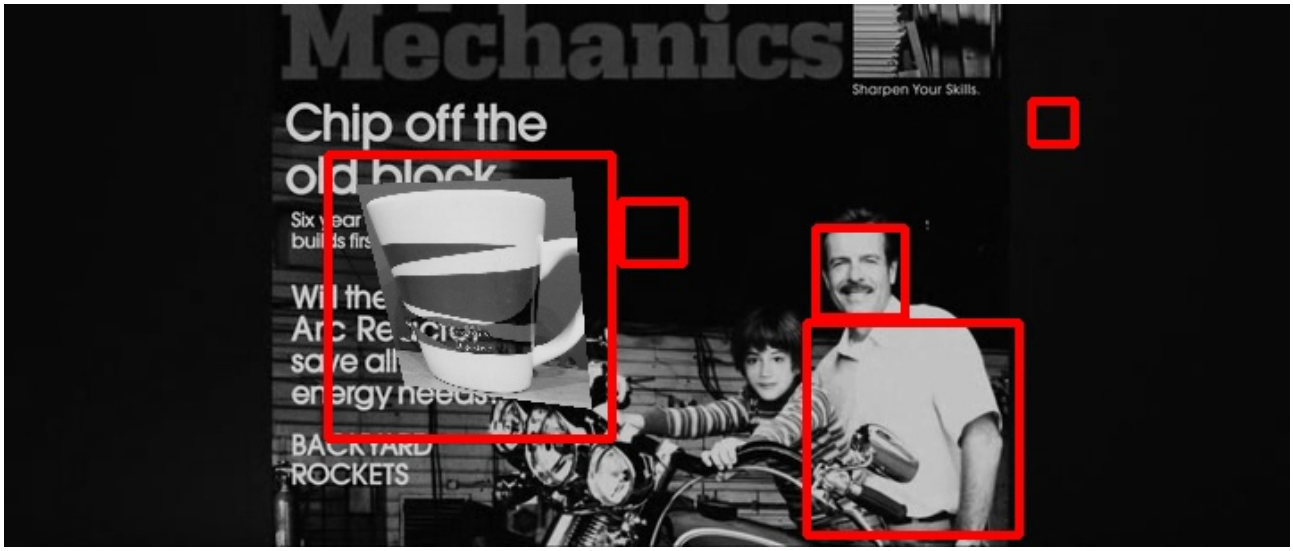
	Caso 1	Caso 2
Stages	14	14
Splits	2	2
Imagenes positivas	10000	5000

Imagenes negativas	16000	8000
Min hit rate	0,999	0,999
Maxfalsealarm	0,4	0,4
Trimming	0,95	0,95
Boosting algoritm	GAB	GAB
Error	Misclass	Misclass
Mode	2|All	2|All
Required false alarm		2,68436e-06

Los resultados obtenidos con la aplicación performance del segundo caso de los dos últimos clasificadores son los reflejados en la siguiente tabla. Dado que en el caso 1 no se ha conseguido obtener el clasificador no se refleja en la tabla.

	Aciertos			Fallos			Falsos			Totales		
Muestras	1	2	3	1	2	3	1	2	3	Aciertos	Fallos	Falsos
Caso 2	10	14	31	40	36	19	3	18	8	55	95	29

Podemos ver que los resultados obtenidos en la herramienta performance no son buenos. Incluso habiendo aumentado considerablemente el número de muestras los resultados no mejoran. Aunque el número de falsos positivos a disminuido con respecto al mejor clasificador anterior, también ha disminuido el porcentaje de detección a un 33,6%. En la imagen podemos observar como el clasificador ha detectado falsos positivos y basándonos en los resultados vemos que el porcentaje de éstos es demasiado alto como para incluirlo en el proyecto.



Esta técnica de detección de objetos es muy prometedora y hoy en día se utiliza para detección de caras muy eficazmente pero a la hora de obtener un clasificador propio los resultados no son nada halagüeños.

Por una parte el hecho de tener que poseer un número muy alto de imágenes nos dificulta la tarea de conseguir cualquier clasificador, por lo que se entiende que este tipo de detectores son adecuados para detectar objetos o formas genéricas como pueden ser caras, cuerpos humanos u objetos que posean rasgos similares entre los diferentes objetos posibles.

Además podemos decir que no existen unas pautas o unos valores para estas herramientas que nos aseguren el éxito en la obtención del clasificador. Por lo que la decisión de clasificar la taza usando tres vistas diferentes ha podido ser contraproducente. Aún así este tipo de detección no nos es útil para el proyecto ya que si el usuario quiere añadir objetos diferentes que detectar no sería posible por la dificultad técnica que conlleva la consecución de las imágenes tanto positivas como negativas. Por lo que se decide utilizar otras técnicas basadas en detección de puntos invariantes.

4.3 - SIFT & SURF

SIFT(Scale-invariant Feature Transform) es un algoritmo que detecta y describe las características locales de una imagen. El algoritmo fue publicado por David Lowe en 1999. Las características SIFT son locales y se basa en la apariencia del objeto en puntos particulares de interés y son invariantes a escala de la imagen y rotación. También son resistentes a los cambios en la iluminación, el ruido y los cambios de menor importancia en perspectiva .

El primer paso de esta técnica es utilizar la función de diferencia gaussiana para identificar los posibles puntos de interés. El siguiente paso consiste en la localización de los puntos clave, rechazando los puntos de bajo contraste y eliminando la respuesta de borde. Para calcular qué puntos se excluyen se utiliza una matriz Hessiana. Por último se define la orientación de la imagen. Se crea un histograma de orientación a partir de las orientaciones del gradiente de los puntos de muestreo de una región cercana al punto clave.

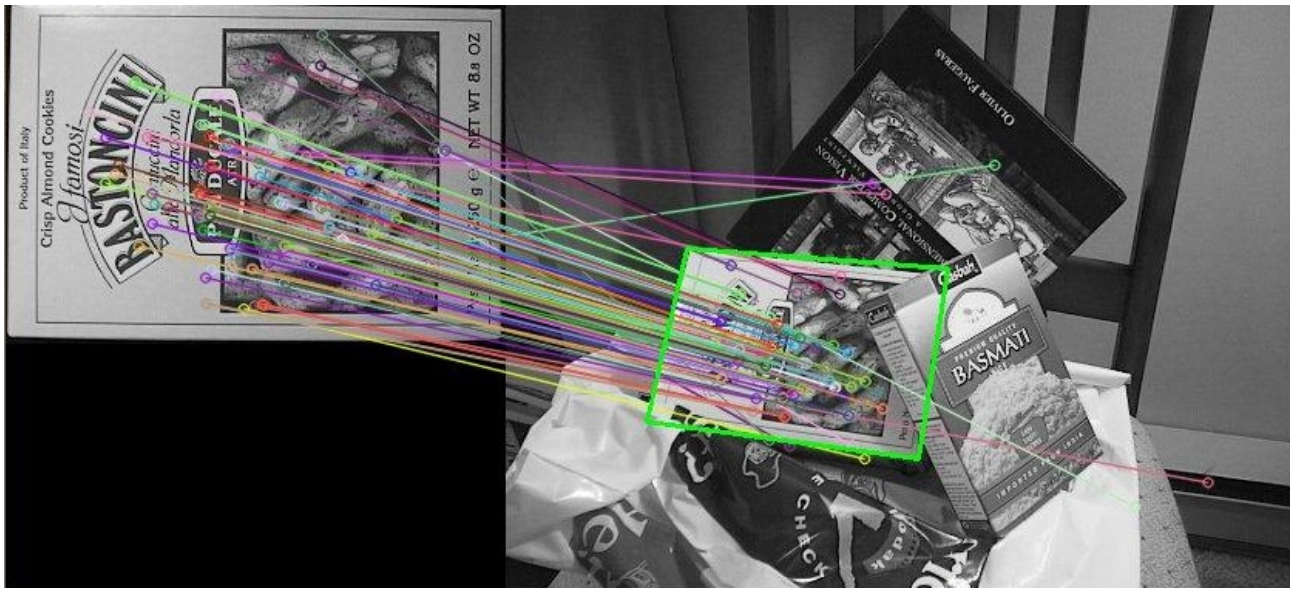
SURF(Speeded Up Robust Features)[13] es un detector basado en una aproximación de la matriz Hessiana presentado en 2008 por Herbert Bay, Andreas Ess, Tinne Tuytelaars y Luc Van Gool y tiene algunas similitudes con el detector SIFT. Pero posee mejoras considerables en el rendimiento, disminuyendo la velocidad de computación y aumentando la robustez del detector sin afectar a la detección de los descriptores. Actualmente es la principal referencia, calificada por diversos estudios (tales como [14] y [20]) como el detector con mejor rendimiento en la actualidad.

Se puede utilizar en visión por computador para tareas tales como reconocimiento de objetos o reconstrucción 3D. Dada una imagen SURF extrae los puntos de interés invariantes de la imagen. Estos puntos son características robustas que pueden ser puntos donde existe una alta alteración del contraste como esquinas o manchas.

Para utilizar este método con OpenCV en el proyecto partimos del programa de ejemplo que propone OpenCV. El algoritmo empieza con la obtención de los puntos clave de la imagen que contiene el objeto a buscar, la imagen de muestra. Una vez cargada la imagen y extraídos los puntos clave se entra en el bucle donde obtiene las imágenes de la webcam y se extraen los puntos clave para compararlos con los de la imagen de muestra. Cuando detecta correlación entre los puntos dibuja dichos puntos en las imágenes y una recta uniendo cada par de puntos concordantes. Basándose en los puntos similares se calcula la posición, tamaño y orientación del

objeto. Tras determinar las características del objeto en la imagen se dibuja un recuadro para resaltarlo y se obtiene su posición.

La ilustración 16 muestra el ejemplo de detección.



Este detector es muy potente y útil. Sólo incluyendo una imagen que contenga el objeto que queramos reconocer conseguiríamos realizar la detección pero se ha comprobado que ralentiza el programa a la hora de procesar las imágenes y calcular los puntos clave. Esta ralentización se da con solo un objeto a detectar por lo que a priori no parece una buena opción final para nuestro proyecto porque es necesario la detección de diferentes objetos. Además la imagen ha de estar en primer plano, tal y como se ve en la ilustración 16, para conseguir una correcta detección. En el proyecto se ha incluido esta técnica de detección como ejemplo de uso pero nos vemos obligados a buscar otra solución más rápida.

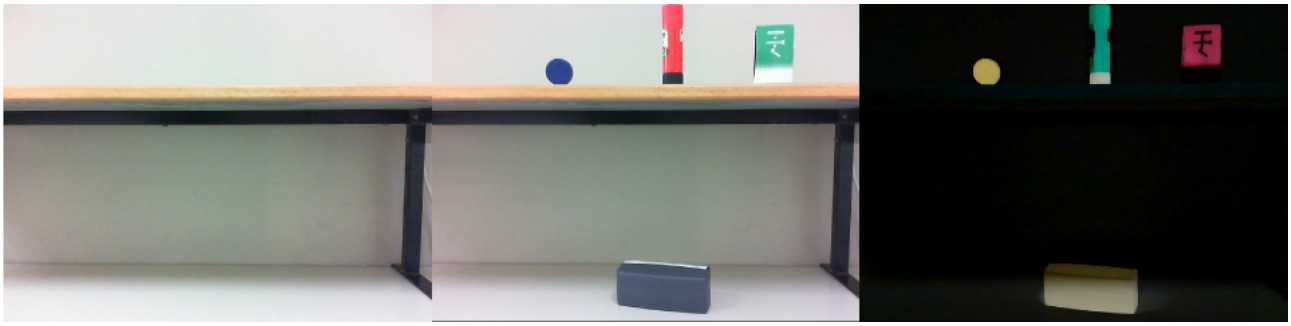
4.4 - Diferencia de imágenes y CvBlobsLib

Este método desarrollado consta de dos partes, la primera es la obtención de la imagen con la que vamos a trabajar para detectar los objetos y la segunda es el uso de la librería CvBlobsLib.

4.4.1 – Diferencia de imágenes

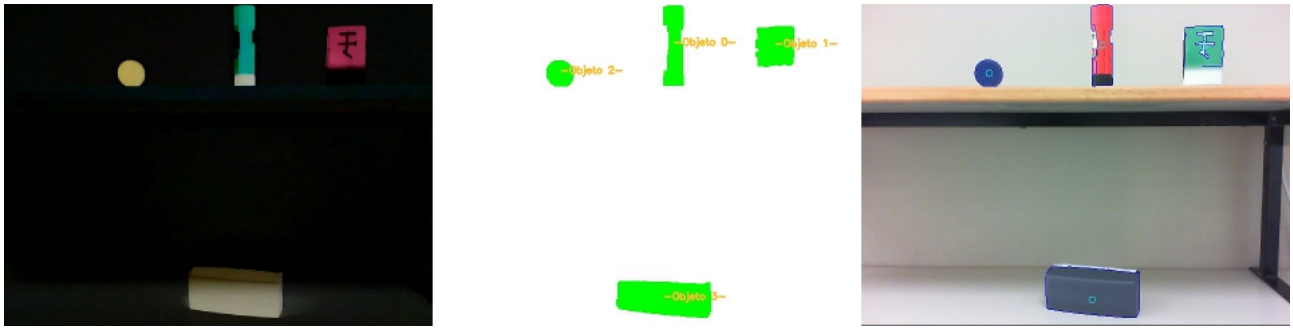
La diferencia de imágenes se usa habitualmente para detectar movimiento, por ejemplo en un andén de tren para detectar personas o movimiento en un parking. En nuestro caso hemos extrapolado esta idea a entornos estructurados donde la webcam está en una posición fija y solo varían los diferentes objetos a manipular por el robot dentro de la zona visualizada.

Esta técnica se basa en obtener una imagen de fondo donde solo se muestra el entorno de trabajo, esta imagen se resta a la imagen obtenida desde la webcam obteniendo así solo el conjunto de objetos presentes en la imagen. Indicar también que cambios en la luz provocan que esta diferencia no sea precisa dado a que los brillos no se eliminan de la imagen.



4.4.2 – CvBlobsLib

Una vez obtenida la imagen en la que sólo aparecen los objetos se procesa la imagen con la librería CvBlobsLib. Esta librería obtiene los conjuntos de píxeles que tienen en común alguna característica. Además la librería calcula la posición más alta del conjunto, la posición más baja, el área, el perímetro y más características útiles en la detección de objetos.



Después de obtener la secuencia de conjuntos ya podemos obtener la posición de cada uno y realizar las tareas que necesitemos.

Esta técnica es eficaz y muy útil pero plantea un problema, solo se puede aplicar esta solución a un entorno estructurado donde además se tenga el control de la iluminación.

4.5 - Segmentación

La segmentación de imágenes se basa en seleccionar un conjunto de píxeles adyacentes que poseen características similares como la intensidad, el color o la textura. Para ello existen diversos métodos de segmentación. A continuación se presentan algunos métodos de segmentación[15].

Umbralización

La umbralización trata de determinar un valor, llamado umbral, que separa las clases deseadas. Este valor determina el color, la intensidad o por ejemplo el brillo. La segmentación se consigue agrupando todos los píxeles con un valor de la característica mayor al umbral en una clase y los demás en otra clase. De esta manera conseguimos una imagen binaria en la que los valores positivos se determinan de color blanco y los negativos de color negro. Esta imagen la denominamos máscara.

Este método presenta notables carencias con respecto a cambios en la iluminación, por ello solo se recomienda solo como paso previo a otras técnicas de reconocimiento o en escenarios estructurados donde tenemos control absoluto de la iluminación.

Clasificadores

Los métodos clasificadores son técnicas de reconocimiento de patrones que buscan determinar un espacio característico derivado de la imagen usando datos con etiquetas conocidas. El espacio característico más común hace referencia a la intensidad de la imagen. Este método requiere de un entrenamiento previo. Todos los píxeles que posean un valor cercano al entrenado serían agrupados en una clase.

Los clasificadores son conocidos como métodos supervisados debido a que requieren datos de entrenamiento. Hay una gran cantidad de maneras en la que los datos de entrenamiento pueden ser aplicados en los métodos de clasificación. Un clasificador simple es el clasificador del vecino más cercano, donde cada píxel es clasificado en la misma clase que el dato del entrenamiento con la intensidad más cercana. Los k vecinos más cercanos, K Nearest Neighbor, es una generalización de este método.

Clustering

Los algoritmos de agrupamiento(clustering) son similares a los métodos clasificadores, pero sin utilizar datos de entrenamiento. Por tanto son métodos no supervisados. Para compensar la falta de entrenamiento los métodos de agrupamieno iteran entre segmentar la imagen y caracterizar las propiedades de la clase. Un algoritmo de agrupamiento común es el algoritmo K-Medias.

Este algoritmo agrupa datos calculando iterativamente la media de la intensidad para cada clase y segmentando la imagen mediante la clasificación de cada píxel en la clase con la media más cercana.

Aunque los algoritmos de agrupamiento no requieren de entrenamiento, si requieren una segmentación inicial o parámetros iniciales.

4.6 - Nuestro sistema de detección por segmentación

Tras ver las diferentes técnicas de detección mediante segmentación nos decantamos por la utilización de segmentación por umbralización. Este método nos ofrece la posibilidad de detectar diferentes objetos basándonos en color y forma sin un entrenamiento previo.

Para ello creamos un nuevo programa definiendo como objetivo la detección de una pelota amarilla. Para ello se utilizarán diversas técnicas de preprocesamiento de imágenes y un conjunto

de segmentaciones.

Lo primero que hacemos es obtener la imagen de la webcam y seguidamente aplicamos la función de OpenCV 'cvSmooth' para suavizar la imagen. Tras suavizar la imagen procedemos a segmentar la imagen, para ello se ha implementado la siguiente función en la que segmentamos la imagen un total de tres veces. Primero se segmenta en base al color, la segunda por la saturación y por último en base al brillo. A continuación se muestran dos extractos de las dos funciones que ilustra la segmentación en tres pasos.

```
void segmentarColor(IplImage* input, IplImage* output, int color1, int color2, int sat1, int sat2, int brillo1, int brillo2){

        //Creamos las diferentes mascaras intermedias
        ...
        //Segmentamos la imagen en base al color
        segmentar(input,color,color1,color2,0);
        cvAnd(input,color,mascara);

        //Segmentamos la imagen en base a la saturacion
        segmentar(mascara,saturacion,sat1,sat2,1);
        cvAnd(input,saturacion,mascara);

        //Segmentamos la imagen en base al brillo
        segmentar(mascara,brillo,brillo1,brillo2,2);
        cvAnd(input,brillo,output);

        //Erosionamos la imagen para reducir el ruido x3
        cvErode( output, output, NULL, 1);

        //Dilatamos la imagen para recuperar las perdidas producidas a partir de la erosion
        cvDilate( output, output, NULL, 1);

        //Suavizamos la imagen
        cvSmooth( output, output, CV_GAUSSIAN, 9, 9 );

}

void segmentar(IplImage* input,IplImage* output, int umbral1, int umbral2, int canal ){
```

```
        //Convertimos la imagen de entrada al espacio de color HSV
        IplImage* inputHSV = cvCreateImage( cvGetSize(input), 8, 3 );
        cvCvtColor(input,inputHSV,CV_BGR2HSV);

        //Creamos las mascaras con las que vamos a trabajar
        ...

        //Separamos los canales de la imagen
        cvSplit(inputHSV,canales[0],canales[1],canales[2],0);

        //Discriminamos los valores mas bajos
        cvThreshold(canales[canal],mascara1,umbral1,255,CV_THRESH_BINARY );

        //Discriminamos los valores mas altos
        cvThreshold(canales[canal],mascara2,umbral2,255,CV_THRESH_BINARY_INV );

        //Juntamos las mascaras utilizando la operacion AND
        cvAnd(mascara1,mascara2,mascara);

        //Convertimos la mascara resultante a BGR guardandola en la imagen de salida
        cvCvtColor(mascara,output,CV_GRAY2BGR);

}
```

Antes hemos visto que el gran problema de la segmentación por umbralización es la iluminación por lo que esta función intenta mejorar la calidad de la segmentación. Aún así se hace necesario mantener una luminosidad constante para evitar problemas.

En ocasiones la máscara obtenida contiene ruido del tipo “sal y pimienta”, este ruido son como motas de polvo en la imagen. Para eliminar este ruido procedemos a erosionar la imagen dos veces utilizando la función de OpenCV cvErode, esta función 'lima' los bordes de los conjuntos de píxeles. De esta manera los puntos sueltos se eliminan. Ahora es necesario obtener la imagen máscara original, para ello utilizamos la función cvDilate con la que dilatamos los conjuntos de píxeles. Dado que los puntos se han eliminado no aparecen de nuevo, pero el conjunto de los píxeles que determinan la posición de la pelota vuelve a su estado inicial.

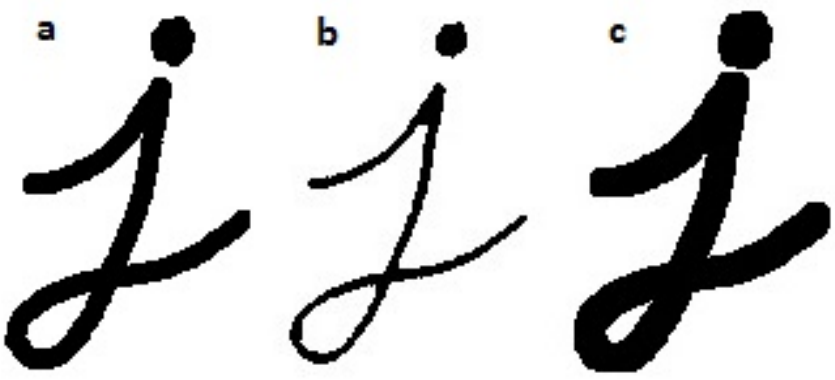


Ilustración 26: a)imagen original b)resultado de aplicar el algoritmo cvErode a la imagen original c)resultado de aplicar cvDilate a la imagen original

Ya hemos conseguido la máscara optimizada, ahora es el momento de aplicar la transformada de Hough para la detección de círculos dentro de la máscara. Para ello utilizamos la función 'cvHoughCircles' que nos devuelve una secuencia de círculos encontrados en la máscara.

Por último dibujamos los círculos detectados que se encuentran en la secuencia obtenida en la imagen inicial para poder visualizar los datos de la webcam y la pelota dibujada en la imagen.

Ilustración 28: Imagen de la webcam con la detección desactivada

Ilustración 27: Ilustración 18: Imagen de la webcam con la detección activada

La ilustración27 muestra como se ha dibujado la detección de la pelota. Una vez detectada la pelota se puede calcular su posición pero estará dentro del sistema de coordenadas de la cámara por lo que no se puede mandar al robot directamente. Previamente se debe transformar la

posición del sistema de coordenadas de la cámara al del robot. Para realizar esta transformación se han calculado las posiciones del robot que equivalen a las cuatro esquinas de la cámara. Tras ello se aplica la siguiente fórmula para cada eje.

$$nuevoValorEje1 = ((valorMaxEje1 - valorMinEje1) * posEje1) / valorPantallaMaxEje1$$

Donde:

- NuevoValorEje1 = Valor de la posición con respecto al robot que queremos obtener.
- PosEje1 = Valor de la posición con respecto al eje de la cámara.
- ValorMaxEje1 = Valor de la posición del robot que equivale al punto máximo de la webcam del eje 1.
- ValorMinEje1 = Valor de la posición del robot que equivale al punto mínimo de la webcam en el eje 1.
- ValorPantallaMaxEje1 = Tamaño de la pantalla en píxeles del eje1.

Finalmente mandamos la nueva posición al programa de control.

Obviamente el sistema de coordenadas de la cámara tiene dos ejes y el del robot tiene 3 por lo que se ha de fijar previamente el valor del tercer eje. En este caso la posición que no se puede calcular es la relativa a la profundidad del objeto en la escena. Para ello se necesitaría una segunda cámara que obtuviera las imágenes desde un ángulo perpendicular a la ya existente.

Capítulo 5 - Desarrollo

En este capítulo se muestran las diferentes etapas alcanzadas en el transcurso del proyecto. Para ello se explicarán los diferentes programas desarrollados para controlar el brazo robótico.

5.1 - Primeros pasos

El primer paso a realizar es familiarizarse con las funciones del robot. Para ello se ha utilizado el programa de ejemplo propuesto por el fabricante. Partiendo de este y teniendo en cuenta nuestras necesidades se ha creado un programa con el que controlamos el robot mediante el teclado. En esta fase nos hemos encontrado con las funciones del robot más relevantes, estas funciones han sido utilizadas en todo el transcurso del proyecto y las podemos ver en el apéndice 2.

Este programa inicial nos ha servido para establecer una primera toma de contacto con el entorno de desarrollo y con las funciones indicadas en el apéndice 2.

5.1.1 - Control del brazo mediante Joystick

Uno de los objetivos es aumentar la accesibilidad del robot por lo que se decide incluir un joystick en el manejo del robot.

El joystick utilizado se ha especificado en el capitulo de Hardware utilizado. Para controlar el joystik se ha utlizado la librería SDL. SDL es una librería multimedia, multiplataforma y diseñada para proporcionar acceso a bajo nivel a diferentes elementos hardware como teclado, ratón o joysticks. SDL está escrito en C pero funciona con C++ de forma nativa y tiene enlaces a otros lenguajes como ADA, C#, Pascal, PHP... SDL se distribuye bajo licencia GNU LGPL versión 2, esta licencia permite utilizar dicha librería libremente en programas comerciales, siempre y cuando se enlace con la biblioteca dinámica.

Dado que el joystick posee un rango de valores elevado se ha implementado la siguiente función, donde valorEjeX es el valor del eje x obtenido desde el joystick:

```
if(valorEjeX<-15){
        nuevoValorEjeX = 20;
```

```
}else if(valorEjeX>15){
        nuevoValorEjeX = -20;
}else{
        nuevoValorEjeX = 0;
}
```

Este es un extracto de la función que controla la velocidad del robot. Este extracto calcula la nueva velocidad del eje X por lo que ha de repetirse con cada eje.

La función completa clasifica los valores obtenidos en tres grupos donde definirá el valor final de la velocidad. La razón de esta clasificación es que el robot se bloquea cuando recibe constantemente diferentes valores de velocidad, y dado que es irrelevante la exactitud en la velocidad se ha decidido que si el valor obtenido del joystick es alto, mayor de 15, se define la velocidad -20. Si el valor es bajo, menor de -15, se define la velocidad 20. Por ultimo si ronda el valor 0, mayor de -15 pero menor de 15, la velocidad será 0. Esta nueva velocidad se determina en la variable nuevoValorEjeX que posteriormente se utiliza para mandarla al robot. Como vemos hemos alterado los signos de los valores, esto es dado a que el eje X del sistema de coordenadas del joystick es contrario al eje X del sistema de coordenadas del robot.

Con esta aplicación hemos conseguido dotar de una nueva posibilidad en lo que se refiere al control del robot, pero no es suficiente ya que el usuario objetivo puede no ser capaz de utilizar un joystick tan grande. Además este tipo de joysticks solo proporcionan movimientos en dos ejes por lo que se estudia la posibilidad de incluir diferentes elementos más afines al proyecto final.

5.1.2 - Control del brazo mediante Phidgets

Tal y como hemos visto en la aplicación anterior del joystick necesitamos dispositivos más adecuados para aumentar la manejabilidad del robot. En este sentido se han estudiado los dispositivos Phidgets que permiten diferentes formas de control del brazo articulado. Así podemos incluir dispositivos aumentando enormemente las posibilidades del proyecto.

Por ello se han incluido diferentes dispositivos hardware, que se han detallado en el apéndice 3 de Hardware utilizado.

El minijoystick modifica la posición de la pinza dentro del eje X e Y del robot y el deslizador determina la posición del eje Z del robot.

Para conseguir un control eficiente de la velocidad de los movimientos de cada eje dependiendo del valor de cada sensor, se ha adaptado la función creada en el programa anterior de control mediante joystick:

```
case 0://Numero de sensor
        if(value>=880){
                velocidad=-2;
        }else if((value<880)&&(value>650)){
                velocidad=-1;
        }else if((value<=650)&&(value>=450)){
                velocidad=0;
        }else if((value<450)&&(value>100)){
                velocidad=1;
        }else if(value<=100){
                velocidad=2;
        }
```

Esta función obtiene los valores de cada dispositivo Phidget y decide la nueva velocidad del robot. Dado que cada dispositivo posee un rango alto de valores se han fijado diferentes cotas, dependiendo en que zona de valores esté se decide un valor disminuyendo a 5 los posibles valores. Estos valores pueden ser {-2, -1, 0, +1, +2}, una vez determinado este valor se multiplica por la velocidad base determinada y se manda a la función que actualiza la velocidad del robot.

Se ha comprobado que si se ejecuta la actualización de las velocidades constantemente el robot se bloquea, por lo que la función que actualiza las velocidades tiene en cuenta la velocidad anterior para evitar mandar valores redundantes.

5.2 - Windows forms y el brazo iArm

Para realizar este proyecto se ha usado el entorno de desarrollo Visual Studio 2010. Para realizar el entorno gráfico se utiliza la tecnología Windows Forms. Esta tecnología dificulta el uso de OpenCV ya que tenemos que integrar las imágenes en la interfaz, pero OpenCV hace uso de ventanas propias para mostrar dichas imágenes. Para solucionar este problema se ha implementado la siguiente función, en la que obtenemos los datos de la imagen y la adaptamos al

tipo de datos de imagen que nos brinda Windows Forms.

```
//actualizamos el picturebox donde se muestra el video, equivalente a cvShowImage
video->Image = gcnew System::Drawing::Bitmap(frame->width,frame->height,frame->widthStep,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,(System::IntPtr) frame->imageData);

//Refrescamos el video
video->Refresh();
```

Además tendremos que configurar el proyecto incluyendo las librerías de OpenCV, del robot y de los Phidgets.

En Windows Forms utilizamos timers que evalúan el estado del robot, de los phidgets y de la webcam. Estos timers comprueban el estado cada cierto intervalo del tiempo consiguiendo adaptar la prioridad a nuestro gusto. Esto se hace así porque Windows Forms no contiene un bucle en el que podamos evaluar indefinidamente los cambios de estado.

En nuestro caso hemos priorizado los cambios en las velocidades del robot evaluando su estado cada 60 ms. Los timers creados son los siguientes.

- Timer de actualización de velocidades. Este timer se ejecuta cada 60 ms y evalúa las nuevas velocidades calculadas a partir de la interfaz o los phidgets.
- Timer estado. Evalúa el estado del robot desde la conexión abierta con el robot cada 100 ms. Este estado puede ser fin de movimiento, error, atasco en una articulación, pinza cerrada completamente, etc.
- Timer actualización de vídeo: Este timer actualiza la imagen obtenida desde la webcam. Se ejecuta cada 120ms.
- Timer apertura de pinza. Este timer deja que la pinza se abra durante 15 segundos. Al alcanzar este tiempo se para el timer y la apertura de la pinza. Solo se utiliza a la hora de recoger el objeto.
- Timer comprueba fin de movimiento. Comprueba si el movimiento anterior se ha terminado o no evaluando el estado del robot. Se evalúa cada 100ms ya que si se hace en un bucle el programa se bloquea.
- Timer cerrar pinza. Cada 180ms comprueba si el estado del robot es de pinza cerrada, en

este caso para el timer y empieza otro movimiento. Este timer se activa solo a la hora de coger y entregar la pelota.

5.3 – Sistema desarrollado

El resultado final de este proyecto es la interfaz gráfica obtenida desde la que tenemos todas las funciones del iArm implementadas. La siguiente imagen muestra la interfaz final.

Ilustración 29: Interfaz gráfica del sistema final desarrollado

Tal y como se ha comentado anteriormente esta interfaz es un Windows Form, en ella se reflejan las funciones del robot donde se ha priorizado la visibilidad de las funciones de movimiento cartesiano del TCP.

En la parte superior izquierda se muestran las opciones para mover el robot mediante movimientos cartesianos. De esta manera definimos una velocidad en el eje que queramos que se mueva la pinza. Este movimiento puede ser en un eje o en varios.

La zona superior derecha tenemos la detección por visión artificial que muestra los datos obtenidos desde la cámara y con la que podemos proceder a la detección. Podemos seleccionar

qué tipo de detección queremos. Hemos visto en el capítulo de Visión artificial las diferentes técnicas de detección que hemos implementado. Las tres técnicas han sido incluidas pero no todas permiten coger objetos.

La primera técnica que encontramos es SURF. Ésta solo se ha incluido a modo de ejemplo. Para este tipo de detección el objeto debe estar en primer plano y la posición obtenida no es suficientemente fiable por lo que no se puede coger ningún objeto.

La siguiente técnica es por segmentación. Ésta solo permite coger una pelota amarilla. Una vez elegida la detección por segmentación podemos indicarle al robot que coja la pelota mediante el botón 'Coger objeto'. Tras coger el objeto definimos la posición en la que queremos situar el objeto mediante la lista desplegable situada a la derecha del botón 'Dejar en'. Por último seleccionamos el botón 'Dejar en' para proceder a la entrega del objeto.

La última técnica incluida es de Blobs. Esta técnica nos permite coger cualquier objeto dentro de la escena con la excepción de objetos de color blanco o muy claros que no resalten con el fondo. Tras seleccionar esta técnica se puede elegir qué objeto coger mediante la lista desplegable. Los objetos se numeran en la imagen y éstos se ven reflejados en la lista. Tras seleccionar el objeto en la lista desplegable se pincha en el botón 'Coger objeto'. Una vez que el robot haya cogido el objeto se puede definir una posición para dejarlo mediante la lista desplegable ubicada a la derecha del botón de 'Dejar en'. Tras seleccionar la posición pinchamos en el botón de 'Dejar en'.

La parte inferior izquierda se encuentran los botones para seleccionar los movimientos automáticos. Estos movimientos son:

- 'Fold-In'. Se encarga de desplegar el robot para comenzar a utilizarlo.
- 'Fold-Out'. esta opción es la función contraria a Fold-In, en la que plegamos el robot una vez queramos terminar con el uso del brazo.
- 'Mover a inicio'. Mueve la pinza a la posición inicial de trabajo. Se ha detectado que en ocasiones el robot no es capaz de plegarse desde cualquier punto, para evitar este problema se ha incluido este botón. Cuando nos encontramos que el robot se ha atascado procedemos a utilizar este botón para volver a un estado conocido de seguridad.

La zona inferior derecha nos encontramos con los movimientos articulares. Estas funciones

mueven las articulaciones independientemente y aunque puedan parecer poco útiles para manejar el robot su uso es indispensable en ciertas ocasiones. Estas ocasiones son aquellas en las que el robot se ha atascado, ya sea por un movimiento inadecuado o por llegar a la máxima extensión del brazo. No se recomienda su uso continuado pero nos pueden resolver diversos problemas. Para ello se muestra una imagen en la que se detallan los pares articulación-número para comprender mejor que movimiento deseamos hacer.

Además se incluyen los movimientos de la pinza en sus diferentes grados de libertad. Estos movimientos son Pitch, Yaw y Roll.

Todos los botones de la interfaz que controlan la velocidad de algún eje o articulación están programados para que solo se mueva mientras el botón esté pulsado. Una vez suelta el usuario el botón de la interfaz el robot se para.

5.3.1 - OpenCV

En el programa final se han introducido tres técnicas de detección. La primera es la técnica SURF, que nos da la posibilidad de detectar una botella de agua. Tal y como se ha explicado este método no es lo suficientemente rápido por lo que solo se hace posible la detección pero no la recogida del objeto mediante el robot.

La segunda opción es la de umbralización, en la que detectamos una pelota amarilla, tal y como se ha visto anteriormente. En este proyecto solo se ha implementado la detección de una pelota amarilla pero se existe la posibilidad de modificar el algoritmo y detectar cualquier forma y color. Una vez detectada la pelota tenemos la posibilidad de recogerla. Para ello se calcula 5 veces la posición de la pelota y se hace una media para aumentar la fiabilidad de la detección. Una vez obtenido este dato se recoge y tenemos la posibilidad de mover la pelota al punto de entrega que hemos fijado previamente.

Y por último, se ha incluido el método de diferencia de imágenes junto con CvBlobsLib. Dado que la librería no nos proporciona el punto medio de los conjuntos de píxeles obtenidos tenemos que calcular. Para ello se ha implementado la siguiente función:

```
for ( i = 0; i < blobs.GetNumBlobs(); i++ )
{
	CvPoint centro;

	objetoBlob = blobs.GetBlob( i );
	centro.x = (objetoBlob->MinX()+objetoBlob->MaxX())/2;
	centro.y = (objetoBlob->MinY()+objetoBlob->MaxY())/2;


	//Dibujamos el punto central en el frame
	cvCircle(frame, centro, 5, cvScalar(255,255,0), 1);
}
```

En esta función recorremos todos los Blobs obtenidos y para cada Blob obtenemos el máximo y mínimo de X y dividimos entre 2 consiguiendo así el punto medio del eje X. Seguidamente realizamos la misma operación con el eje Y. Para marcar ese punto central dibujamos un punto en la imagen de la webcam.

5.3.2 - Phidgets

Los elementos Phidgets incluidos son dos. Ambos son para manejar el TCP del robot en los diversos ejes. Disponemos de un minijoystick para mover el TCP en dos dimensiones y un deslizador para la tercera dimensión. Ambos elementos generan una velocidad en la dirección en la que movemos cada dispositivo.

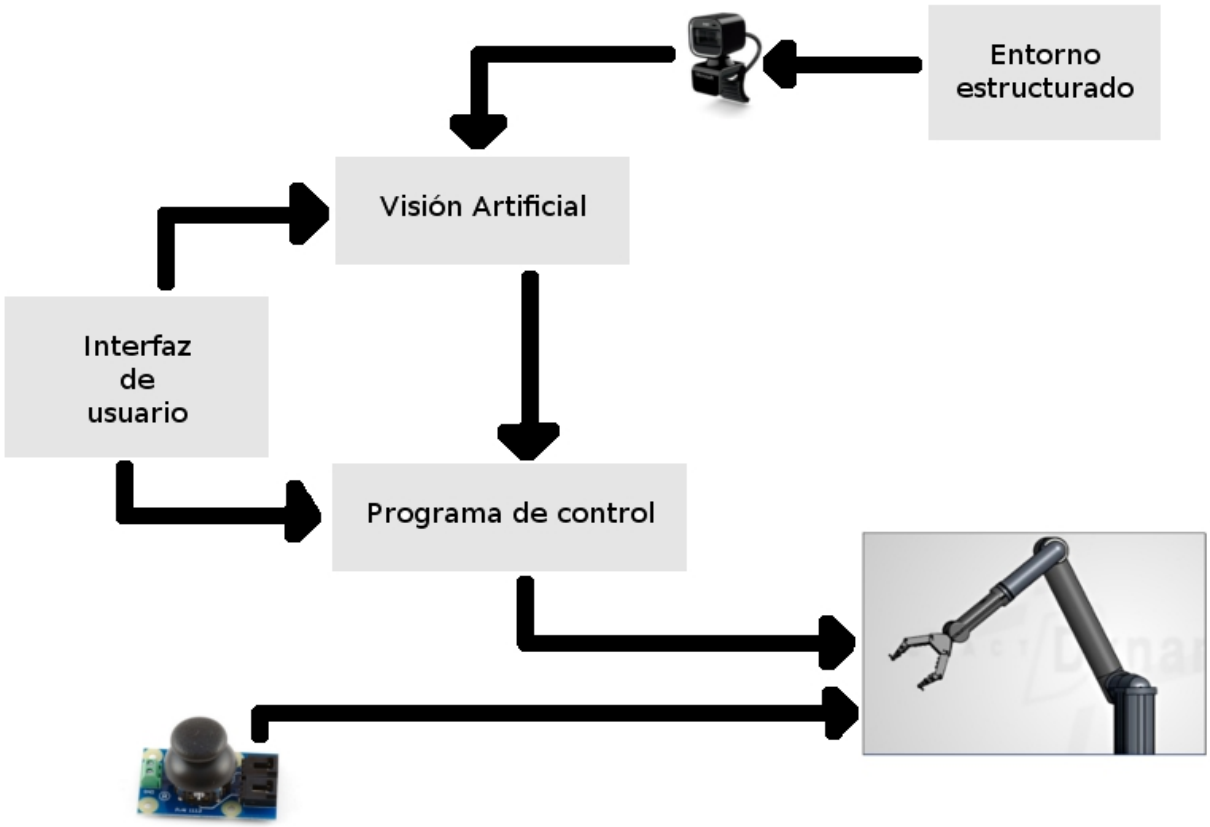
5.3.3 - Programa iArm

El programa final de control del robot iArm se ha creado a partir del primer programa creado en el que el control se realizaba mediante teclado. En este caso se han intercambiado las teclas por diferentes botones en la interfaz gráfica, además de poder usar estas funciones mediante Phidgets.

Las diferentes funciones son movimientos articulares, movimientos cartesianos, desplegar, plegar, mover a inicio, empezar detección, coger objeto y dejar objeto.

5.3.4 – Flujo de datos del sistema

En la ilustración 31 se refleja el flujo de datos que genera el sistema.



Por un lado se observa como el programa de control tiene dos entradas de datos. Por una parte las posiciones generadas desde la Interfaz Gráfica de Usuario, desde donde se puede manejar directamente el robot con los controles cartesianos o articulares. La otra entrada es desde las técnicas de Visión Artificial. Para que el programa de Visión Artificial genere estas nuevas posiciones se debe activar algún tipo de detección desde la Interfaz de Usuario. Tras activar alguna detección el programa de Visión Artificial recoge los datos de la cámara que a su vez obtiene los datos del estado del Entorno Estructurado. El programa de control decide finalmente la posición final a la que debe ir el robot. En ocasiones puede desestimar el envío de la nueva posición para evitar colapsar el iArm con datos redundantes. Además podemos manejar el iArm desde los dispositivos Phidgets directamente. Para evitar colisiones en el control del robot se da prioridad al programa de control.

Capítulo 6 - Gestión de proyecto

En esta sección se abordarán los puntos más relevantes sobre la gestión llevada a cabo durante el proyecto.

6.1 - Comunicaciones

Para mantener una uniformidad en las comunicaciones se ha establecido que la vía de comunicación estándar es vía mail en el que se seguirán unas pautas a la hora de la redacción de éste. Se diferencian dos tipos de comunicación:

- Envío de información: Para el envío de información se ha establecido que cada mail se identifique en el asunto la naturaleza de la información identificando previamente que se contiene en el marco de la realización del PFC. Para ello se empezará en el apartado asunto de cada nuevo mail con las siglas “PFC” seguido de dos puntos y por último la naturaleza del mail. Ejemplo: “PFC:Dudas con iArm”
- Envío de documentos: En el caso de envío de documentos se establecen las mismas pautas a seguir en el envío del mail pero en este caso se establecen también los formatos que deben tener los documentos para mantener una limpieza y uniformidad. Por ello se establece que el formato de envío sea en PDF. El asunto deberá empezar con las siglas “PFC” seguido de dos puntos y terminar con el nombre del documento enviado. Ejemplo: “PFC:Memoria final”

6.2 - Control de versiones

Este apartado nos ayuda a solventar los diferentes problemas existentes en todo proyecto minimizando el impacto negativo en el proyecto. Para ello se ha establecido que:

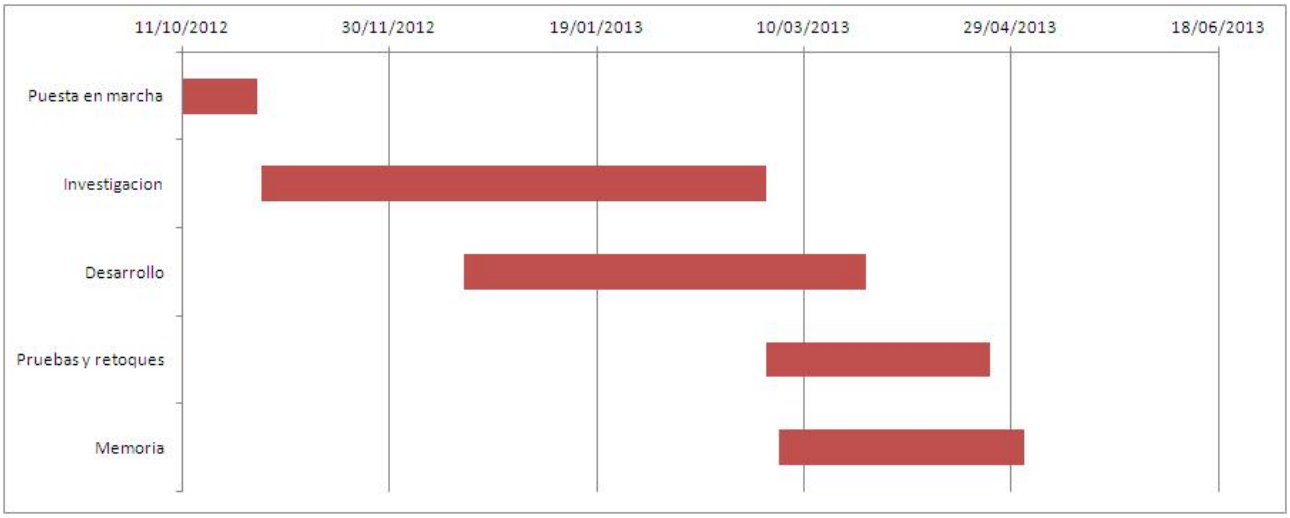
- Backups: Se establece que diaria y semanalmente se crean backups de cada programa desarrollado estableciendo la fecha de cada backup. De esta manera podemos volver a un estado anterior estable sin mucho esfuerzo siempre y cuando lo necesitemos. Por ejemplo al desarrollar un programa e integrarlo en otro programa la mejora no funciona

correctamente e incluso hace que el programa completo no funcione correctamente.

- Copias de respaldo: Además de los backups con sus diferentes versiones se ha utilizado la herramienta Dropbox. Esta herramienta nos brinda la posibilidad de recuperar documentos perdidos dado que se albergan todos los documentos en la nube.

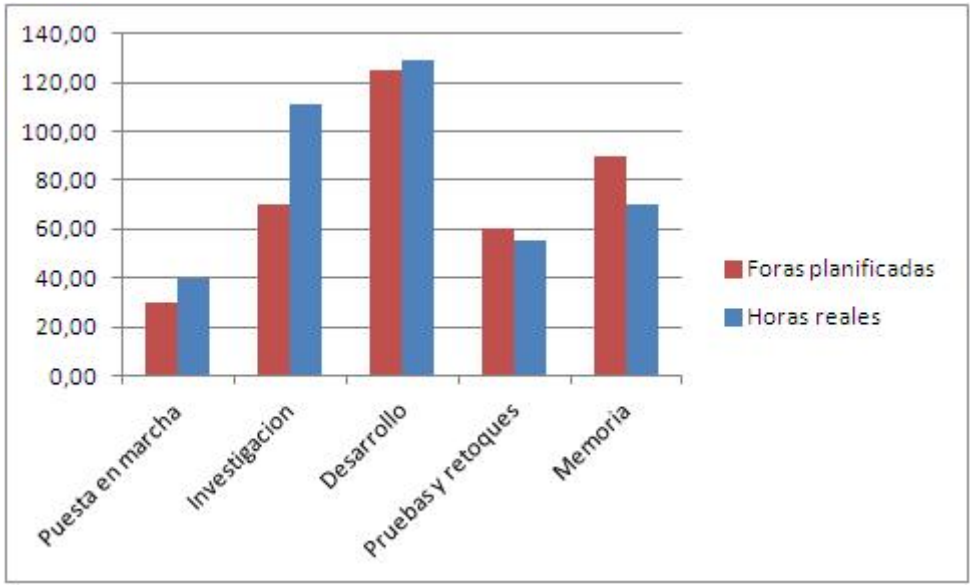
6.3 - Diagrama de Gantt

Finalmente el diagrama de Gantt generado es el siguiente. El diagrama refleja los días dedicados a cada fase con sus respectivos inicios.



En este diagrama podemos ver las diferentes fases del proyecto con sus respectivas fechas, las de inicio y las de final. Podemos ver como la fase de investigación ha llevado más tiempo de lo deseado ya que se ha tenido que llevar a cabo junto con la fase de desarrollo provocando así un retardo en la realización de esta fase.

A continuación se muestra una gráfica de horas planificadas frente a las horas reales.



En esta gráfica podemos ver que en todas las fases se han superado las horas previstas, pero los más reseñable reside en las horas superadas en investigación. Como vemos las horas dedicadas finalmente a investigación casi doblan a las planificadas, esto se ha debido a que en las investigaciones llevadas a cabo no se han obtenido los resultados deseados y esto ha obligado a repetir dichas investigaciones.

Capítulo 7 - Conclusiones y líneas futuras

En este capítulo se exponen las conclusiones obtenidas tras la consecución del proyecto y las líneas futuras de investigación.

Conclusiones

Uno de los objetivos del proyecto era mejorar la accesibilidad del robot de cara a personas discapacitadas aportando diferentes interfaces accesibles ya sea un entorno gráfico como elementos hardware. Para ello se ha creado una interfaz accesible que aúna las diferentes funciones del robot, además de la sincronización adecuada para el control de éste.

Para conseguir diferentes mecanismos de control del robot se han incluido dos elementos Phidgets. Estos elementos son un deslizador y minijoystick que nos permiten mover el TCP en todos sus ejes cumpliendo así con el objetivo de controles alternativos.

Otro objetivo conseguido pero no desarrollado completamente es la inclusión de una cámara. En un principio se pensó en la idea de incluir más de una cámara para detectar los objetos y obtener su posición en relación a sus tres ejes pero a medida que se avanzaba en el proyecto se tuvo que desestimar esta idea por la complejidad y se decidió utilizar solo una obteniendo la posición del objeto respecto dos ejes.

Uno de los puntos fuertes del proyecto desarrollado es que se ha aplicado programación modular. Es decir, se ha creado un programa para la visión artificial, otro para el control de los Phidgets, otro para las funciones del robot y otro para la interfaz. De esta manera en un futuro se pueden intercambiar fácilmente cada apartado sin tener que modificar todo el proyecto. Además la comprensión del proyecto se simplifica al estar bien estructurado.

Por lo que podemos decir que todos los objetivos iniciales planteados se han alcanzado, aunque no en todos los aspectos se hayan conseguido los resultados deseados.

Líneas futuras

Partiendo de este proyecto como base se presentan diferentes mejoras para mejorar la calidad, accesibilidad y aumentar las funcionalidades del sistema desarrollado. A continuación se presentan las líneas más interesantes bajo mi punto de vista.

Visión artificial

Hoy en día la visión artificial es una realidad pero su desarrollo está privatizado por lo que en ese aspecto este proyecto se ha visto mermado. Para solventar este hándicap se presentan diferentes mejoras que se podrían dar en un futuro.

1. La inclusión de una segunda cámara, e incluso de una tercera cámara, mejoraría la calidad del sistema de visión. La segunda cámara brindaría la posibilidad de obtener la posición de los objetos respecto del tercer eje. La tercera cámara se instalaría en la pinza dando opción a realizar las recogidas de los objetos más eficazmente.
2. Implementar un software más avanzado de detección. Con el paso del tiempo los sistemas de visión libres van mejorando, por ello se podrá conseguir un sistema más adecuado de aquí a unos años. Las mejoras del sistema podrían ser desde disminuir el tiempo de detección y procesamiento, aumentar la cantidad de objetos a detectar, mejorar la calidad de detección de los objetos o implementar otra técnica de detección.

Accesibilidad

El objetivo final de este proyecto es aumentar la accesibilidad del robot creando para ello diferentes interfaces compuestas tanto por software como hardware. En ambos casos se plantean diferentes mejoras.

1. En este proyecto se han utilizado dos elementos Phidgets. Con estos elementos conseguimos mover el robot en los diferentes ejes pero no tenemos la opción de utilizar el resto de opciones que el robot nos da. Para ello debemos usar la interfaz gráfica. Para solventar este problema sería interesante crear un mando que incluyera ambos elementos pero además tuviera diferentes botones para dar acceso a dichas funciones. Este mando se haría en base a diferentes aspectos accesibles, unificando todas las funciones, y nos daría la posibilidad de no tener que utilizar en ningún momento la interfaz gráfica.

2. Otra opción para mejorar la accesibilidad es la inclusión de reconocimiento de voz. En un principio se pensó en la idea de incluir este sistema de control pero se descarto ya que el posible usuario final podría no tener la capacidad para usar este sistema. Aún así y como mejora al sistema creado se podría estudiar este aspecto para dotar de diferentes opciones al usuario y sea éste quien decida al final.
3. El diseño gráfico de la interfaz es otro aspecto que se debería revisar en un futuro proyecto. Dado que este proyecto debe dotarse de diferentes mejoras sería necesario reorganizar los elementos gráficos para que la inclusión de otras funciones no entorpezca las ya existentes. Por ello ha de estudiarse algún sistema de organización dentro de la interfaz por algún experto en interfaces accesibles. Además de mejorar la interfaz estéticamente, ya que la actual es muy austera.

Parametrizar el sistema

Este proyecto dispone de diferentes mecanismos para controlar el robot. Estos mecanismos son fijos y no se brinda la posibilidad de modificar las diferentes opciones de cada método de control. Una mejora significativa de la interfaz creada es la inclusión de un menú de opciones en la que modificar los valores fijos. Para ello habría que parametrizar todos los valores, estos valores pueden ser la velocidad base del movimiento cartesiano, la velocidad base del movimiento articular, cambiar la posición inicial, aumentar o disminuir la sensibilidad del los Phidgets e incluso permitir la utilización de diferentes joysticks.

Pruebas con el usuario final

Otro aspecto a tener en cuenta es la necesidad de realizar diferentes pruebas con el usuario final. Estas pruebas podrían integrarse en cada proyecto de mejora del sistema para comprobar la accesibilidad y verificar la utilidad de cada proyecto y del sistema completo en sí.

Partiendo de los resultados de estas pruebas se podrían determinar las mejoras más adecuadas con respecto del usuario final. Como ejemplo las diferentes formas que podría tomar el mando, la utilidad o no de éste o la inclusión o no del reconocimiento de voz.

Acoplamiento del robot a una silla de ruedas

El acoplamiento del robot a una silla de ruedas es la finalidad del proyecto pero para ello es necesario haber incluido y probado previamente con el usuario final el sistema y las posibles mejoras. Dado que la creación del sistema no significa que sea una opción final adecuada de cara al usuario no es recomendable acoplarlo al robot antes de realizar las pruebas pertinentes. Además evaluar el gasto energético del ordenador y la usabilidad y accesibilidad de los componentes tanto gráficos como físicos se hace necesario antes de dicho acoplamiento.

Tablets

Hoy en día las tabletas están en pleno auge desbancando a los PC de las ventas. Existen infinidad de tabletas con infinidad de características. Una línea de investigación muy ambiciosa sería la adaptación del sistema a diferentes tabletas. De esta manera el usuario final tendría la opción de escoger la tableta más adecuada. Además, a la hora de acoplar el robot a una silla de ruedas el peso no sería una carga añadida para la batería y el movimiento de la silla.

Capítulo 8 - Referencias y Bibliografía

[1]https://es.wikipedia.org/wiki/Robot

[2]http://www.exactdynamics.nl

[3]http://www.tiposde.org/general/460-tipos-de-robots/

[4]4.Robots asistenciales: http://campus.usal.es/~inico/investigacion/jornadas/jornada2/comun/c26.html

[5]Javier González Jiménez "Visión por computador" 1999

[6]http://visionartificial-zuleydis.blogspot.com.es/p/navengando-en-internet-me-encontre-con.html

[8]http://opencv.org/

[9]http://ivt.sourceforge.net/

[10]http://www.aforgenet.com/

[11]http://www.phidgets.com/

[12]Reconocimiento facial http://futura.disca.upv.es/imd/cursosAnteriors/2k8-2k9/videos/trabajandoConOpenCV/secciones/Interaccion.html

[13]https://en.wikipedia.org/wiki/SURF

[14] Tinne Tuytelaars and Krystian Mikolajczyk: "Local Invariant Feature Detectors: A Survey". Foundations and Trends in Computer Graphics and Vision, vol. 3, pp. 177-280, January 2008.

[15]http://es.wikipedia.org/wiki/Segmentación_(procesamiento_de_imágenes)

[16]http://opencv.willowgarage.com/wiki/cvBlobsLib

[17]http://kinovarobotics.com/

[18]R. Cammoun et al. «Robotised Workstations for Handicapped People» *1st. European Conference on Medical Robotics,* Barcelona-1994.

[19]”Design and evaluation of a visual control interface of a wheelchair mounted robotic arm for user with cognitive impairments” Katherine Tsui 2008

[20]”A Comparison of SIFT, PCA-SIFT and SURF” Luo Juan & Oubong Gwun 2009

Apéndice 1: Tipos de datos y funciones de OpenCV

En este apéndice se muestran los diferentes tipos de datos y funciones de OpenCV utilizados en el proyecto.

Tipos de datos:

A continuación se detallan las principales estructuras de datos de OpenCV utilizadas en el proyecto.

IplImage

Esta estructura define el formato de una imagen de OpenCV. Las imágenes están compuestas por una cabecera y una zona de datos. Los campos más relevantes pertenecientes a esta estructura son:

- Width: Ancho de la imagen.
- Height: Alto de la imagen
- nChannels: Numero de canales de la imagen. Si este valor es 1 la imagen estará en escala de grises, si es 3 estará en color.
- ImageData: Puntero a los datos de la imagen.
- ROI: Estructura que nos permite operar solo sobre una región de la imagen, denominada región de interés, sobre un solo plano o ambas cosas a la vez.
- Depth: Tipo de valor que puede tomar cada píxel, las opciones son:

  IPL_DEPTH_8U Unsigned 8-bit integer (8u)

  IPL_DEPTH_8S Signed 8-bit integer (8s)

  IPL_DEPTH_16S Signed 16-bit integer (16s)

  IPL_DEPTH_32S Signed 32-bit integer (32s)

  IPL_DEPTH_32F 32-bit floating-point single-precision (32f)

  IPL_DEPTH_64F 64-bit floating-point double-precision (64f)

CvCapture

La estructura de captura de vídeo. No posee interfaz pública y es usada solamente como parámetro para las operaciones de captura de video.

CvPoint

Define las coordenadas de un punto de la imagen. Tiene dos campos, los valores enteros de la coordenada X y de la coordenada Y.

CvRect

Define la posición y tamaño de un rectángulo. Posee cuatro campos, dos para las posiciones X e Y de la esquina superior izquierda del rectángulo y los otros dos con los valores del ancho y largo del rectángulo.

CvScalar

Es un contenedor de hasta cuatro valores de tipo Double. Útil para almacenar el valor de un píxel en los diferentes canales.

CvSize

Estructura para definir tamaños de dos dimensiones. Posee dos campos, la altura y la anchura.

CvMemStorage

En ocasiones algunas funciones de OpenCV necesitan un área de almacenamiento para guardar los resultados. Esta estructura define dichas áreas.

Funciones:

A continuación se detallan las funciones utilizadas en el proyecto.

cvCreateImage

Inicializa una imagen creando la cabecera y reservando memoria. Devuelve un puntero a la cabecera de la imagen creada.

IplImage* cvCreateImage(CvSize size, int depth, int channels);

· Size: Tamaño de la imagen.

· Depth: Tipo de datos de los píxeles

· Channels: Número de canales

cvReleaseImage

Libera la memoria utilizada de una imagen.

void cvReleaseImage(IplImage** image);

· Image: Doble puntero a la cabecera de la imagen.

cvCaptureFromCAM

Inicializa la captura de video de la webcam.

CvCapture* cvCaptureFromCAM (int index);

· Index: Índice de la cámara que se desea utilizar.

cvQueryFrame

Devuelve un puntero a una captura de un frame de la cámara.

IplImage* cvQueryFrame (CvCapture* capture);

· Capture: Estructura de captura de video.

cvReleaseCapture

Libera la estructura CvCapture.

void cvReleaseCapture (CvCapture** capture);

· Capture: Doble puntero a la estructura de captura de video.

cvSetImageROI

Establece la región de interés de una imagen, delimitada por el rectángulo dado.

void cvSetImageROI (IplImage* image, CvRect rect);

· Image: Puntero a la imagen.

· Rect: Rectángulo que determina la región de interés.

cvResetImageROI

Elimina el área de interés creado para incluir la imagen completa y libera la estructura ROI.

void cvResetImageROI (IplImage* image);

cvCopyImage

Copia una imagen en otra. Las imágenes o regiones de interés que se copiaran deben ser del mismo tamaño y mismo número de canales.

void cvCopyImage(IplImage* src, IplImage* dst);

· Src: Puntero a la imagen origen.

· Dst: Puntero a la imagen destino.

cvNamedWindow

Crea una ventana para visualizar las imágenes.

int cvNamedWindow(const char* name, int flags);

· Name: Nombre de la ventana.

· Flags: Opciones de la ventana.

cvShowImage

Muestra una imagen en una ventana.

void cvShowImage(const char* name, const CvArr* image);

· Name: Nombre de la ventana.

· Image: Puntero a la imagen.

cvDestroyWindow

Destruye una ventana.

void cvDestroyWindow(const char* name);

· Name: Nombre de la ventana.

cvCvtColor

Convierte la imagen de un espacio de color a otro.

void cvCvtColor (const CvArr* src, CvArr* dst, int code);

· Src: Puntero a la imagen origen.

· Dst: Puntero a la imagen destino.

· Code: Operación de conversión de color.

cvWaitKey

Espera hasta que una tecla es pulsada.

int cvWaitKey (int delay=0);

· Delay: Retraso en milisegundos.

cvGet2D

Devuelve el valor especifico de un elemento de un array. Puede ser por ejemplo un píxel de una imagen.

CvScalar cvGet2D (const CvArr* arr, int idx0, int idx1);

· Arr: Array de origen.

· Idx0, idx1: Índices del elemento.

cvSet2D

Cambia el valor de un elemento de un array. Por ejemplo el valor de un píxel de una imagen.

CvScalar cvSet2D (const CvArr* arr, int idx0, int idx1);

· Arr: Array de origen.

· Idx0, idx1: Índices del elemento.

Apéndice 2: Funciones del robot iArm

En este apéndice veremos las funciones más relevantes del robot iArm para nuestras necesidades.

ArmOpen

Esta función abre una conexión con el robot y devuelve el controlador. Este controlador es necesario para ejecutar las diferentes funciones de movimiento del robot. Como parámetro se pasa el número de puerto en el que esta conectado el bus CAN al ordenador.

Ejemplo:

```
ArmOpen(int puertoCan);
```

ArmClose

Cierra la conexión con el robot, esta función posee un parámetro. Dicho parámetro es el controlador obtenido al abrir la conexión.

Ejemplo:

```
ArmClose(TranspHandle controladorArm);
```

ArmFoldIn

Despliega el robot para comenzar la utilización del brazo articulado. Es necesario ejecutar esta función en primera instancia para poder manipular el robot.

Ejemplo:

```
ArmFoldIn(TranspHandle arm);
```

ArmFoldOut

Pliega el robot para terminar su utilización. Es necesario ejecutar esta función al terminar de usar el brazo para evitar problemas de articulaciones y de conexión.

Ejemplo:

```
ArmFoldOut(TranspHandle arm);
```

ArmCartesianPositionsMovement

Esta función mueve el TCP a la posición indicada. Dicha posición se definirá en un array de tipo float de 6 posiciones. Cada valor hace referencia a la posición cartesiana del robot relativa a la base

del iArm. Los tres primeros valores hacen referencia a la posición del robot, en cambio las tres ultimas hacen referencia a la orientación del TCP. Este array se pasar como segundo parámetro. Además se pasa como parámetro la conexión con el robot, la velocidad lineal de movimiento del TCP y la velocidad angular de las articulaciones.

Ejemplo:

```
float pos[6]={0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
ArmCartesianPositionMovement(TranspHandle  arm, pos, float 0,float 0);
```

ArmCartesianVelocityMovement

Mueve el TCP desde la posición actual en múltiples direcciones a las velocidades especificadas. Recibe dos parametros, el primero es la conexión con el robot y el segundo es un array de tipo float en donde se especifican la velocidad de cada grado de libertad.

Ejemplo:

```
float cartSpeed[6]={0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
ArmCartesianVelocityMovement(TranspHandle arm, cartspeed);
```

ArmJointVelocityMovement

Mueve el robot en relación a la velocidad de cada articulación indicada por parámetro. Tiene tres parámetros, el primero hace referencia al controlador del robot que contiene la conexión con el robot. El segundo es un array de tipo float que contiene las velocidades de cada articulación y por ultimo es la velocidad de la apertura de la pinza.

Ejemplo:

```
float Jointspeed[6]={0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
ArmJointVelocityMovement(TranspHandle arm, Jointspeed, float gripperspeed);
```

ArmJointPositionMovement

Esta función es similar a la función ArmCartPositionMovement, la diferencia reside en que en este caso se indica el angulo de cada articulación que se desea alcanzar. Los parámetros necesarios son cinco. Como es habitual el primero es el controlador que contiene la conexión con el robot, el segundo es un array de tipo float que contiene los valores de los ángulos de cada articulación, el

tercero es la posición en mm de la pinza que se desea alcanzar, el siguiente es la velocidad del movimiento angular y por ultimo es la velocidad de apertura de la pinza.

Ejemplo:

float JointVelocity[6]={0.0, 0.0, 0.0, 0.0, 0.0, 0.0};

ArmJointVelocityMovement(TranspHandle arm, JointVelocity, float aperturaPinza, float velocidadAngular, float velocidadPinza);

ArmStop

Para todos los movimientos del robot.

Ejemplo:

ArmStop(TranspHandle arm);

Apéndice 3: Herramientas y plataforma

En este capítulo se muestran los principales recursos tanto hardware como software utilizados en el proyecto.

3.1 - Hardware

Ordenador

El ordenador utilizado es este proyecto es un ordenador de mesa con las siguientes caracteriticas:

- Fabricante: Intel
- Modelo del sistema: DG965SS
- Procesador: Intel Core 2 6320 @1,86GHz(2 CPU)
- Memoria RAM: 3072MB

Webcam

La webcam utilizada es la Lifecam 6000 HD de Microsoft. Las características son las siguientes.

- Zoom digital: 4x
- Captura de vídeo: 30 frames por segundo.
- Tamaño de imagen fija: 1280 x 800
- Ajuste de foco: Automático
- Extras: TrueColor Technology

Joystick

El josytick utilizado para realizar las pruebas de control del robot es el modelo Force3DPro de Logitech. Este joystick nos brinda multitud de botones, gatillo y minijoystick. Pero lo mas reseñable es el forcefeedback, esta función hace vibrar al joystick y opone una pequeña resistencia al usuario pudiendo avisar, por ejemplo, de que se está alcanzando un punto singular y así poder evitar un posible bloqueo del robot. Características generales:

- Numero de botones: 12
- Tipo de conexión: USB
- Forcefeedback: Sí

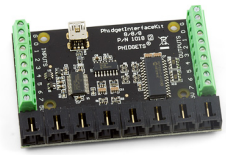
Phidgets

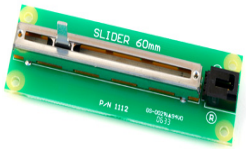
En un principio se pensó manejar el robot con el joystick anteriormente mencionado pero a medida que aumentaban las necesidades de añadir diferentes dispositivos se estudió la posibilidad de la utilización de Phidgets, este conjunto de interfaces hardware incluyen diferentes joystick, por lo que podemos adecuar el joystick mas apto para el usuario, ademas posee diferentes elementos como deslizadores y botonería que nos abren un abanico muy amplio de posibilidades.

Para el desarrollo del proyecto se ha utilizado el dispositivo Phidget Interface Kit que nos da la posibilidad de conectar 8 entradas digitales, 8 salidas digitales y 8 entradas analógicas. Esta interfaz se conecta al ordenador mediante USB.

Los elementos utilizados han sido finalmente:

- Phidget Interface Kit
- Slider 60mm
- Minijoystick





iArm

El iArm es un brazo robótico articulado fabricado por la empresa holandesa Exact Dynamics. Su uso en robótica asistencial es muy extendido por diversas razones que a continuación se detallan:

- Ofrece la posibilidad de acoplarlo a una silla de ruedas eléctrica mecánicamente. Este detalle hace que el acoplamiento sea lo más seguro posible.
- Su tamaño compacto no entorpece la utilización de la silla dónde esté acoplado.
- Posee un peso ligero para minimizar la carga de la silla.
- Consumo de energía bajo.
- Proporciona una API para poder desarrollar diferentes sistemas adaptando el brazo a nuestras posibilidades. Gracias a esta API la inclusión de joysticks o diferentes tipos de dispositivos hardware es sencillo.

Este brazo articulado posee seis grados de libertad, por lo que la pinza puede alcanzar cualquier punto del espacio y adoptar la orientación necesaria o adecuada. Todas las articulaciones están impulsadas por motores de 24V que se encuentran en la base. La ubicación de los motores permite que el brazo tenga un tamaño pequeño de modo que los pares en cada articulación son mínimos en caso de colisión. Al no encontrarse los motores en el propio brazo, el accionamiento puede ser integrado y los cables no limitan el número de rotaciones en la misma dirección. Cada motor tiene su propio engranaje para proporcionar el par necesario. Por razones de seguridad la

velocidad máxima que puede alcanzar es de 9cm/s.

El brazo tiene una capacidad de carga de hasta 1,5Kg. La transmisión de la energía se logra por medio de correas dentadas. En el hombro se utiliza un sistema de ejes huecos concéntricos mientras que en el codo hay una unidad de engranaje cónico. Los movimientos de la muñeca se realizan a través de un sistema mecánico diferencial.

El robot está equipado con dos dedos en la pinza que se abren y cierran mediante un resorte. La máxima apertura es de 9,5cm. Cuando el usuario cierra la pinza se aplica una tensión en el resorte. La pinza posee un sensor de presión que le permite detener el cierre al activarse, de esta manera evita dañar los objetos manipulados.

Para controlar la posición del manipulador se utiliza un encoder incremental. Los encoders están integrados en el motor y envían 400 pulsaciones por rotación. De esta manera se pueden obtener altas resoluciones en las medidas de posición del extremo del eje. Además se utilizan encoders absolutos durante el arranque para inicializar los encoders incrementales y para la detección del deslizamiento.

3.2 - Software

Sistema operativo Windows XP

El sistema operativo elegido ha sido el Windows XP. Este sistema operativo es una versión de Microsoft Windows, línea de sistemas operativos desarrollado por Microsoft. Lanzado al mercado el 25 de octubre de 2001. Dispone de versiones para varios entornos informáticos, como PC domésticos, de negocios o portátiles como netbooks, tabletas o centros multimedia.

Las características principales introducidas en este sistema operativo son:

- Interfaz gráfica más agradable al de sus predecesores.
- Secuencias de inicio e hibernación mas rápidas.
- Desconexión de dispositivos externos, instalación de nuevas aplicaciones y controladores sin necesidad de reiniciar.
- Uso de varias cuentas, lo que permite que un usuario guarde el estado actual y aplicaciones abiertos en su escritorio y permita que otro usuario abra sesión sin perder información.
- Escritorio remoto, que permite a los usuarios abrir una sesión con una computadora que funciona con Windows XP a través de una red o Internet.
- Soporte de módems ADSL y Wireless y establecimiento de una red FireWire.

La facilidad de conexión de los diferentes elementos hardware como los joysticks ha decantado la balanza hacia este sistema operativo. Windows XP nos proporciona los controladores automáticamente y el control sobre ellos esta bien documentado en la web. Además el robot iArm proporciona una API junto con un ejemplo de utilización enmarcado en un proyecto de Visual Studio de Microsoft.

Lenguaje de programación C++

C++ es un lenguaje de programación diseñado a mediados de los años 80 por Bjarne Stroustrup. La intención de su creación fue la de extender al exitoso lenguaje programación C con

mecanismos que permitieran la manipulación de objetos.

Posteriormente se añadieron facilidades de programación genérica, que se sumo a los otros dos paradigmas que estaban admitidos, la programación estructurada y la programación orientada a objetos. Por eso se dice que C++ es un lenguaje de programación multiparadigma.

Una particularidad de C++ es la posibilidad de redefinir los operadores y de poder crear nuevos tipos que se comporten como tipos fundamentales.

El nombre de C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre de C con clases.

La elección de este lenguaje de programación fue porque el proyecto de ejemplo propuesto por el fabricante del iArm sólo brindaba este lenguaje.

Entorno de desarrollo

Microsoft Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación como Visual C++, Visual C#, Visual J# y Visual Basic .NET.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. De esta manera se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio fue elegido para el desarrollo del proyecto ya que el proyecto de ejemplo proporcionado por el fabricante estaba también creado con este entorno. En un principio se estudiaron diferentes entornos como el QTCreator, pero la dificultades encontradas a la hora de instalar diferentes compiladores de C++ en Windows me hicieron decantarme por este entorno ya que posee un compilador interno.

Librería OpenCV

La versión de la librería de OpenCV elegida es la 2.1. En el capitulo 2: Estado del arte se han introducido la historia y sus principales características. En el Apéndice 1 podemos encontrar las

diferentes estructuras de datos y funciones utilizadas en el proyecto.

Librería CvBlobsLib

Además de la librería de visión por computador OpenCV se ha incluido una librería externa para poder detectar conjuntos de píxeles que mantengan alguna característica similar. La versión utilizada es la última lanzada a fecha de este documento y es la 8.3. En [16] encontramos todo lo necesario para poder trabajar con esta librería.

Anexo 1: Manual de instalación

A1.1- Introducción

En este manual se detallan los pasos necesarios para instalar y configurar el proyecto de tal forma que sean posibles futuras mejoras. Por un lado se explican las librerías y el entorno de desarrollo utilizados. Por otro lado se detalla la configuración a seguir para su correcto funcionamiento. Este manual es meramente orientativo ya que se puede cambiar tanto de sistema operativo como de entorno de desarrollo.

A1.2- Requisitos

Para la instalación de este proyecto necesitamos tener los códigos fuente de cada módulo. En la siguiente tabla se encuentran listados todos los fuente necesarios.

Cabeceras	Códigos fuente
• miIARM.h • miOpenCV.h • miPhidget.h	• miIARM.cpp • miOpenCV.cpp • miPhidget.cpp

En la siguiente tabla se muestra la relación de software y hardware utilizado así como las versiones de cada uno.

Software/Hardware	Nombre	Versión
Sistema operativo	Windows XP	Indiferente
Entorno de desarrollo	Visual Studio 2010	2010
Librería de VA	OpenCV	2.1
Webcam	Lifecam 6000HD	Indiferente
Dispositivos Hardware	Phidgets	Indiferente
Blobs	cvBlobs	0.10.4

A1.3- Instalación previa

Este proyecto se compone de diversos elementos que se abordarán por partes. El primer paso es tener el iArm debidamente conectado al ordenador. La ilustración 37 muestra el esquema de conexión.

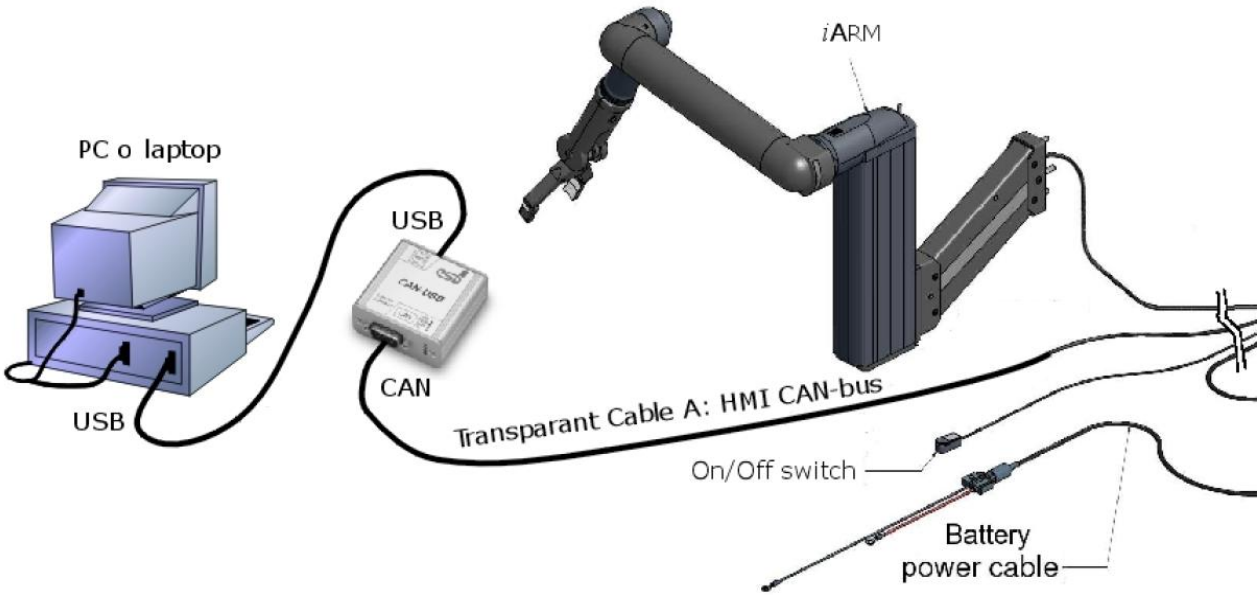


Ilustración 37: Esquema de conexión

Vemos en el esquema como el iArm se conecta al ordenador mediante bus CAN, para ello debemos conectar el iArm al conversor USB-CAN y éste al ordenador. Para que esta conexión funcione correctamente debemos tener el software adecuado instalado en el sistema. Dado que estamos usando Windows XP podemos instalar este software automáticamente, para ello conectaremos el USB-CAN al ordenador y el gestor de software del Windows XP buscará el controlador más adecuado para la interfaz de conexión. Si este gestor no encuentra ningún tipo de software tendremos que meter el CD donde viene el software. Este CD se encuentra junto a las especificaciones del robot. En este caso hemos instalado el software Ntcan.NET. Para terminar con la instalación del robot comentar que aunque es un bus CAN no se puede tener conectados más de un dispositivo de entrada al robot. Es decir, al tener conectado el robot al ordenador debemos desconectar el teclado alfanumérico para el correcto funcionamiento del robot.

El siguiente paso es conectar la webcam que vayamos a utilizar, en este caso se utiliza la webcam Lifecam 6000 HD de Microsoft. Al igual que el conversor anterior, conectaremos la cámara y esperaremos a que el gestor de software instalé el controlador adecuado para la cámara.

Ya tenemos la cámara conectada por lo que debemos instalar la librería de visión artificial. En nuestro caso instalaremos la librería de OpenCV versión 2.1.0. Esta librería está contenida en el

CD adjunto a esta memoria y también se puede descargar desde la página de descargas de OpenCV[1].

En este proyecto se han utilizado diferentes elementos Phidgets por lo que debemos instalar el software adecuado. En este caso no funciona el gestor de Windows XP. Encontraremos el archivo de instalación en el CD adjunto y también lo podemos descargar directamente desde la web[2]. Desde la página de descarga encontraremos diferentes versiones de descarga, en este caso no tenemos limitaciones por una u otra por lo que la más reciente será la adecuada.

Para comenzar a utilizar el sistema desarrollado tenemos que abrir el proyecto completo desde el entorno de desarrollo Visual Studio 2010. Una vez dentro nos aseguramos que todos los elementos están conectados. Estos elementos son la cámara, el robot mediante el USB-CAN y los Phidgets. Si no están conectados debidamente el sistema no se iniciará. Tras comprobar estas conexiones pulsamos el botón F5 para iniciar el programa. Esta opción además de iniciar el programa inicia el sistema de Debug.

A1.4- Proyecto nuevo: Configuración

A continuación se explican los pasos a seguir para crear un nuevo proyecto utilizando los programas modulares. La interfaz no se incluye ya que si se quiere partir desde el proyecto final no es necesario ningún tipo de configuración añadida. El proyecto completo se encuentra en el CD adjunto a esta memoria.

A1.4.1- Creación de proyecto y configuración

Lo primero que tenemos que hacer es crear un proyecto nuevo, este proyecto nuevo será un 'Windows Forms Application'. Una vez seleccionado el tipo de nuestro proyecto debemos darle un nombre, para este manual crearé un nuevo proyecto llamado 'Manual' y se guardará en el escritorio. Este paso lo vemos en la ilustración 38.

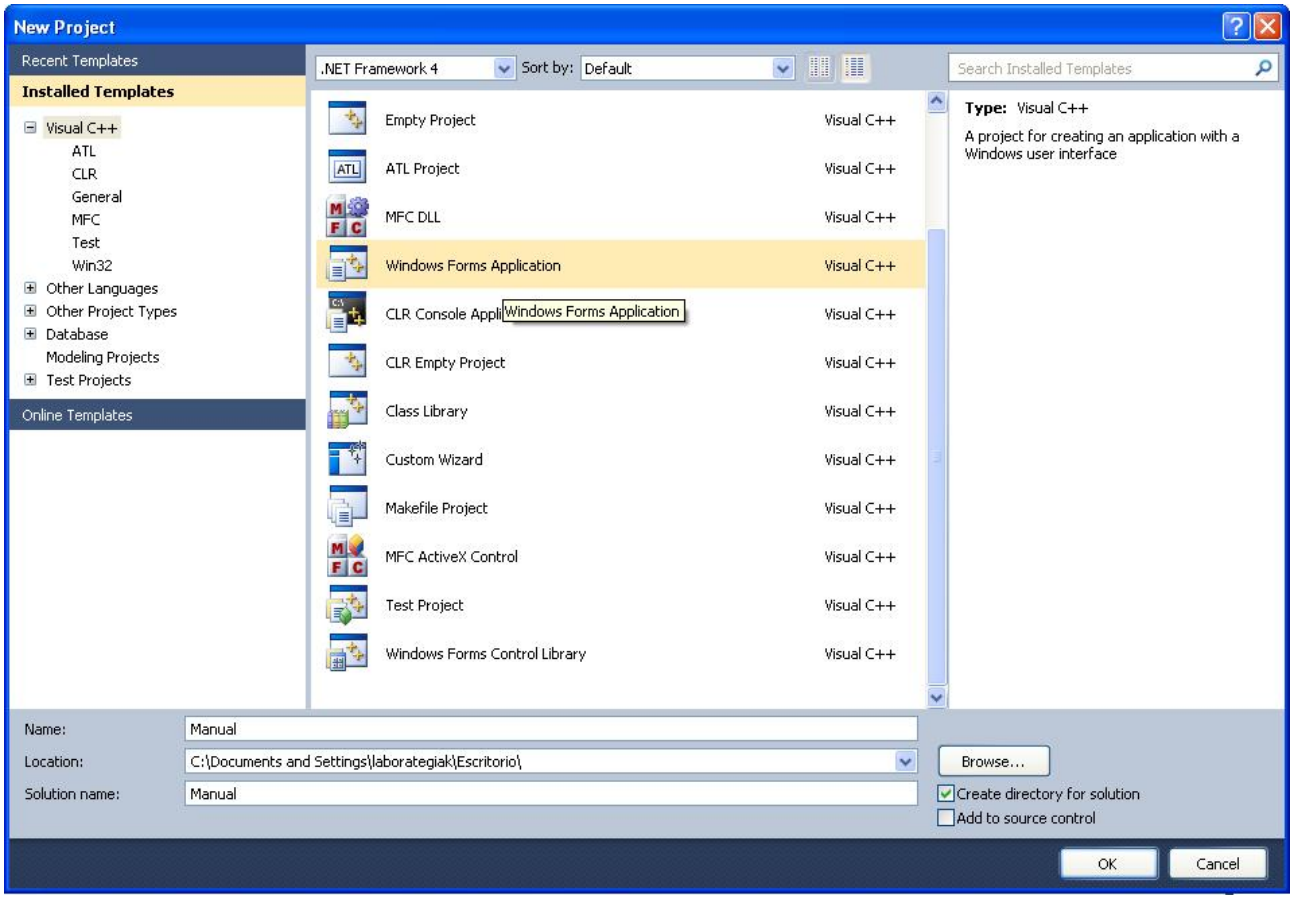


Ilustración 38: Creación de proyecto nuevo

Ya tenemos nuestro proyecto creado y ahora es el paso de las configuraciones previas. Clicamos con el botón derecho del ratón en el nombre del proyecto y seleccionamos propiedades. Una vez dentro de las propiedades seleccionamos 'Configuration Propierties/General' y nos mostrará un menú de opciones similar al mostrado en la ilustración 39.

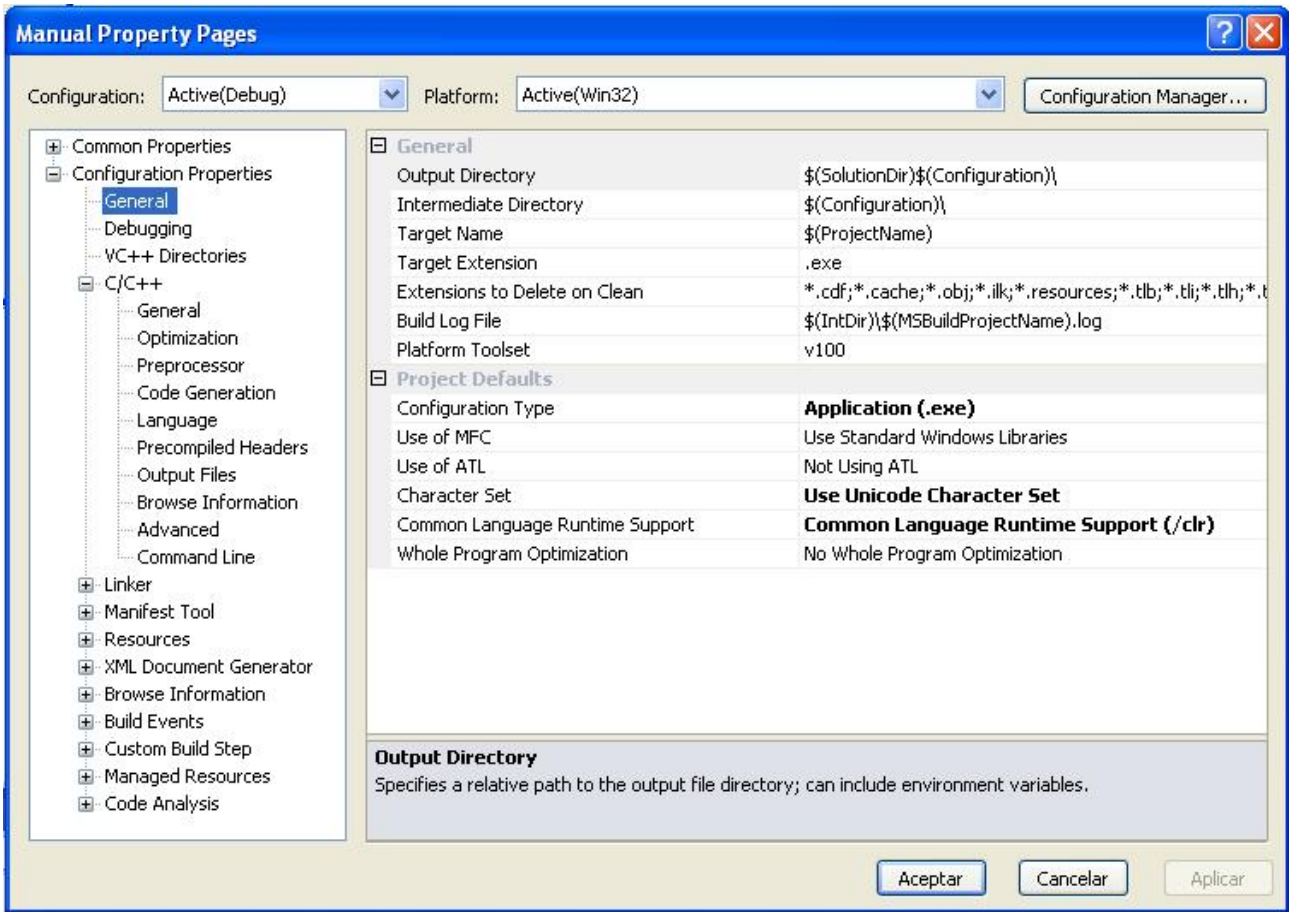


Ilustración 39: Propiedades de proyecto

En el apartado de 'Project Defaults' debemos modificar las siguientes características:

Configuration type	Application(.exe)
Use of MFC	Use standard windows libraries
Use of ATL	Not using ATL
Character Set	Use unicode Character Set
Common Language RunTime Support	Common Language RunTime Support(/clr)
Whole program optimization	No whole program optimization

Una vez hecho esto debemos modificar un atributo en el apartado de 'Configuration Properties/C/C++/General'.

Common Language RunTime Support	Common Language RunTime Support(/clr)

A continuación debemos modificar diferentes atributos en las secciones de 'Configuration Properties/Linker/General' y 'Configuration Properties/Linker/System'.

Enable incremental linking	NO(/incremental:no)
Subsystem	Console(/SubSystem:console)

Tras esto tenemos configurado el proyecto inicialmente. Ahora debemos introducir y configurar las librerías.

A1.4.2- Instalación de librerías

El orden de la inclusión y configuración de las librerías es irrelevante. En mi caso se irán añadiendo por orden de necesidad.

La primera librería que necesitamos instalar es la del robot iArm. Para ello necesitamos los archivos 'transparent.dll', 'transparent.lib', transparent_static.lib' , 'Transparent_Types.h' y 'Transparent.h'. Estos archivos se encuentran en el CD adjunto a este documento, dentro de las carpetas 'Transparent mode/bin' y 'Transparent mode/include'. El primer paso para incluir esta librería al proyecto es copiar los archivos mencionados y dentro de la carpeta del proyecto recién creado. Tras esto debemos incluir estas cabeceras en el código. Windows Forms utiliza un mecanismo de encabezados precompilados lo que significa que aúna todas las cabeceras para reducir el tiempo de compilación. Esta unificación no es más que la creación de una cabecera que contenga todas y en cada programa utilizar únicamente esta cabecera. Por consiguiente las cabeceras del iArm se deben incluir en el archivo común que se llama 'stdafx.h'. En este archivo debemos incluir las siguientes líneas.

```
#include 'Transparent.h'
#include 'Transparent_Types.h'
```

Además tenemos que incluir la dependencia con la librería en el proyecto. Para ello accederemos a las propiedades del proyecto y añadiremos en 'Additional Dependencies' dentro del apartado 'Configuration Properties/Linker/Input' la siguiente línea.

transparent.lib

En este momento ya tenemos las librerías del robot incluidas en el robot, ahora incluiremos las de los Phidgets. Para ello necesitamos los archivos 'phidget21.h', 'phidget21.dll' y 'phidget21.lib' que encontraremos en la carpeta instalada previamente. Estos archivos debemos copiarlos en la carpeta del proyecto. Al igual que antes debemos incluir al proyecto la cabecera. En el archivo 'stdafx.h' añadiremos la siguiente línea.

```
#include 'phidget21.h'
```

Ahora debemos incluir la dependencia del proyecto con la librería de Phidgets. Accederemos a las propiedades del proyecto y, al igual que antes, añadiremos en 'Additional Dependencies' dentro del apartado 'Configuration Properties/Linker/Input' la siguiente línea.

phidget21.lib

Ahora incluiremos las librerías de Visión Artificial. En este caso no copiaremos los archivos sino que incluiremos la carpeta donde hemos instalado las cabeceras de la librería al proyecto. Accedemos en las propiedades del proyecto y en la sección de 'Configuration Properties/VC++ Directories" y editaremos primero 'Include directories' incluyendo la dirección de la carpeta donde se encuentran las cabeceras de OpenCV, en nuestro caso:

C:\OpenCV2.1\include\opencv

Después editaremos "Library directories" incluyendo la dirección donde se encuentran las librerías de OpenCV, en nuestro caso:

C:\OpenCV2.1\lib

Hecho esto podemos incluir las cabeceras en el proyecto. Incluiremos en 'stdafx.h' las siguientes líneas.

```
#include <cv.h>
#include <highgui.h>
```

Por último debemos incluir las dependencias en las propiedades. Incluiremos en 'Additional dependencies' las siguientes líneas.

cv210.lib
cvaux210.lib
cxcore210.lib
cxts210.lib
highgui210.lib
ml210.lib
opencv_ffmpeg210.lib

Además de la librería de OpenCV tenemos que incluir la extensión cvBlobs. Para ello copiaremos todos los archivos que contenga la carpeta 'cvBlobs' y los copiamos en la carpeta del proyecto. En la cabecera 'stdafx.h' añadiremos las siguientes líneas.

```
#include <blob.h>
#include <BlobContour.h>
#include <BlobLibraryConfiguration.h>
#include <BlobOperators.h>
#include <BlobProperties.h>
#include <BlobResult.h>
```

Además tenemos que incluir la dependencia. En 'Additional dependencies' añadiremos:

cvblobslib.lib

Para terminar con la configuración de librerías añadiremos una cabecera de Windows. Esta cabecera es 'conio.h'. Debemos incluirla en el archivo 'stdafx.h'. Por lo que finalmente el archivo nos quedará como sigue.

```
#pragma once

#include <conio.h>

#include "Transparent.h"
#include "Transparent_Types.h"

#include "phidget21.h"

#include <cv.h>
#include <highgui.h>
#include "blob.h"
#include "BlobContour.h"
#include "BlobLibraryConfiguration.h"
#include "BlobOperators.h"
#include "BlobProperties.h"
#include "BlobResult.h"
```

A1.4.3- Códigos fuente

Ya tenemos las librerías incluidas y listas. Ahora toca incluir los programas modulares. Estos programas son:

Cabeceras	Códigos fuente
• miIARM.h • miOpenCV.h • miPhidget.h	• miIARM.cpp • miOpenCV.cpp • miPhidget.cpp

Estos archivos los añadimos a la carpeta del proyecto. Tras ello pinchamos con el botón derecho del ratón en el proyecto y seleccionamos la opción de 'Add/Existing Item...'. Seleccionaremos todos los archivos y se incluyen en el proyecto. Ahora debemos incluir las nuevas cabeceras propias en 'stdafx.h'. Las nuevas líneas a incluir son:

```
#include "miIARM.h"
#include "miPhidget.h"
#include "miOpenCV.h"
```

Por último debemos incluir las imágenes de muestra para la Visión Artificial. Debemos copiar la carpeta 'muestras' a la carpeta del proyecto.

En este manual se ha descrito la configuración de todos los módulos exceptuando la interfaz pero no es necesario instalar e incluir todos los módulos. Podemos incluir solo los que vayamos a utilizar, de esta manera reducimos el tiempo de ejecución del programa.

Ya tenemos nuestro proyecto listo para crear una nueva interfaz. La interfaz que tenemos que modificar es el formulario llamado por defecto Forms1 que se encuentra en la sección de 'Headers files'.

A1.5 Enlaces

[1]Página de descargas de OpenCV. http://opencv.org/downloads.html

[1]Página de descargas de Phidgets. http://www.phidgets.com/docs/Operating_System_Support