
Departamento de Ingeniería de Sistemas y Automática

TRABAJO FIN DE GRADO

# Diseño e implementación de un sistema de reconocimiento vocal a través de información RGB-D

*Autor:* Alfonso Conti Morera

*Director:* José Carlos Castillo Montoya
*Tutor:* Irene Pérez Encinar

Leganés, septiembre 2015

**Título:** Diseño e implementación de un sistema de reconocimiento vocal a través de información RGB-D
**Autor:** Alfonso Conti Morera
**Director:** José Carlos Castillo Montoya
**Tutor:** Irene Pérez Encinar

# EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ... en ....., en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

# Agradecimientos

De antemano, pido disculpas si se me pasa mencionar a alguien o no le dedico todo el texto que debería.

Creo conveniente agradecer en primer lugar a la familia. Gracias a mis padres por apoyarme en todo momento y confiar siempre en mí. Han sido, son y serán siempre un pilar fundamental en mi vida. Gracias también a mi hermana por estar cuando se la necesita, por ese sentimiento de saber que está ahí sin necesidad de comprobarlo.

A Verónica, por estar a mi lado y animarme en todo momento, por haber hecho y hacer todo lo que está en su mano para que logre mis objetivos, y lo más importante, por alegrarme la vida y por su amor.

A las amistades, gracias a Rubén D. por esa confianza depositada en mí y ese "acaba ya macho, que sé que tú puedes", por estar ahí en los momentos difíciles. A Carlos y Rubén B. por las revisiones, ánimos y ayuda en general, por los momentos de desconexión, viajes, y porque habéis demostrado con creces lo buenos amigos que sois. Gracias también a David, Cris, Nori, Luis y Rebe por estar ahí, sois geniales.

Agradecer también a mis compañeros de clase Álvaro, Georgi, Andrea, Carlos, Rubén y Jorge, que me aceptaron como el "abuelo", me ayudaron siempre que podían e hicieron que terminar la carrera y haber coincidido con ellos haya merecido la pena. Mención especial para Álex y Raúl, por darme la oportunidad de trabajar con ellos en grupo y que ese grupo se haya convertido en amistad. Sin duda habéis formado parte de que considere buenísima esta experiencia en la universidad.

Gracias a mis tutores, Irene y José Carlos, por el tiempo y la dedicación que habéis tenido conmigo. Gracias por las reuniones y

la implicación que habéis demostrado durante todo el trabajo.

Gracias Javi, simple y claramente, porque sin ti nada de esto habría sido posible, por tu ayuda en los momentos de frustración. Te considero un gurú de la informática, y en general, alguien de quien siempre se pueden aprender cosas.

A mis compañeros de la oficina, y en especial a Rubén A., por darme ánimos cuando lo veía difícil.

Y por último, agradecer a todos los profesores y compañeros restantes, buenos y malos, por llenar de anécdotas mi paso por la universidad.

Gracias a todos, por haberme hecho llegar a lo que soy.

# Resumen

Actualmente, los estudios e investigaciones en el ámbito de la robótica social asistencial van en aumento debido a la necesidad latente de sistemas de ayuda y asistencia para personas con necesidades especiales. En este contexto, el proyecto RobAlz busca introducir un robot social como elemento asistencial durante las terapias de enfermos de Alzheimer o con otros tipos de demencia. La terapia vocal es una de las tareas a cubrir para el tratamiento de estas personas. A través de ejercicios de pronunciación de sílabas o palabras cortas, el robot debe ser capaz de asistir al terapeuta para facilitar las sesiones con los enfermos.

Por ello, en este proyecto se pretende diseñar e implementar un sistema de reconocimiento visual de vocales mediante la fusión de información 2D y 3D. De esta forma se dispondrá de un sistema alternativo, o complementario, al reconocimiento de voz, que sea capaz de cubrir los casos en los que los sistemas ya integrados obtienen resultados poco fiables, además de apoyar las funciones de logopeda para preservar la vocalización.

El sistema desarrollado es capaz de recibir los datos 2D y 3D captados por el sensor Microsoft Kinect, detectar la cara del usuario y extraer los puntos que forman la boca, para emparejar esta información en 2D con la información recibida en 3D. Una vez unificada la información de la imagen, el sistema realiza la clasificación de los valores mediante un clasificador ya entrenado, para asignar la vocal que más se ajuste a los datos captados.

**Palabras clave:** robótica social, reconocimiento visual, clasificación, RGB-D.

# Abstract

Nowadays, the assistance has gone grown by the basic need of care and assistance systems for people with special necessities. In this context, the RobAlz project looks for introduce a social assistance robot into the therapies of Alzheimer's patients or other kinds of dementia. The vocal therapy is one of the tasks to cover for the treatment of these people. Using exercises of pronunciation of syllables or short words, the robot should assist the therapist to ease the sessions with this patients.

For this, this project pretends to design and implement a system for visual recognition of vowels using the fusion of information in 2D and 3D. Thus it's available of an alternative system, or complementary, for the voice recognition, able to cover the cases in which the integrated systems obtain results less reliable, also for help the functions of speech therapies.

The developed system is able to receive the 2D and 3D data from the sensor Microsoft Kinect, find the face of the user and extract the mouth points, for match the information in 2D with the 3D points. Once unified the data of the image, the system classified this with a trained classificator, for obtain the vowel that more adjustment to the received information.

**Keywords:** social robotic, visual recognition, classification, RGB-D.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

En la actualidad, cada vez es mayor la influencia de la tecnología como mecanismo de mejora en la calidad de vida de las personas. Multitud de dispositivos se usan para facilitar o ayudar a la realización de tareas cotidianas, de forma que las personas que los utilicen puedan desempeñar estas tareas sin necesidad de grandes conocimientos o una supervisión constante durante el proceso de las mismas. Estos dispositivos también permiten que personas con necesidades especiales puedan valerse por sí mismas y llevar el día a día de la forma más normal posible, es decir, otorgando un nivel de autosuficiencia que serían incapaces de tener sin ellos.

En el ámbito de la robótica también se busca este objetivo, y por ello cada vez existen más proyectos que hacen uso de los robots para la asistencia de personas con necesidades especiales. Casos como los niños con déficit de atención, síndrome de Asperger, autismo o hiperactividad, o personas adultas con demencia senil o deterioros cognitivos. En ese contexto, el proyecto RobAlz, colaboración entre el RoboticsLab de la Universidad Carlos III de Madrid y la Fundación Alzheimer España (FAE), busca asistir a enfermos de Alzheimer u otros tipos de demencia mediante el uso de un robot social.

La robótica social asistencial [2] es una rama de la robótica dedicada a la asistencia de personas a través de la interacción. En el proyecto RobAlz se están realizando diferentes aplicaciones para que el robot Mini, creado para este proyecto y cuyas características pasarán a detallarse en la sección 2.4 del documento, sea capaz de brindar asistencia, entretenimiento y estimulación cognitiva a las personas con Alzheimer, tratando de mejorar la calidad de vida de

estas personas.

## 1.1. Descripción del problema

Las terapias de estimulación cognitiva favorecen la ralentización del deterioro en los pacientes de Alzheimer. Esto implica que para el proyecto RobAlz interesa la implementación en Mini de un gran abanico de ejercicios y actividades para realizar estas terapias. Entre las que se encuentran los ejercicios pedagógicos para tratar los problemas de logopedia en que pueda derivar la enfermedad.

Para ello, Mini dispone de sistemas de reconocimiento de voz integrados cuyos resultados son fiables para palabras o frases, pero no obtienen buenos resultados cuando se trata de palabras cortas, sílabas o letras sueltas. Además, estos sistemas no aportan información sobre cómo se está realizando la pronunciación, cuánto se está articulando o moviendo la boca. Esto dificulta la realización de ejercicios de pronunciación y movimientos de los músculos de la boca.

En el proyecto RobAlz se requiere una herramienta auxiliar que sirva de aproximación para la realización de estas tareas. Este primer objetivo es la detección de vocales mediante un sistema visual, de tal forma que, para un futuro, brinde la posibilidad de desarrollar módulos que permitan completar las tareas que forman parte de las terapias de estimulación cognitiva, como puede ser detectar si la posición de la boca es la adecuada para la vocal pronunciada, ejercicios para los músculos de la boca, etc.

## 1.2. Motivación

Dentro del proyecto RobAlz, en colaboración con un centro de día donde se están realizando los experimentos, se ha detectado la utilidad de que el robot sea un asistente para la terapia vocal para las personas con demencia, ya que puede ayudar a ralentizar el deterioro en la capacidad de vocalizar. Este sistema deberá reconocer la palabra dicha por el enfermo, normalmente sílabas cortas o palabras sencillas, y comprobar si se ha vocalizado de forma correcta.

De este modo, se plantea el desarrollo de un sistema alternativo,

que utilice técnicas de visión y que permita reconocer las vocales pronunciadas por los usuarios, de forma que sirva como estudio previo de la utilidad de estas técnicas de reconocimiento y un prototipo base para futuros desarrollos. Si el resultado del trabajo resulta satisfactorio, posibilitará la realización de estudios posteriores más detallados y que permitan la implementación de futuros desarrollos de las terapias de pronunciación y movimientos de músculos de la boca.

## 1.3. Objetivos del trabajo

El principal objetivo de este trabajo es el **diseño e implementación de un sistema de reconocimiento visual de vocales mediante la fusión de información 2D y 3D**. El sistema funciona bajo *Robot Operating System* (ROS, sección 2.6.1), que permite la integración en cualquier otro robot sin dependencias de hardware. La información visual se obteniene mediante el sensor Microsoft Kinect que permite capturar los datos en 2D y 3D, en un formato compatible con OpenCV, librería que se detallará en la sección 2.6.3, y que facilita el manejo de estas tareas. Además, al ser un sensor de bajo coste e independiente del robot, posibilita la compatibilidad del programa en otros entornos.

A continuación se definen los hitos que constituyen la realización del proyecto:

- **Estudio y análisis de los detectores faciales**
  Se realiza un estudio y análisis de detectores faciales especificados en el estado del arte. Este estudio determinará una técnica capaz de identificar la cara del sujeto y los distintos elementos que la componen (ojos, nariz, boca...).
- **Diseño e implementación de un módulo de detección facial**
  Este módulo será el encargado de realizar la detección de la cara del sujeto mediante la integración de la librería escogida en el hito anterior. Se pretenden obtener los puntos que forman la boca para su análisis y clasificación. Este módulo se encargará de emparejar la imagen recibida por RGB con la nube de puntos de profundidad enviada por el sensor Kinect.
- **Diseño e implementación de un módulo de generación**

Recibirá los datos obtenidos por el módulo de detección y generará un fichero compatible con Waikato Environment for Knowledge Analysis (WEKA) para su análisis.

- **Estudio y análisis de los clasificadores que mejor se adecuen al problema**
  Se realizará un estudio de las técnicas de clasificación, teniendo en cuenta las distintas alternativas que proporciona la herramienta WEKA, para entrenar y encontrar aquel que obtenga el mayor porcentaje de clasificaciones positivas en la detección de vocales, a partir de los datos generados en el análisis. Para dicho análisis se comprobarán los resultados obtenidos con el conjunto de datos generado con RGB y con RGB-D por separado, para analizar cómo repercute el tipo de imagen. En este punto y en base a los resultados obtenidos, se realizará el análisis de los clasificadores mostrados en el estado del arte.
- **Diseño e implementación de un módulo de clasificación**
  Una vez definido el modelo de clasificador a utilizar, se realizará la integración del clasificador para su uso en el proyecto. Este módulo recibirá los datos enviados por el módulo de detección y será el encargado de, mediante la configuración obtenida en el estudio del clasificador en WEKA, obtener para los valores de entrada (posición de los puntos que conforman la boca) un resultado que indique la vocal que el paciente esté pronunciando.

## 1.4. Estructura del documento

A continuación y para facilitar la lectura del documento, se detalla el contenido de cada capítulo.

- En el **capítulo 1** se realiza la **introducción al proyecto** y las **razones que han motivado el trabajo**. También se detallan en este apartado los objetivos que se desean cumplir y la estructura del documento.
- En el **capítulo 2** se trata el **estado del arte**, haciendo mención a las aplicaciones o ideas existentes que se puedan asemejar a la solución y que han servido de base para el desarrollo del proyecto. También se detallan las **tecnologías y herramientas utilizadas** para la realización del trabajo.

- En el **capítulo 3** se elabora la descripción del proyecto, que incluye el **análisis**, indicando las **restricciones, casos de uso y requisitos del sistema** que se utilizarán como guía en el posterior apartado de diseño, así como el **diseño** de la solución, incluyendo diagramas que definan las **funcionalidades** a implementar y los mecanismos de **comunicación** entre los distintos elementos.
- En el **capítulo 4** se desarrollan las distintas decisiones tomadas a la hora de **implementar** el sistema, así como las **pruebas unitarias** y los resultados del **estudio de clasificadores**.
- En el **capítulo 5** se realiza la **experimentación** y **pruebas del sistema**, analizando los **resultados obtenidos**.
- En el **capítulo 6** se incluye todo lo referente a la **gestión del proyecto**, es decir, el modelo de desarrollo y la planificación utilizada, que detallan los elementos a tener en cuenta para la elaboración del proyecto. Además del presupuesto asignado para su realización al cual se deben ajustar los costes de personal, infraestructuras y material.
- En el **capítulo 7**, por último, se reflejan las **conclusiones** obtenidas al finalizar el proyecto y los posibles **trabajos futuros** a realizar en este ámbito.

# Capítulo 2

# Estado del arte

El objetivo de este capítulo es describir el marco teórico en el que se enmarca el proyecto, también hará mención de las alternativas ya existentes en el mercado y que han servido para encaminar el presente proyecto o asentar las bases del mismo. Así como las herramientas utilizadas para el desarrollo del trabajo.

## 2.1. Sistemas y soluciones similares

Como se ha comenzado a desarrollar en la introducción, los sistemas de reconocimiento de voz actuales se basan en el reconocimiento de sonidos. Sin embargo, existen estudios que buscan alternativas a estos sistemas, desarrollos que cubran necesidades o deficiencias no contempladas por los otros.

Ya en 1989 había interés en encontrar mecanismos, basándose en la capacidad del ser humano de leer los labios, para identificar las distintas fonéticas de las vocales en inglés a través de mecanismos audiovisuales y el uso de redes neuronales [3]. En esta ocasión quedó patente que el sistema era competente y capaz de cosechar resultados parejos a los demostrados por el ser humano.

Estudios más recientes basan el reconocimiento de voz a través de la detección visual del contorno de los labios [4]. Para ello hacen uso de dos mecanismos de detección, el primero y con menor exactitud, basa la identificación en base a los umbrales de color y diferenciación de todos entre piel y labios. El segundo, más eficaz, realiza la detec-

ción mediante el uso de modelos de localización, forma y tamaño de labios. El objetivo del estudio es generar un reconocedor visual que sea capaz de clasificar las vocales cortas y largas del inglés mediante mediante el uso de Máquinas de Sopote Vectorial (SVM). Finalizan el estudio afirmando que aunque la exactitud de la clasificación no sea suficiente para casos reales, sí que demuestra que el sistema basado en la detección del contorno es viable.

Otro de los estudios busca demostrar la viabilidad de un sistema de reconocimiento visual de voz, aplicado al reconocimiento de números [5]. En esta ocasión el artículo analiza los resultados obtenidos usando clasificadores basados en el algoritmo K-NN, concluyendo que la información visual no es suficiente para el reconocimiento del habla, pero que combinados con reconocimiento auditivo podría mejorar sustancialmente los sistemas actuales basados únicamente en el sonido.

En general, se han encontrado bastantes estudios relacionados con la temática que aborda el proyecto, pero se ha creído conveniente resumirlo en los aquí mostrados, ya que el objetivo y los resultados resultaban parejos.

## 2.2. Sistemas de reconocimiento facial

Un sistema de reconocimiento facial es un sistema que, de forma automática, es capaz de identificar una cara en una imagen o vídeo digital. Esto se hace mediante el análisis de las características de la cara y la comparación de estas con una base de datos [6].

En la mayoría de los casos, estos sistemas se utilizan para el reconocimiento e identificación de un sujeto dado, buscando coincidencias de las características de la cara visualizada con los datos registrados anteriormente en la base de datos. Sin embargo, para el caso que nos atañe, se pretende únicamente realizar la detección del rostro, y una vez diferenciada la cara dentro de la imagen, el sistema debe ser capaz de ubicar la boca del sujeto , con el objetivo de poder utilizar esta información para analizar los movimientos de la misma al pronunciar las vocales.

Durante el estudio y análisis de los detectores faciales se encontraron varias alternativas que se proceden a analizar a continuación.

### 2.2.1. Discrete Area Filter (DAF) Face Detector library

Esta fue la primera librería de detección analizada [7]. Es gratuita para uso académico y permite detectar hasta 15 puntos de la cara como se muestra en la Figura 2.1. Sin embargo el análisis de la herramienta y el estudio de los resultados muestra que, aunque sea capaz de realizar detecciones aceptables, el nivel de detalle para los puntos que forman la boca no es suficiente [8].

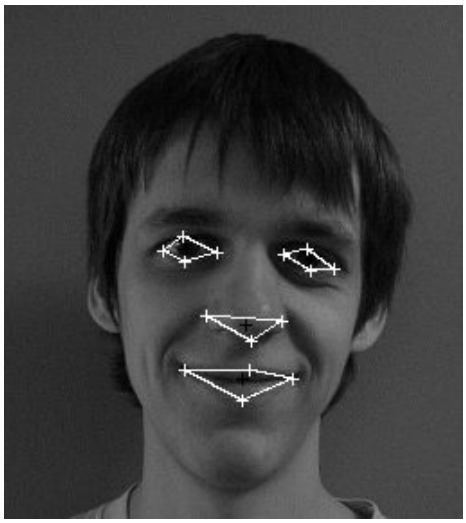

Figura 2.1: Ejemplo de detección con Semantic vision technologies

### 2.2.2. Active Shape Model library (asmlib)

Tras los primeros resultados de la búsquedas de detectores faciales compatibles con OpenCV y de código abierto se encontró esta librería [9]. En principio parecía cumplir con los requisitos deseados, ya que estaba en C++, compatible con OpenCV y de código abierto, pero analizando los ejemplos y pruebas se pudo apreciar que las detecciones no eran demasiado precisas (la localización de la boca está desviada y es más grande, Figura 2.2) y por tanto no cumplía con los requisitos deseados.

Figura 2.2: Ejemplo de detección con asmlib

### 2.2.3. IntraFace

Desarrollada por el Robotics Institute de la Universidad de Carnegie Mellon [10], esta librería dispone de un sistema de reconocimiento muy eficiente en condiciones de todo tipo. Sin embargo, el acceso a la última versión no está habilitado por encontrarse en fase de pruebas. Por ello, aunque los resultados y demostraciones vistas resulten atrayentes, no es viable esperar a incluirla en este proyecto.

### 2.2.4. Stasm

Esta es la librería de detección facial que se ha integrado en el proyecto [11]. Funciona bajo la licencia BSD, programada en C++ y compatible con OpenCV, por lo que encaja en los requisitos del proyecto para su uso.

Stasm permite el reconocimiento múltiple de caras, aunque para el caso del proyecto sólo sea necesaria la detección de una. Esta herramienta realiza la búsqueda de patrones y formas que se correspondan con una cara, en base a los modelos que tiene incorporados, por lo que las caras con inclinaciones, giradas o con expresiones exagera-

das empeoran los resultados [12]. También se pueden diseñar nuevos modelos de reconocimiento de cara, lo que dota a la herramienta de un interesante potencial de mantenibilidad.

Para la detección, basta con disponer de una imagen en formato OpenCV (Mat), hacer una conversión de la misma a monocromática y pasársela como argumento a la función de detección de Stasm. Con ello se obtiene una matriz de posicionamiento de los 77 puntos que es capaz de distinguir, como se puede apreciar en la Figura 2.3.

Figura 2.3: Ejemplo de detección con Stasm y los puntos que la conforman mostrado en [1]

Gracias a esto, permite la diferenciación de los puntos de la boca, compuesta por 18 puntos. Se espera que cuanto mayor sea el número de puntos que componen la boca, dote de mayor exactitud en la detección y que consiga mejores resultados en la clasificación.

## 2.3. Clasificadores

Un clasificador es un algoritmo de Inteligencia Artificial para poder asociar elementos de entrada a la clase o categoría correspondiente. Para ello, en primer lugar se debe entrenar al clasificador con un conjunto de datos lo suficientemente grande como para que el algoritmo sea capaz de discernir entre las distintas categorías. La fase de entrenamiento puede ser supervisada, no supervisada o semi-supervisada, y en las supervisadas se requiere del conocimiento

previo de las categorías a las que pertenecen los distintos elementos del conjunto de datos. Después se pasa a la fase de testeo, en la que se utiliza un conjunto de datos diferente o se usan mecanismos como la validación cruzada para comprobar los resultados del clasificador obtenido.

Para el estudio de los clasificadores se ha hecho uso de la herramienta WEKA, como se detallará más adelante, y se han analizado varias alternativas para su implementación. A continuación se muestran los distintos algoritmos analizados que disponen del clasificador escogido en el análisis.

### 2.3.1. RandomTrees

Algoritmo de clasificación basado en Random Forest implementado en OpenCV [13]. Se descartó el uso de este algoritmo porque hace uso de matrices de datos de OpencV y lo que ello conlleva, tratamiento de valores en 2D, cuando el objetivo principal del trabajo es el estudio y clasificación de valores en 3D.

### 2.3.2. random-forest

Implementación del algoritmo Random Forest en C++ desarrollada por Steffen Kirchhoff y Bjoern Andres [14]. Esta librería permite realizar el entrenamiento del clasificador con una gran cantidad de datos, pero se demora en exceso y es sólo válida para clasificaciones binarias de sólo 2 clases (0 y 1). Debido a este contratiempo el uso de esta librería también quedó descartado, ya que para el proyecto se requiere que sea capaz de distinguir y clasificar entre 5 categorías (a, e, i, o, u).

### 2.3.3. Waffles

Waffles es una herramienta disponible en C++ que provee de múltiples algoritmos para el análisis de datos, entre los que se encuentran varios modelos de clasificadores [15]. Esta librería permite realizar el entrenamiento, testeo y clasificación de los elementos que recibe, con multitud de opciones en la introducción de estos datos y sin restricciones en las categorías de clasificación. Además, es ca-

paz de importar los conjuntos de datos de ficheros WEKA de forma sencilla y rápida, por lo que aprovecha los ficheros generados para el análisis de clasificadores. También permite almacenar y cargar el modelo de clasificador en un fichero JSON [16], gracias a esta funcionalidad, evita al sistema estar generando un clasificador en cada ejecución y lo almacena para su posterior carga.

## 2.4. Mini

Mini es un robot creado por el grupo RoboticsLab de la Universidad Carlos III de Madrid, dentro del proyecto RobAlz y en colaboración con la Fundación Alzheimer España (FAE), que pretende cubrir las necesidades existentes dentro del ámbito de los robots sociales y de asistencia a personas. El objetivo de Mini no es sustituir a los cuidadores, sino servir de herramienta o ayuda a los mismos. Los usos que se pretenden dar a Mini son los siguientes: seguridad, asistente personal, entretenimiento, estimulación.

Es un robot de escritorio ideado para la interacción cercana con el usuario (inferior a 2m), su tamaño no supera los 50cm de alto, como se puede observar en la Figura 2.4. Dispone de numerosos sensores que favorecen la interacción con los pacientes, como el sensor Kinect, varios táctiles y un sistema de reconocimiento de voz básico. Tiene una apariencia cercana y amigable gracias a los actuadores y dispositivos luminosos que lo dotan de expresividad. Mini tiene una arquitectura de control modular, organizada en diferentes niveles e implementada en ROS.

Figura 2.4: Robot social Mini

## 2.5. Sensor Microsoft Kinect

Aunque el desarrollo no está ligado a este sensor gracias al sistema ROS basado en nodos y *topics*, que se detallará más adelante en el punto 2.6.1, se ha optado por hacer uso de este dispositivo por su eficiencia y bajo coste, que permite su implantación en multitud de escenarios posibles [17].

El sensor Microsoft Kinect, en su origen, es un controlador de juego desarrollado por Microsoft para controlar e interactuar con la consola Xbox 360. Sin embargo, debido a la liberación del código del controlador y los bajos costes del dispositivo en comparación con otros de su misma calidad, han hecho que sea muy utilizado entre la comunidad investigadora y los desarrolladores domésticos. Cuenta con una cámara RGB y un sensor de profundidad que servirán para captar las imágenes y los datos necesarios para cumplir con el objetivo del trabajo.

## 2.6. Análisis de tecnologías utilizadas

Todas las tecnologías que se han integrado o han intervenido en el desarrollo del proyecto se describen en los siguientes apartados. Con ello se pretende favorecer la comprensión del resto del documento y el funcionamiento de los elementos utilizados. Se ha hecho especial hincapié en la búsqueda y uso de herramientas de código libre o gratuitas para no incurrir en licencias o costes extras en el desarrollo.

### 2.6.1. ROS - Robot Operating System

ROS[18] es una colección de frameworks de código abierto para el desarrollo de software de robots que provee funcionalidades similares a un sistema operativo. Fue desarrollado inicialmente en 2007 por el Laboratorio de Inteligencia Artificial de Stanford y en 2008 este desarrollo fue continuado por Willow Garage, un instituto de investigación de robótica formado por más de veinte instituciones.

El sistema funciona a través de nodos distribuidos que se coordinan entre sí para realizar tareas más complejas. Como se muestra en la Figura 2.5, estos nodos se comunican mediante un sistema de envío (publicación) y recepción (suscripción) de mensajes a través de *topics* de forma que puedan compartir cualquier tipo de flujo de datos disponible en el sistema.

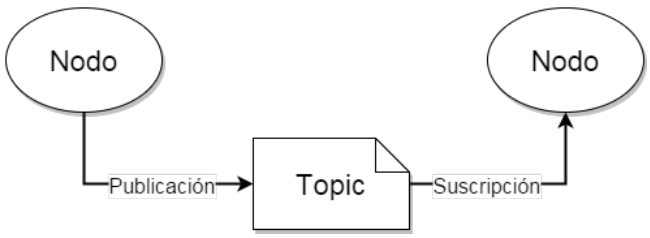


Figura 2.5: Comunicación en ROS

Dispone de librerías para el desarrollo de los nodos en varios lenguajes como Java, Lisp, Lua, C++ y Python, siendo estos dos últimos los estables.

Para el desarrollo de este proyecto se optó por la versión de ROS Groovy Galapagos. Esta decisión se llevó a cabo debido a que Mini y la mayoría de robots del grupo de investigación funcionan en esta

versión y ROS no asegura retrocompatibilidad entre las distintas versiones. Por ello y para asegurar el correcto funcionamiento de la solución, se realizaron todos los módulos en esta versión de ROS, aunque no sea la más moderna.

ROS dispone de múltiples librerías para el desarrollo de los módulos que permiten su integración con el sistema. Sin embargo, por compatibilidad y estabilidad, este listado se redujo a *roscpp* (C++) y *rospy* (Python). Se decidió hacer uso de la librería *roscpp* dada su mayor eficiencia y extensión, ya que uno de los requisitos del sistema son los bajos tiempos de respuesta. Esto, sumado al resto de módulos ya implementados para Mini en C++ disponibles en el grupo de investigación, facilita la gestión y mantenimiento de los distintos elementos desarrollados en el trabajo.

Además, ROS dispone de un sistema de *bags* mediante el cual se pueden grabar todos los *topics* y mensajes existentes durante una sesión con el robot, permitiendo su ejecución a modo de simulación y facilitando enormemente las tareas de testeo. Este sistema ha sido el utilizado para la mayor parte del desarrollo, ya que permite simular una sesión sin necesidad de encontrarse en el laboratorio.

Para la generación y compilación de proyectos en C++ se utiliza habitualmente un gestor de paquetes que simplifica y sirve de guía en la realización de estas tareas. CMake es uno de estos gestores, que haciendo uso de ficheros de configuración sencillos e independientes de la plataforma, es capaz de generar y compilar los proyectos deseados.

Catkin es una colección de macros de CMake utilizada para la generación de paquetes ROS, que simplifica la configuración de los ficheros de descripción del proyecto o programa ROS para su posterior compilación. Esta herramienta incluye todas las dependencias descritas en el fichero de descripción del proyecto, permitiendo asociar las librerías disponibles en ROS sin necesidad de realizar difíciles configuraciones y creando las relaciones del entorno para que el sistema tenga en cuenta y sea capaz de ejecutar los nuevos desarrollos.

### 2.6.2. Point Cloud Library

*Point Cloud Library* (PCL)[19] es una librería independiente bajo licencia *Berkeley Software Distribution* (BSD), escalable y de código abierto que permite procesar imágenes y nubes de puntos. Gracias

a esta librería se puede convertir la nube de puntos recibida por el sensor Kinect a un formato legible y apto para el programa. Dicha conversión genera una matriz de elementos de tipo Punto que contienen cada una de las componentes x, y, z.

También, aunque no se haya hecho uso de ellos en el proyecto, PCL dispone de varios módulos y filtros que al igual que para el caso de la conversión, simplifican las tareas de manipulación y tratamiento de imágenes.

### 2.6.3. OpenCV

OpenCV[20] es una librería libre de visión artificial desarrollada inicialmente por Intel y publicada bajo licencia BSD. Es un proyecto que proporciona un entorno de desarrollo fácil y altamente eficiente, además de integrar multitud de funciones para el cómputo y manipulación en el proceso de visión.

Las funciones incluidas en OpenCV podían servir para la realización del proyecto, pero esto implicaba el desarrollo completo de un módulo de detección, cosa que no formaba parte del objetivo del trabajo, y por este motivo se decidió realizar la búsqueda de una librería ya implementada.

### 2.6.4. WEKA

WEKA[21] es una herramienta enfocada al estudio y análisis del aprendizaje automático y la minería de datos. Contiene una gran colección de herramientas para el preprocesamiento de datos y su modelado, así como multitud de algoritmos para el análisis de datos o predictivo.

En el caso del presente proyecto, se ha utilizado para realizar el estudio y análisis de los distintos clasificadores disponibles en la herramienta, para así obtener el que mejor se ajusta a la problemática. Los clasificadores evaluados han sido los siguientes:

- **J48**, implementación del algoritmo C4.5 en Java basado en los árboles de decisión.
- **LibSVM**, es una librería que implementa las Máquinas de Soporte Vectorial. Es un sistema de clasificación basado en la

división mediante planos de un conjunto de datos de entrenamiento, generando así las zonas que corresponden a cada una de las clases disponibles en la clasificación. Una vez hecho esto, basándose en los cálculos de proximidad con respecto al dato de entrada, asocia o clasifica con la clase correspondiente.

- **MultilayerPerceptron**, es un tipo de redes de neuronas artificiales formada por múltiples capas. Se caracteriza por tener tres zonas, o capas, claramente diferenciadas: capa de entrada, capas ocultas y capa de salida.
- **BayesNet**, es un tipo de modelo probabilístico representado por un grafo. Los elementos se conectan entre sí siempre y cuando sean dependientes unos de otros, y conforman así el grado de probabilidad de cada situación o variable.
- **NaivesBayes**, similar a BayesNet, este modelo probabilístico se caracteriza por aplicar hipótesis simplificadoras a las variables predictoras.
- **OneR**, es un algoritmo de clasificación que selecciona la regla de decisión que genera menores errores de predicción.
- **IBk**, es una implementación del algoritmo k-NN. Este algoritmo de clasificación asigna la clase más frecuente a la que pertenecen los K vecinos más cercanos al elemento introducido.
- **Random Forest**, es una combinación de árboles predictores que permite gestionar grandes cantidades de variables y generar un clasificador muy certero.

# Capítulo 3

# Descripción del sistema

## 3.1. Análisis

En el análisis del sistema se detalla el proceso de descubrimiento, refinamiento, modelado y especificación de los elementos que se desean desarrollar. Permite especificar las características operacionales del proyecto y las restricciones que debe cumplir. Así pues, este apartado contiene información acerca de las restricciones principales y el entorno en que debe operar el sistema, los casos de uso que se deben satisfacer y los requisitos finales a los que debe ceñirse el desarrollo.

### 3.1.1. Descripción de las características funcionales

Como se ha indicado en la introducción, el objetivo de este trabajo es diseñar e implementar un sistema de reconocimiento visual de vocales mediante la fusión de información 2D y 3D para el robot Mini.

Para cumplir con este objetivo, el sistema debe poseer una serie de funcionalidades básicas. Primero debe ser capaz de identificar y situar la boca de un sujeto dentro de una imagen o vídeo captado por el sensor Kinect del robot Mini. Una vez hecho esto, el sistema debe ser capaz de analizar los distintos puntos que corresponden a la boca y asociarlos con su correspondiente valor de la nube de puntos, o lo que es lo mismo, realizar el paso de datos 2D a valores 3D (ver Figura 3.1).

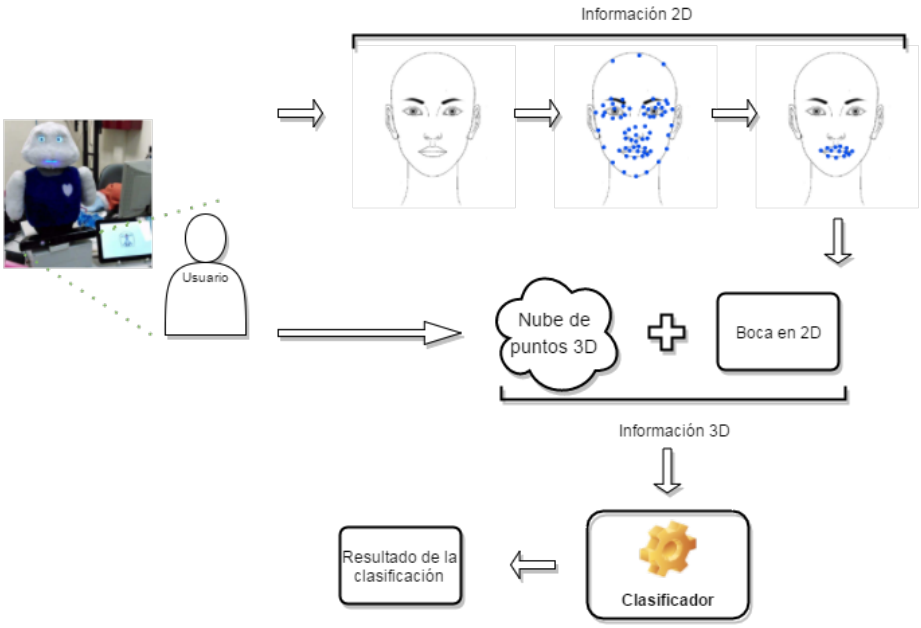


Figura 3.1: Sketch del funcionamiento

Finalmente, el sistema debe ser capaz de clasificar, en base a los puntos obtenidos en el paso anterior, la vocal correspondiente a la que el usuario estaba pronunciando.

### 3.1.2. Restricciones del sistema

El proyecto debe seguir una serie de restricciones que especifican y delimitan el desarrollo del sistema. Estas restricciones se realizarán en líneas generales, ya que el análisis minucioso forma parte de la especificación de requisitos.

- El sistema debe funcionar bajo ROS.
- El sistema debe ser compatible con la versión de ROS Groovy.
- La finalidad del sistema es la de reconocer la vocal pronunciada por el usuario mediante imágenes o vídeos captados por el robot.
- El sistema debe ser capaz de reconocer los puntos de la boca.
- El sistema debe clasificar las vocales con un nivel de acierto aceptable (superior al 80 % de tasa de aciertos).

- El lenguaje utilizado debe ser Python o C++, ya que son las dos librerías de clientes estables.
- El sistema debe permitir compartir la información con otros nodos de ROS para que pueda ser reutilizado.

### 3.1.3. Entorno operacional

A continuación se detallan las configuraciones y elementos necesarios para la realización del proyecto.

- Entorno operacional:
  - Se requerirá de un equipo con sistema operativo Ubuntu 12.04 LTS.
  - La versión de ROS debe ser Groovy Galapagos.
  - Se debe disponer de un sensor Microsoft Kinect conectado al robot.
  - WEKA 3.7 con gestor de complementos y la función LibSVM para el análisis del clasificador Máquinas de Soporte Vectorial.

### 3.1.4. Especificación de casos de uso

Los casos de uso se encargan de dar a conocer y plasmar los comportamientos del sistema con los actores involucrados en su uso. Para ello, en primer lugar se deben detallar los actores que intervienen en los casos de uso y su rol en el uso del sistema:

- Robot: en este proyecto el robot hace mención a las comunicaciones realizadas entre el programa del proyecto y Mini, que es el robot implicado en el desarrollo. Estos dos elementos son los encargados de captar mediante el sensor Microsoft Kinect la imagen del usuario y analizar y clasificar la vocal correspondiente.
- Usuario: es la persona que pronuncia las vocales delante del robot. Únicamente interactúa con el sistema al pronunciar las vocales delante del robot y el sensor de imagen.

### Diagramas de casos de uso

Los diagramas de caso de uso sirven para visualizar de forma sencilla y directa los actores involucrados en el uso del sistema, así como las tareas que realizarán durante la ejecución del software. Como se ha indicado en el apartado anterior, para este proyecto únicamente intervienen tres actores y sólo hay una secuencia de casos de uso, como se puede observar en la Figura 3.2. Esto se debe a que el programa desarrollado tiene un carácter de ejecución secuencial y una limitada interactividad, por lo que con un diagrama basta para reflejar su único proceso.

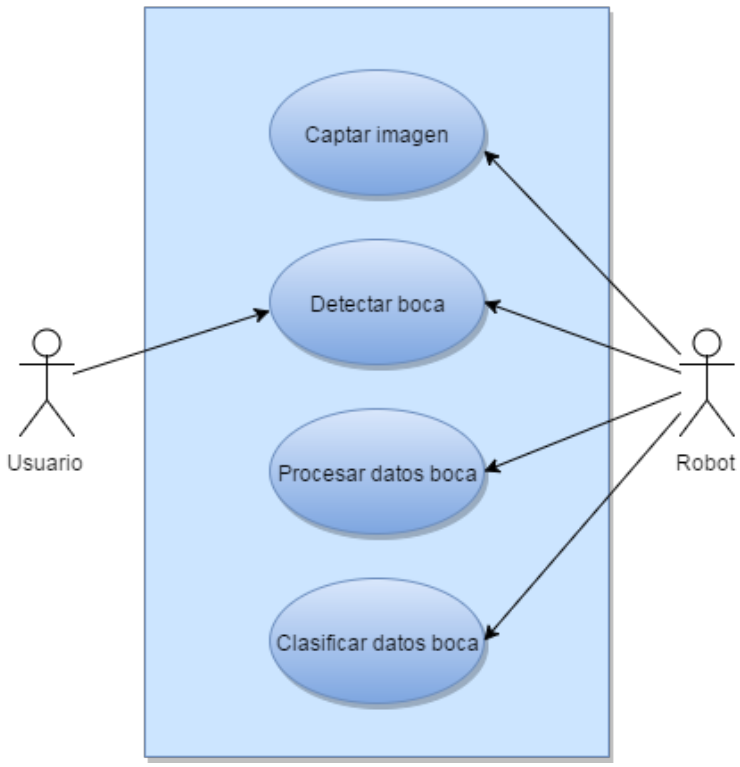


Figura 3.2: Diagrama de casos de uso

### Descripción textual de los casos de uso

A partir del diagrama de casos de uso reflejado en la Figura 3.2 se pueden obtener los siguientes casos de uso que pasarán a describirse en esta sección. Cada caso de uso se incluirá en una tabla independiente con los atributos que correspondan, realizando una

descripción mucho más detallada de los casos de uso que determinan el desarrollo de las actividades y sus actores.

| Caso de uso | |
|---|---|
| **Código** | CU-001 |
| **Nombre** | Captar imagen |
| **Actor** | Robot |
| **Descripción** | El robot comienza a capturar las imágenes recibidas a través del sensor Microsoft Kinect. Estas imágenes incluyen información 2D y 3D. |
| **Precondiciones** | • El sistema debe haber sido iniciado. |
| **Curso de eventos** | 1. El operador inicia el sistema.<br>2. Se activa la comunicación con el sensor Kinect. |
| **Efectos** | • El robot comienza a recibir imágenes capturadas por el sensor. |

Tabla 3.1: Caso de uso CU-001

| Caso de uso | |
|---|---|
| **Código** | CU-002 |
| **Nombre** | Detectar boca |
| **Actor** | Robot, Usuario |
| **Descripción** | El robot, gracias a las imágenes capturadas a través del sensor, realiza la búsqueda y detección de la boca del usuario. |
| **Precondiciones** | • El robot debe estar captando imágenes.<br>• El usuario debe estar mirando de frente al robot.<br>• El usuario debe pronunciar una vocal gesticulando y de forma clara. |
| **Curso de eventos** | 1. El robot capta al usuario pronunciando la vocal y envía la información.<br>2. Se realiza la búsqueda y detección de la boca del usuario.<br>3. Se extraen los puntos pertenecientes a la boca. |
| **Efectos** | • Se dispone de los datos 2D de la boca del usuario. |

Tabla 3.2: Caso de uso CU-002

| Caso de uso | |
|---|---|
| **Código** | CU-003 |
| **Nombre** | Procesar datos de la boca |
| **Actor** | Robot |
| **Descripción** | El robot, en base a los datos recibidos y tras la detección de la boca, fusiona o asocia el contenido 2D con la correspondiente nube de puntos (3D) que capta el sensor. |
| **Precondiciones** | • El usuario debe haber pronunciado una vocal.<br>• El robot debe detectar la boca del usuario. |
| **Curso de eventos** | 1. Se reciben los datos en 2D.<br>2. Se detecta la boca.<br>3. Se comprueba que existen datos en 3D.<br>4. Se fusionan los datos 2D con los datos 3D. |
| **Efectos** | • El robot genera el listado de puntos que conforman la boca en el entorno 3D y prepara su envío para el resto de módulos. |

Tabla 3.3: Caso de uso CU-003

| Caso de uso | |
|---|---|
| **Código** | CU-004 |
| **Nombre** | Clasificar los datos obtenidos de la boca |
| **Actor** | Robot |
| **Descripción** | El robot realiza una clasificación de los datos obtenidos en la detección y procesado de la boca para asociar la vocal correspondiente. |
| **Precondiciones** | • El robot debe haber procesado los datos de la boca. |
| **Efectos** | • El robot devuelve el valor obtenido durante la clasificación. |
| **Curso de eventos** | 1. El robot procesa los datos de la boca [CU-005].<br>• El robot clasifica los datos obtenidos.<br>• El robot devuelve el valor asociado durante la clasificación. |

Tabla 3.4: Caso de uso CU-004

### 3.1.5. Especificación de requisitos

Los requisitos son los elementos encargados de delimitar las funcionalidades y las características del sistema a desarrollar. Existen dos tipos de requisitos en base a esto: funcionales y no funcionales.

Por tanto en este apartado se realiza un estudio de los requisitos previos al diseño de la solución y que sirven de guía para los desarrollos posteriores.

### 3.1.6. Requisitos funcionales

Establecen los comportamientos, las funcionalidades y los objetivos específicos que el sistema debe cumplir en determinadas situaciones. Sirven además para describir los servicios que hay que proporcionar y se han descrito en los casos de uso.

| **Requisito software funcional** | | | |
|---|---|---|---|
| **Código** | RF-001 | **Fuente** | CU-001, CU-002 |
| **Nombre** | Reconocimiento facial | | |
| **Descripción** | El software debe ser capaz de detectar la cara del usuario que tiene en frente (sólo uno). | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.5: Requisito funcional RF-001

| **Requisito software funcional** | | | |
|---|---|---|---|
| **Código** | RF-002 | **Fuente** | CU-002 |
| **Nombre** | Reconocimiento de la boca | | |
| **Descripción** | El software debe ser capaz de posicionar la boca como parte de la cara del usuario. Además debe obtener los puntos que la forman. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.6: Requisito funcional RF-002

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-003 | **Fuente** | CU-001 |
| **Nombre** | Recepción de los datos de la imagen 2D | | |
| **Descripción** | El software debe recibir la imagen 2D del sensor Kinect del robot. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Alta | **Verificabilidad** | Alta |

Tabla 3.7: Requisito funcional RF-003

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-004 | **Fuente** | CU-001 |
| **Nombre** | Recepción de los datos de la imagen 3D | | |
| **Descripción** | El software debe recibir la imagen 3D del sensor Kinect del robot. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Alta | **Verificabilidad** | Alta |

Tabla 3.8: Requisito funcional RF-004

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-005 | **Fuente** | CU-003 |
| **Nombre** | Fusión de datos en 2D y datos en 3D | | |
| **Descripción** | El software debe ser capaz de, una vez posicionada la boca, realizar el emparejamiento de los datos obtenidos en 2D con los datos de profundidad (3D). | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.9: Requisito funcional RF-005

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-006 | **Fuente** | CU-003, CU-004 |
| **Nombre** | Generación de ficheros para el estudio de clasificadores | | |
| **Descripción** | El software debe generar los ficheros a utilizar en WEKA para realizar el estudio de los clasificadores. | | |
| **Necesidad** | Deseable | **Prioridad** | Alta |
| **Estabilidad** | Media | **Verificabilidad** | Alta |

Tabla 3.10: Requisito funcional RF-006

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-007 | **Fuente** | CU-003 |
| **Nombre** | Generación y envío de mensaje de detección | | |
| **Descripción** | El software debe generar y enviar el mensaje con los datos obtenidos en la detección. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.11: Requisito funcional RF-007

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-008 | **Fuente** | CU-004 |
| **Nombre** | Generación y envío de mensaje de clasificación | | |
| **Descripción** | El software debe generar y enviar el mensaje con el resultado obtenido en la clasificación. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.12: Requisito funcional RF-008

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-009 | **Fuente** | CU-003 |
| **Nombre** | Resolución de valores atípicos en la nube de puntos (3D) | | |
| **Descripción** | El software debe ser capaz de lidiar con posibles errores en la nube de puntos 3D. | | |
| **Necesidad** | Deseable | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.13: Requisito funcional RF-009

| Requisito software funcional | | | |
|---|---|---|---|
| **Código** | RF-010 | **Fuente** | CU-004 |
| **Nombre** | Reproducción del resultado de la clasificación | | |
| **Descripción** | Una vez obtenido el resultado de la clasificación, se envía un mensaje al robot para que reproduzca la vocal resultante. | | |
| **Necesidad** | Opcional | **Prioridad** | Baja |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.14: Requisito funcional RF-010

### 3.1.7. Requisitos no funcionales

Los requisitos no funcionales abarcan todos aquellos requisitos que, sin definir el comportamiento o funcionalidades de un sistema, se usan para restringir características del diseño, estándares de calidad o rendimiento.

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-001 | **Fuente** | Cliente |
| **Nombre** | Compatibilidad con Mini | | |
| **Descripción** | El software a desarrollar debe estar preparado para recibir e interactuar con los topics de Mini. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.15: Requisito no funcional RNF-001

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-002 | **Fuente** | Cliente |
| **Nombre** | Compatibilidad con ROS Groovy Galapagos | | |
| **Descripción** | El software a desarrollar debe ser compatible con ROS y su versión Groovy Galapagos dado que es la versión instalada en Mini | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.16: Requisito no funcional RNF-002

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-003 | **Fuente** | Analista |
| **Nombre** | Plataforma estable y de alto rendimiento. | | |
| **Descripción** | El software a desarrollar debe desarrollarse en una plataforma estable y de alto rendimiento. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.17: Requisito no funcional RNF-003

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-004 | **Fuente** | Cliente |
| **Nombre** | Sistema operativo Ubuntu 12.04 LTS | | |
| **Descripción** | El software debe funcionar en el sistema operativo Ubuntu 12.04 LTS. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.18: Requisito no funcional RNF-004

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-005 | **Fuente** | Analista |
| **Nombre** | El clasificador debe ser eficiente | | |
| **Descripción** | El clasificador a desarrollar en el proyecto debe ser eficiente, logrando realizar la clasificación con un alto porcentaje de aciertos (superior al 80 %). | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.19: Requisito no funcional RNF-005

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-006 | **Fuente** | Cliente |
| **Nombre** | Diseño modular de la solución | | |
| **Descripción** | La solución debe ser un sistema modular diferenciado para permitir la reutilización de los módulos en otros desarrollos. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.20: Requisito no funcional RNF-006

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-007 | **Fuente** | Analista |
| **Nombre** | Uso de las librerías de OpenCV y PCL | | |
| **Descripción** | El software debe hacer uso de las librerías de OpenCV y PCL para la manipulación y tratamiento de las imágenes y vídeos recibidos por el sensor Kinect. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.21: Requisito no funcional RNF-007

| Requisito software no funcional | | | |
|---|---|---|---|
| **Código** | RNF-008 | **Fuente** | Analista |
| **Nombre** | Mensajes personalizados para la comunicación en ROS | | |
| **Descripción** | Para la comunicación entre los distintos módulos se deben utilizar los mensajes personalizados que el analista diseñe para cada caso. | | |
| **Necesidad** | Esencial | **Prioridad** | Alta |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

Tabla 3.22: Requisito no funcional RNF-008

### 3.1.8. Matriz de trazabilidad

Para finalizar el análisis, la matriz de trazabilidad, Tabla 3.23, simplifica las relaciones existentes entre los casos de uso y los requisitos funcionales, asegurando la correspondencia entre ellos a lo largo del ciclo de vida del proyecto. Esto significa que una vez completada la matriz, la gran mayoría de los casos de uso deberían tener asociado requisito que lo satisfaga. También sirve para, en caso de realizar cambios en el desarrollo, ver de forma sencilla a qué elementos del proyecto afectan estos cambios.

**Resumen Casos de uso**

CU-001: Captar imagen
CU-002: Detectar boca
CU-003: Procesar datos de la boca
CU-004: Clasificar los datos obtenidos de la boca

**Resumen Requisitos funcionales**

RF-001: Reconocimiento facial
RF-002: Reconocimiento de la boca
RF-003: Recepción de los datos de la imagen 2D
RF-004: Recepción de los datos de la imagen 3D
RF-005: Fusión de datos en 2D y datos en 3D
RF-006: Generación de ficheros para el estudio de clasificadores
RF-007: Generación y envío de mensaje de detección
RF-008: Generación y envío de mensaje de clasificación
RF-009: Resolución de valores atípicos en la nube de puntos (3D)
RF-010: Reproducción del resultado de la clasificación

| | CU-001 | CU-002 | CU-003 | CU-004 |
|---|---|---|---|---|
| **RF-001** | X | X | | |
| **RF-002** | | X | | |
| **RF-003** | X | | | |
| **RF-004** | X | | | |
| **RF-005** | | | X | |
| **RF-006** | | | X | X |
| **RF-007** | | | X | |
| **RF-008** | | | | X |
| **RF-009** | | | X | |
| **RF-010** | | | | X |

Tabla 3.23: Matriz de trazabilidad entre Casos de Uso y Requisitos Funcionales

## 3.2. Diseño

En el apartado de diseño se muestran los gráficos y decisiones tomadas para definir el sistema a desarrollar. Dado que es el paso previo a la implementación, todas las decisiones llevadas a cabo en este apartado se verán directamente reflejadas en el desarrollo de la solución. Inicialmente se muestra una vista general del proyecto para ir pasando al desglose de los elementos que lo componen.

### 3.2.1. Descripción general de la solución

El sistema esta formado por tres módulos independientes, detección, generación y clasificación, que interactúan entre sí para formalizar los requisitos definidos en el análisis. Estos módulos se envían la información mediante mensajes publicados en un topic específico de ROS, para disponer cada uno de los datos que se hayan generado en los pasos previos. Por tanto, el sistema debe ser capaz de, recibido un mensaje del robot con los datos de la imagen obtenida mediante el sensor Kinect, analizarlo y detectar los puntos que conforman la boca (módulo de detección). Una vez hecho esto, el módulo de detección debe generar el mensaje con los puntos obtenidos y enviarlo para que los otros dos módulos puedan recibirlos. El módulo de generación recibe el mensaje generado por el de detección y lo adecua para almacenarlo en un fichero con el formato compatible para WEKA. Con este mismo mensaje, el módulo de clasificación realiza los cálculos pertinentes para asociar los valores al resultado más conveniente y mostrarlo. En la Figura 3.3 se pueden apreciar mejor los elementos aquí descritos y cómo interactúan entre sí. El recuadro gris alberga la solución descrita, incluyendo todos los módulos a desarrollar y los mensajes que generan.

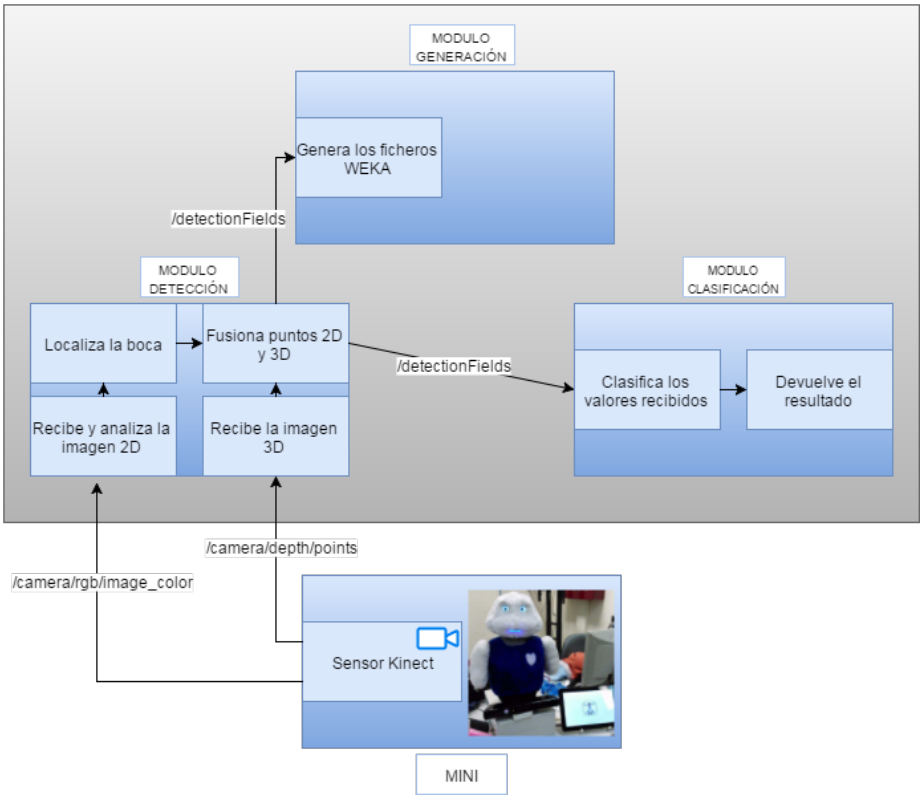


Figura 3.3: Diagrama general de la arquitectura de la solución

### 3.2.2. Arquitectura detallada del sistema

Realizada la descripción general de la solución, se pasa a detallar la arquitectura del sistema de forma más precisa. Para ello el sistema se desglosa en los módulos a realizar y cuyas funcionalidades están claramente delimitadas.

#### Módulo de detección

Como se observa en la Figura 3.3, el módulo de detección tiene dos procesos de escucha para la recepción de la imagen RGB y la nube de puntos recibidas por el sensor Kinect.

Este módulo recibe y trata los mensajes enviados por el robot, por tanto es el encargado de, a partir de la imagen en 2D, identificar la boca del usuario mediante el uso de la librería Stasm. Esta

librería requiere que la imagen sea monocromática, por lo que se debe hacer la conversión para que realice la detección de la boca. Una vez obtenidos los 18 puntos que es capaz de diferenciar la librería, esta información se asocia con los valores de la nube de puntos correspondientes, o lo que es lo mismo, fusiona la información de la imagen en 2D con la imagen 3D, aumentando la exactitud o calidad de la identificación. Asociados los puntos, el módulo finaliza generando un mensaje con toda la información de que se dispone hasta el momento, para que los módulos que usen las funcionalidades de éste puedan continuar las tareas con toda la información existente hasta el momento. El proceso queda detallado en la Figura 3.4.

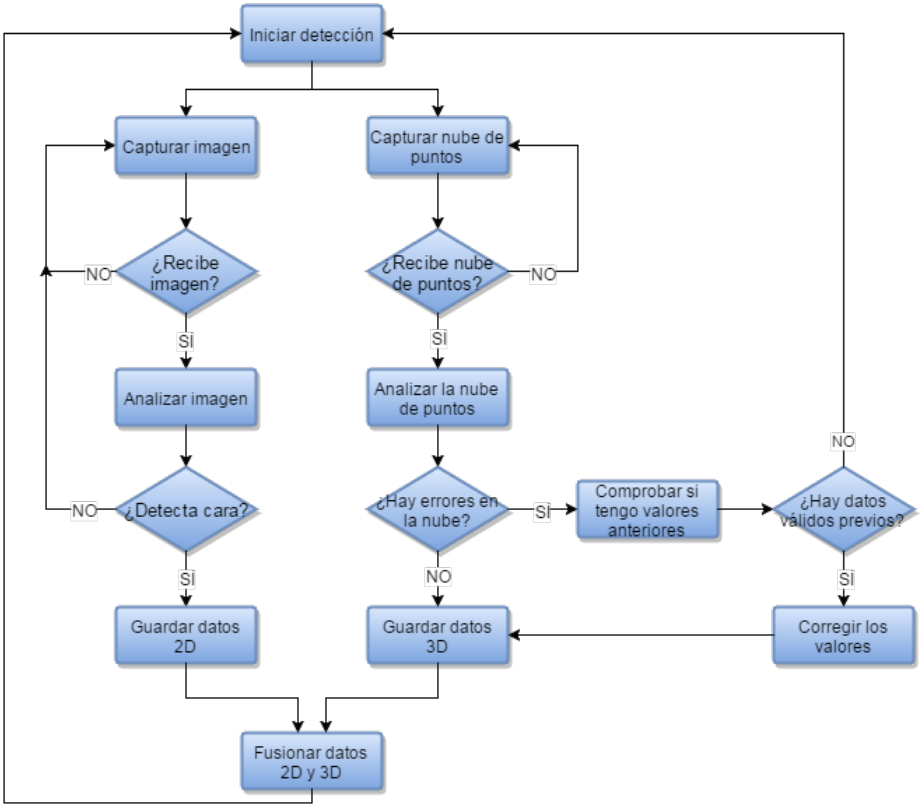


Figura 3.4: Diagrama de flujo del módulo de detección

### Módulo de generación

Los mensajes generados en el apartado anterior son recibidos por el módulo de generación (Figura 3.3). Éste se encarga de insertarlos en un fichero compatible con WEKA, con la estructura deseada para

su posterior uso.

Este módulo, como muestra la Figura 3.5, captura los mensajes enviados por el módulo de detección y se encarga de incluir todos los puntos que forman la cara de cada mensaje recibido (cada punto contiene las coordenadas x, y, z) dentro del fichero y se le asigna la clasificación correspondiente. Con ello se preparan los datos para su posterior uso en WEKA y el estudio del clasificador con mayor tasa de acierto.

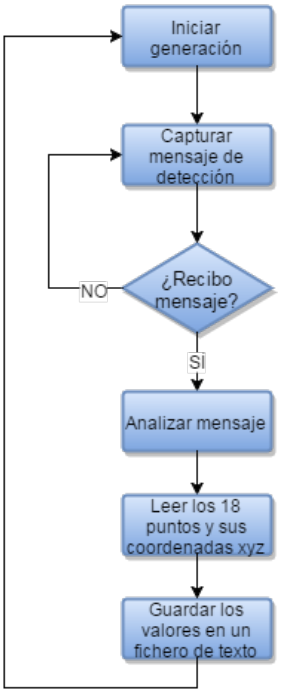


Figura 3.5: Diagrama de flujo del módulo de generación

**Módulo de clasificación**

Por último, este módulo es el encargado de recibir los mensajes del módulo de detección y asignarles una clase representativa en base al estudio de los clasificadores realizado en WEKA, como puede apreciarse en la Figura 3.3. Los datos utilizadas para el estudio confrontan los clasificadores bajo un entorno 2D, con un entorno 3D generado mediante la fusión de los datos recibidos por el sensor Kinect.

El módulo por tanto es capaz de generar un modelo de este clasi-

ficador en base a los datos introducidos en el entrenamiento y almacenarlo, evitando la generación del mismo en cada ejecución. Una vez llega el mensaje del módulo de detección, realiza la clasificación del mensaje en base a este modelo y finalmente, el resultado de dicha clasificación se muestra al operador del robot para que compruebe su validez, como queda detallado en la Figura 3.6.

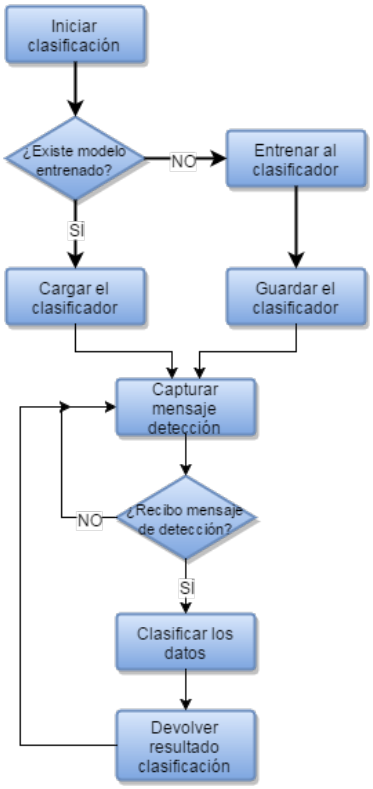


Figura 3.6: Diagrama de flujo del módulo de clasificación

# Capítulo 4

# Implementación y pruebas unitarias

El último hito del desarrollo del sistema consta de dos apartados muy relacionados entre sí, la implementación y las pruebas unitarias, y con los cuales se dá por finalizado el desarrollo de la solución. En primer lugar se detallan los elementos necesarios para construir el entorno del sistema y las distintas decisiones tomadas durante la implementación. Después, se realiza un análisis de las pruebas unitarias y su resultado, para poder comprobar mediante la matriz de trazabilidad que todos los requisitos han quedado cubiertos en el desarrollo.

## 4.1. Implementación

Para la implementación del sistema y asegurar la compatibilidad con Mini y su versión de ROS (Groovy Galapagos), es necesario disponer de una máquina con sistema operativo Ubuntu 12.04 LTS. Se deben instalar también la versión de ROS indicada y configurar el entorno de trabajo, además de ser necesarias las configuraciones de varias librerías que se usan en los distintos módulos. Estas librerías son las encargadas de realizar tareas como la detección de la cara o la clasificación siguiendo el modelo Random Forest. Todo este proceso queda detallado en el Anexo B. Una vez realizadas estas configuraciones se pasa a detallar la implementación de los distintos módulos en base al diagrama de clases de la Figura 4.1.

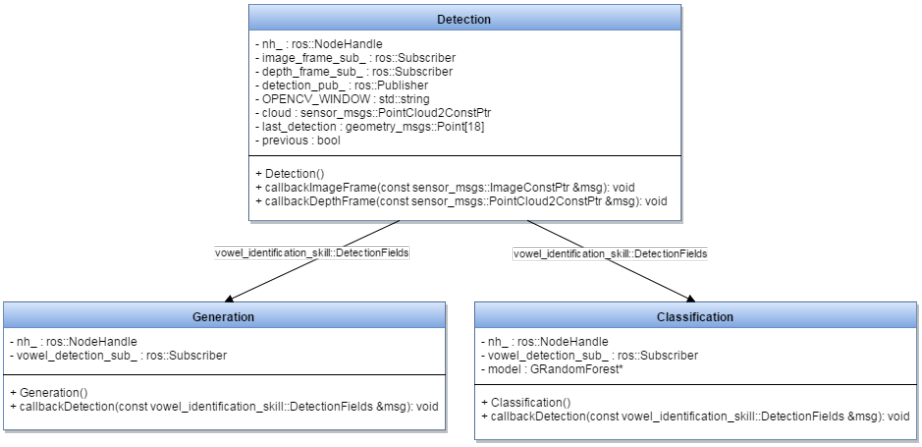


Figura 4.1: Diagrama de clases de la solución

### 4.1.1. Módulo de detección

El módulo de detección es el encargado de recibir la información de los sensores, la imagen en RGB y la nube de puntos de profundidad. Para ello, se crea una suscripción a los *topic* (ver sección 2.6.1) correspondientes para cada uno de los elementos que se desea obtener de los sensores y una función a ejecutar en caso de que reciba esta información.

Las funciones que forman parte de este módulo y que han sido implementadas son:

- ***callbackDepthFrame***
  Puesto que los datos recibidos por el sensor no son enviados de forma simultánea ni conjunta, se requiere de una función dedicada a la recepción de la nube de puntos. Esta función se encarga de ir actualizando el valor de la variable compartida donde se almacenan los distintos valores de la nube. De esta forma será accesible para la función dedicada a la recepción de imágenes RGB cuando precise.

- ***callbackImageFrame***
  Esta es la función de recepción de imágenes RGB, dado que es la encargada de realizar la detección de la boca, pero requiere de la existencia de la nube de puntos para realizar la fusión de los datos, se comprueba cada vez que se reciben imágenes que se haya compartido y actualizado la nube de puntos. Para que la detección se pueda llevar a cabo, se precisa realizar una

conversión de la imagen a monocromática, ya que la librería Stasm sólo acepta este tipo de imágenes para la detección. Si no identifica un rostro se espera a la siguiente imagen, en caso positivo, se procede a analizar la imagen recibida. Dado que Stasm permite la identificación de 77 puntos distintos de la cara, 18 de ellos pertenecientes a la boca, el sistema se centra únicamente en éstos últimos.

Llegado este punto, el módulo convierte a un formato legible la nube de puntos con la librería PCL, pudiendo acceder a las coordenadas xyz de cada elemento de la nube. Debido a errores de registro por parte del sensor Kinect, en ocasiones los puntos que componen la nube vienen determinados por un *nan* (not a number) e invalidarían el registro. Sin embargo, para poder aprovechar un mayor número de imágenes por segundo recibidas, se ha implementado un sistema de corrección de valores para subsanar estos números atípicos. El sistema almacena y actualiza una variable con los puntos válidos, y en caso de que se reciba un valor erróneo, se intenta corregir sustituyéndolo.

### 4.1.2. Módulo de generación

Este módulo es el encargado de generar los ficheros compatibles con WEKA para el análisis de los distintos clasificadores. A partir del mensaje generado en detección, este módulo se encarga de recibirlo, suscribiéndose al *topic* correspondiente, para poder obtender los puntos en 3D que forman la boca detectada.

Para realizar dichas tareas se ha generado la siguiente función:

- ***callbackDetection()***
  Se reciben 18 puntos con sus coordenadas xyz cada uno, la función recorre el vector de puntos para ir accediendo a las distintas componentes de cada uno de ellos e imprimirlas concatenadas en el fichero destino. De esta forma cada punto se verá reflejado por una línea en el fichero generado, y por orden, cada uno de los atributos hará referencia a los puntos y sus componentes.

### 4.1.3. Módulo de clasificación

Para decidir el clasificador que mejor se ajusta a la problemática expuesta, se han analizado 8 modelos distintos, algunos de ellos con varias configuraciones para ver si afectaban a la calidad de la clasificación y que se incluirán en el Anexo C. Debido a que el objetivo del proyecto es implementar un sistema que fusione información 2D y 3D, se han realizado las comparativas para estos dos entornos y queda reflejadas en la Figura 4.2. En ambos casos se ha utilizado un conjunto de datos que combina las grabaciones de 5 sujetos pronunciando todas las vocales a una distancia media y con las grabaciones de 3 sujetos pronunciando todas las vocales a corta, media y larga distancia (dentro del rango de acción en el que se pretende hacer uso de Mini, es decir, inferior a 2m del robot). En total se dispone de 5971 ejemplos para realizar el entrenamiento. Todas los entrenamientos y las posteriores pruebas se han realizado con una configuración de validación cruzada con 10 iteraciones para cada una de ellas.

En la Figura 4.2 se pueden observar los resultados obtenidos tras analizar los distintos clasificadores con los valores 2D, cuyas componentes son x e y para cada punto. Los resultados obtenidos no superan el 80 % de aciertos en ninguno de los casos, por lo que no parece que este entorno se ajuste correctamente al problema.

En el caso de los datos con valores en 3 dimensiones, cuyas componentes son x, y, z para cada punto, como se muestra en la Figura 4.2, los resultados mejoran a los obtenidos para el caso de 2 dimensiones, y se logra superar la barrera de la tasa de aciertos del 80 % con el clasificador Random Forest. Con esta imagen queda de manifiesto la mejoría que supone el uso de información en 3D con respecto al uso de información en 2D.

Por todo ello, los datos a tener en cuenta tanto para entrenar el clasificador, como para su posterior predicción, se realiza íntegramente con datos obtenidos de la fusión de información 2D y 3D. Y tras el estudio y las tasas de aciertos mostradas, el clasificador a implementar se basa en el algoritmo Random Forest.

Basándose en los resultados obtenidos en el análisis de WEKA, el clasificador a desarrollar se ha configurado con los mismos valores de configuración (100 árboles y valor de la semilla aleatoria 1), para así intentar mantener los valores de tasa de acierto. Pa-

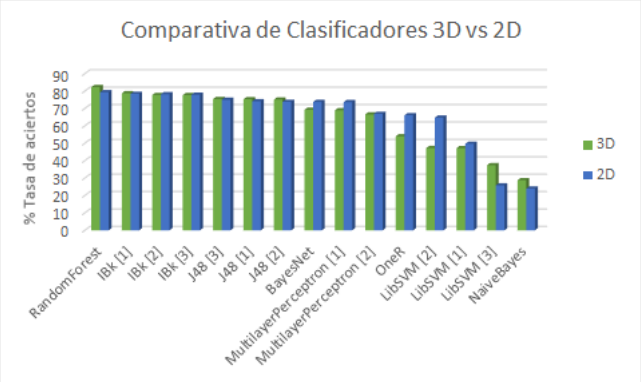


Figura 4.2: Gráfico comparativo de clasificadores

ra ello, se ha integrado en el módulo, una librería llamada Waffles con múltiples herramientas de aprendizaje y clasificación, entre las que se encuentra el algoritmo que se desea utilizar en este proyecto: Random Forest.

El módulo por tanto requiere de las siguientes funciones para su correcto funcionamiento:

- ***Classification()***
  El constructor se encarga de, en caso de que no exista un modelo ya entrenado de Random Forest, entrenar y almacenar el modelo generado con los datos obtenidos en el módulo de generación y guardados en un fichero WEKA. De esta forma se ahorra en tiempo de ejecución, ya que no es necesario estar repitiendo el proceso de entrenar el modelo a usar. En caso de que el modelo se haya salvado correctamente con anterioridad, lo carga y se prepara para su uso.
- ***callbackDetection()***
  Esta función, cada vez que reciba un mensaje generado por el módulo de detección, se introducen en el modelo entrenado y se genera la predicción de la categoría a la que pertenece. Esta predicción se muestra al usuario por consola para su análisis y comprobación.

### 4.1.4. Problemas encontrados durante la implementación

Aunque se han buscado elementos que favorecieran la compatibilidad entre sí, en algunos casos no ha sido del todo sencilla como

se esperaba. Esto ha ocurrido con las dos librerías a integrar en el sistema, que necesitaban de sentencias particulares en el fichero de configuración de la herramienta catkin para que su compilación, generación e inclusión en los módulos fuera correcta.

Durante la búsqueda de las librerías de detección, se probó en primera instancia con la librería de asmlib, pero no fue posible probarla en el sistema por problemas con la integración en la solución. Esto supuso la búsqueda de otras librerías disponibles que igualasen o mejorasen lo visto en asmlib, y se encontró la librería Stasm.

Una vez incorporado el detector de caras, se encontraron dificultades para que los modelos de datos encajaran, y por ello se hizo uso de la librería PCL (sección 2.6.2), que permite la conversión de los modelos de datos involucrados en la detección de forma sencilla.

Con los módulos de detección y generación ya desarrollados, durante la fase de testeo de la generación, se observo que la nube de puntos recibida del sensor Kinect, de forma aleatoria, introducía valores "*nan*" (not a number) en algunos de los puntos. Se analizaron los datos obtenidos, para descartar posibles errores de código, y se confirmó que los valores atípicos provenían de las mediciones del propio sensor, que en algunos casos no es capaz de calcular correctamente las componentes de los puntos y asigna este valor. Para solucionar este imprevisto, se analizaron varias opciones y se implementó la solución que menos desajustes provocara en el diseño del módulo. Dado que las imágenes son continuas capturas de un sujeto pronunciando una vocal, la diferencia de posición de los puntos de la boca entre una y otra son mínimas, por lo que se puede sustituir la captura errónea por la anterior válida. En caso de que no exista una previa, se desecha directamente la detección de esa imagen.

## 4.2. Pruebas unitarias

Como se ha comentado en el capítulo, las pruebas unitarias, es el último hito del desarrollo del proyecto y el que realiza la comprobación de que todos los requisitos han quedado cubiertos. En este apartado se detalla también el entorno sobre el cual se han realizado dichas pruebas, de tal modo que quede constancia de que las pruebas y los requisitos se cumplen bajo unas situaciones prefijadas.

### 4.2.1. Entorno de pruebas del sistema

El entorno sobre le que se han realizado las pruebas es similar en cuanto a configuración al especificado en la implementación, ya que se ha utilizado el mismo equipo. Sin embargo, hay un elemento que varía y es la participación del robot Mini para la consecución de las pruebas. De esta forma, el entorno de pruebas del sistema queda de la siguiente forma:

- Sistema operativo Ubuntu 12.04 LTS.
- ROS Groovy Galapagos instalado y configurado.
- Librería Stasm enlazada al proyecto.
- Librería Waffles enlaza al proyecto.
- Conexión establecida con Mini o uso de *bags* (sección 2.6.1)

### 4.2.2. Descripción textual de las pruebas unitarias

Cada una de las pruebas unitarias se verá reflejada en una tabla independiente, incluyendo los atributos anteriormente citados, de forma que quede clara y concisa la forma de realizarla, el motivo y el resultado deseado.

| **Pruebas unitarias** | | | |
|---|---|---|---|
| **Código** | PRU-001 | **Requisito verificado** | RF-003 |
| **Nombre** | Recepción de imagen 2D | | |
| **Descripción** | El sistema debe suscribirse correctamente al topic correspondiente de Mini para la recepción de imágenes en 2D. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección. | | |
| **Resultado esperado** | • El sistema recibe la imagen 2D. | | |

Tabla 4.1: Prueba del sistema PRU-001

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-002 | **Requisito verificado** | RF-004 |
| **Nombre** | Recepción de nube de puntos | | |
| **Descripción** | El sistema debe suscribirse correctamente al topic correspondiente de Mini para la recepción de la nube de puntos. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección. | | |
| **Resultado esperado** | • El sistema recibe la imagen 3D. | | |

Tabla 4.2: Prueba del sistema PRU-002

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-003 | **Requisito verificado** | RF-001 |
| **Nombre** | Detectar cara | | |
| **Descripción** | El sistema es capaz de detectar la cara del usuario. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección.<br>2. Se situa un usuario delante de Mini. | | |
| **Resultado esperado** | • El sistema detecta la cara y devuelve los puntos que la forman. | | |

Tabla 4.3: Prueba del sistema PRU-003

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-004 | **Requisito verificado** | RF-002 |
| **Nombre** | Detectar boca | | |
| **Descripción** | El sistema es capaz de extraer la boca de la detección de cara realizada. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección.<br>2. Se situa un usuario delante de Mini.<br>3. Se detecta la cara del usuario. | | |
| **Resultado esperado** | • El sistema extrae los puntos que forman la boca (18 en total). | | |

Tabla 4.4: Prueba del sistema PRU-004

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-005 | **Requisito verificado** | RF-005 |
| **Nombre** | Fusionar datos 2D con los datos 3D | | |
| **Descripción** | El sistema unifica la información recibida por el sensor y el módulo de detección para unificar la información 2D y 3D. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección.<br>2. Se situa un usuario delante de Mini.<br>3. Se detecta la cara del usuario. | | |
| **Resultado esperado** | • El sistema auna los datos 2D con los datos 3D . | | |

Tabla 4.5: Prueba del sistema PRU-005

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-006 | **Requisito verificado** | RF-006 |
| **Nombre** | Generación de fichero para análisis en WEKA | | |
| **Descripción** | El sistema es capaz de, en base a los datos generados, crear mediante el módulo de generación, el fichero correspondiente para WEKA. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección.<br>2. Se lanza el módulo de detección.<br>3. Se situa un usuario delante de Mini.<br>4. Se detecta la cara del usuario.<br>5. Se envía el mensaje del módulo de detección. | | |
| **Resultado esperado** | • El sistema recibe el mensaje, lo analiza y lo vuelca a un fichero. | | |

Tabla 4.6: Prueba del sistema PRU-006

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-007 | **Requisito verificado** | RF-007 |
| **Nombre** | Generación y envío del mensaje de detección | | |
| **Descripción** | El sistema es capaz de, en base a los datos obtenidos, generar un mensaje con la información necesaria para el resto de módulos. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección.<br>2. Se situa un usuario delante de Mini.<br>3. Se detecta la cara del usuario. | | |
| **Resultado esperado** | • El sistema genera y envía el mensaje de detección con los datos obtenidos durante la fusión de la imagen 2D y 3D. | | |

Tabla 4.7: Prueba del sistema PRU-007

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-008 | **Requisito verificado** | RF-008 |
| **Nombre** | Generación y envío del mensaje de clasificación | | |
| **Descripción** | El sistema muestra el resultado de la clasificación. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección.<br>2. Se lanza el módulo de clasificación.<br>3. Se situa un usuario delante de Mini.<br>4. Se detecta la cara del usuario.<br>5. Se unifica la imagen 2D con 3D.<br>6. Se envía el mensaje de detección. | | |
| **Resultado esperado** | • El sistema recibe el mensaje de detección y lo asocia mediante el clasificador a la categoría (vocal) más representativa. | | |

Tabla 4.8: Prueba del sistema PRU-008

| Pruebas unitarias | | | |
|---|---|---|---|
| **Código** | PRU-009 | **Requisito verificado** | RF-009 |
| **Nombre** | Resolución de valores atípicos en la nube de puntos (3D) | | |
| **Descripción** | El sistema corrige los valores erróneos de la nube de puntos recibida. | | |
| **Procedimiento** | 1. Se lanza el módulo de detección.<br>2. Se situa un usuario delante de Mini.<br>3. Se detecta la cara del usuario.<br>4. Se unifica la imagen 2D con 3D. | | |
| **Resultado esperado** | • El sistema corrige la información recibida en base los datos de que disponga. Si existe detección previa la corrige basándose en esta. Si no, desecha los datos de la captura. | | |

Tabla 4.9: Prueba del sistema PRU-009

### 4.2.3. Matriz de trazabilidad

La matriz de trazabilidad de la Tabla 4.10 muestra la relación entre las pruebas unitarias realizadas y el requisito funcional que cumplen. De este modo se puede ver de forma esquemática si las pruebas realizadas son suficientes para verificar los requisitos marcados.

**Resumen Requisitos funcionales**
RF-001: Reconocimiento facial
RF-002: Reconocimiento de la boca
RF-003: Recepción de los datos de la imagen 2D
RF-004: Recepción de los datos de la imagen 3D
RF-005: Fusión de datos en 2D y datos en 3D
RF-006: Generación de ficheros para el estudio de clasificadores
RF-007: Generación y envío de mensaje de detección
RF-008: Generación y envío de mensaje de clasificación
RF-009: Resolución de valores atípicos en la nube de puntos (3D)

| | PRU-001 | PRU-002 | PRU-003 | PRU-004 | PRU-005 | PRU-006 | PRU-007 | PRU-008 | PRU-009 |
|---|---|---|---|---|---|---|---|---|---|
| RF-001 | | | X | | | | | | |
| RF-002 | | | | X | | | | | |
| RF-003 | X | | | | | | | | |
| RF-004 | | X | | | | | | | |
| RF-005 | | | | | X | | | | |
| RF-006 | | | | | | X | | | |
| RF-007 | | | | | | | X | | |
| RF-008 | | | | | | | | X | |
| RF-009 | | | | | | | | | X |

Tabla 4.10: Matriz de trazabilidad entre Pruebas Unitarias y Requisitos Funcionales

# Capítulo 5

# Evaluación del sistema

Para la evaluación se han realizado pruebas con el fin de cuantificar la calidad de la solución implementada y para ver los límites de sus capacacidades. De esta forma se analizan los resultados no sólo en las situaciones óptimas, sino que se pueden observar qué problemas o deficiencias tiene el sistema.

Estas pruebas se han realizado con 6 sujetos distintos, 4 hombres y 2 mujeres, para comprobar que el detector y el clasificador funcionan en la mayoría de las situaciones. También se ha probado colocando a los usuarios a diferentes distancias del robot de escritorio para verificar el rango óptimo de medición.

En las figuras que se muestran a continuación, aparece el sujeto pronunciando una vocal a una determinada distancia, el sistema de detección busca y posiciona los puntos que forman la boca (puntos blancos en la imagen), para que posteriormente el sistema de clasificación busque la vocal más representativa para esos datos.

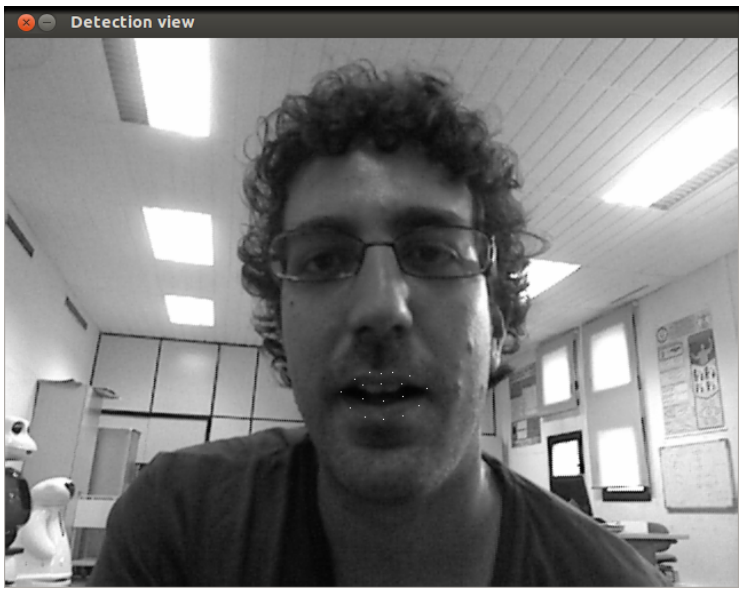


Figura 5.1: Detección correcta de cerca

En la Figura 5.1 se observa un ejemplo de lo que se consideraría distancia próxima. El sistema en esta situación es capaz de identificar correctamente la boca del usuario y por tanto procesa y clasifica los puntos que la forman para reconocer la vocal pronunciada.


Figura 5.2: Detección correcta a media distancia

La distancia habitual de uso del robot se muestra en las Figuras 5.2 y 5.3. Como se puede apreciar en este último caso, el detector en ocasiones no es capaz de posicionar correctamente la boca. Esto se debe a las limitaciones de la librería Stasm, que aunque es capaz de identificar la boca del usuario a través de 18 puntos, disminuye su fiabilidad en giros o inclinaciones de cabeza.

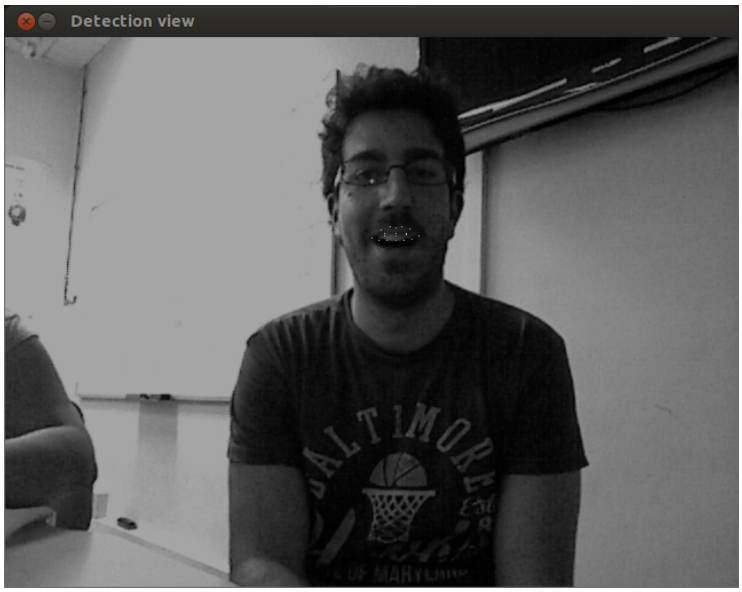


Figura 5.3: Detección incorrecta a media distancia

Por último, se analiza el caso de distancias mayores de las previstas para el uso del robot, únicamente para comprobar y conocer los límites del sistema. En la Figura 5.4 se aprecia que el detector es capaz de situar la boca del usuario a esas distancias, pero la estabilidad del posicionamiento y su calidad no son lo suficientemente constantes como para aceptar el análisis de esta situación.

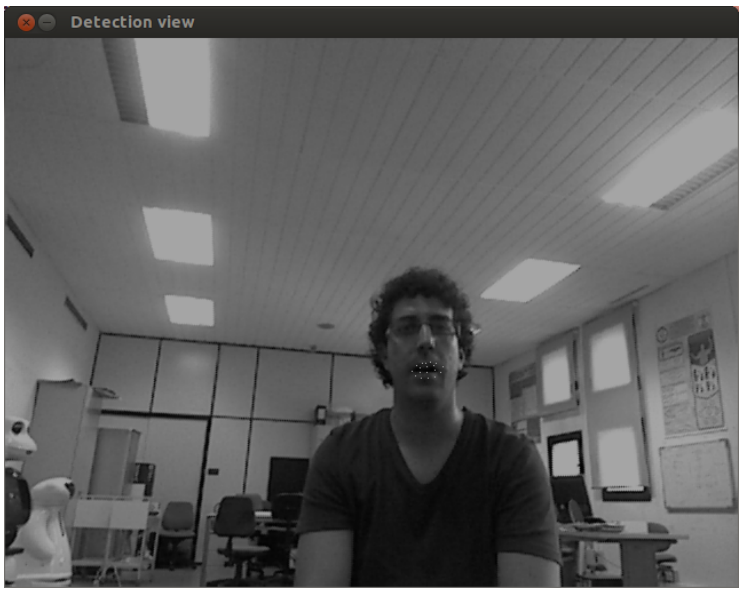


Figura 5.4: Detección incorrecta a larga distancia

Una vez realizadas las pruebas a diferentes distancias del robot, se procede a mostrar los resultados obtenidos durante la experimentación. Cabe mencionar que como se ha indicado anteriormente, debido a la inestabilidad de los resultados a larga distancia, no se han incluido en el estudio. Para distancias próximas y medias se han unificado las pruebas en base la vocal pronunciada y al sexo del sujeto que las ha realizado, esto último para observar si el sistema se ve afectado por un sobreentrenamiento con hombres.

| **Vocal pronunciada** | **Clasificación obtenida** | | | | |
|---|---|---|---|---|---|
| **a** | **a** | **e** | **i** | **o** | **u** |
| **Hombres** | 156 | 11 | 132 | 54 | 18 |
| **Mujeres** | 0 | 0 | 96 | 0 | 0 |

Tabla 5.1: Clasificación de la pronunciación a

Para el caso de la vocal pronunciada 'a', como se puede observar en la Tabla 5.1, se obtienen una mayor tasa de clasificaciones correctas en el caso de los hombres que en el caso de las mujeres, que se clasifican todas las detecciones como 'i'. Sin embargo, estos aciertos suponen un 42 % del total detectado (156 clasificaciones de 371 totales). Este resultado no es excesivamente bueno, pero logra distanciarse en un 6 % de la vocal más problemática para este esce-

nario, la 'i' (36 %, 132 clasificaciones de 371 totales).

| Vocal pronunciada | Clasificación obtenida | | | | |
|---|---|---|---|---|---|
| **e** | **a** | **e** | **i** | **o** | **u** |
| **Hombres** | 72 | 0 | 78 | 37 | 37 |
| **Mujeres** | 24 | 0 | 30 | 2 | 1 |

Tabla 5.2: Clasificación de la pronunciación e

La Tabla 5.2 refleja las clasificaciones cuando la vocal pronunciada es la 'e'. Tanto en el caso de los hombres como en el de las mujeres, la letra 'e' no ha sido correctamente clasificada y se ha confundido con el resto de las vocales disponibles. Se puede destacar la relevancia de las vocales 'a' e 'i', ya que son las vocales que mayor número de falsas clasificaciones han obtenido.

| Vocal pronunciada | Clasificación obtenida | | | | |
|---|---|---|---|---|---|
| **i** | **a** | **e** | **i** | **o** | **u** |
| **Hombres** | 0 | 0 | 168 | 0 | 0 |
| **Mujeres** | 32 | 0 | 71 | 0 | 24 |

Tabla 5.3: Clasificación de la pronunciación i

Sin embargo, el caso de la vocal 'i' queda claramente identificada para ambos sexos. Con una sorprendente tasa del 100 % de tasa de aciertos para los hombres y un 56 % para las mujeres (71 clasificaciones de 127 totales). Con respecto a esta vocal los resultados sí han sido satisfactorios, ya que en el caso de los hombres es un éxito rotundo, y en el de las mujeres logra distanciarse en un 31 % de la siguiente vocal con más clasificaciones ('a' 32 clasificaciones de 127 totales, 25 %).

| Vocal pronunciada | Clasificación obtenida | | | | |
|---|---|---|---|---|---|
| **o** | **a** | **e** | **i** | **o** | **u** |
| **Hombres** | 24 | 0 | 26 | 20 | 18 |
| **Mujeres** | 0 | 0 | 40 | 12 | 24 |

Tabla 5.4: Clasificación de la pronunciación o

El análisis de las clasificaciones de la vocal 'o' obtiene una tasa

de aciertos del 23 % para hombres (20 clasificaciones de 88 totales) y 16 % para mujeres (12 clasificaciones de 76 totales), la 'i' sin embargo alcanza un 30 % y un 53 % respectivamente. Para el resto de vocales, como se puede ver en la Tabla 5.4, destaca que en el caso de los hombres se confunde la 'a' en más ocasiones que aciertos con la 'o', y en el caso de la 'u' logra valores muy próximos.

| **Vocal pronunciada** | **Clasificación obtenida** | | | | |
|---|---|---|---|---|---|
| **u** | **a** | **e** | **i** | **o** | **u** |
| **Hombres** | 0 | 0 | 40 | 12 | 24 |
| **Mujeres** | 0 | 0 | 49 | 0 | 0 |

Tabla 5.5: Clasificación de la pronunciación u

Finalmente, para la prueba de pronunciación con la 'u', como se muestra en la Tabla 5.5, la 'i' vuelve a hacer acto de presencia obteniendo en los hombres unos altos porcentajes de clasificaciones (53 %), y una tasa de acierto del 100 % en el caso de las mujeres. La letra de la prueba obtiene en los hombres un 32 % de aciertos (24 clasificaciones de 76 totales).

Como conclusiones de la experimentación realizada, se debe mencionar que el sistema de detección funciona mejor con sujetos varones, esto puede deberse a que el entrenamiento se ha realizado con una mayoría de hombres y por tanto el sistema no maneja bien los casos de mujeres, es decir, no generaliza correctamente. Con respecto al sistema de clasificación, ha quedado patente la dificultad que supone diferenciar en un entorno real la vocal pronunciada por un usuario en condiciones normales, es decir, sin exagerar la pronunciación o gesticulación de la boca. Ha quedado reflejado en las tablas previas que las vocales que se identifican con mayor facilidad son la 'i' y la 'a' en ambos sexos, por orden de tasa de aciertos. Por contra, la letra 'e' en las pruebas realizadas ha sido la peor parada, ya que únicamente en las primeras pruebas ha habido clasificaciones erróneas de esta vocal, en el resto de los casos ni aparece.

# Capítulo 6

# Gestión del proyecto

La gestión del proyecto abarca todos los elementos que se han de tener en cuenta para la elaboración del mismo y que no tienen que ver de forma directa con el desarrollo per se (análisis, diseño e implementación). En este apartado se justifica la metodología de desarrollo a seguir durante el proyecto, de tal forma que sirva de guía para las distintas tareas a realizar. También se realiza una planificación de las tareas para organizar el trabajo, asignarle fechas de ejecución y disponer de una visión completa de dicho proyecto. Por último, se incluye el presupuesto correspondiente por los gastos asociados al desarrollo del proyecto, desglosando los costes en función de personal, infraestructura y material involucrados.

## 6.1. Metodología de desarrollo

Existen múltiples metodologías de desarrollo de software disponibles y cada una de ellas se ajusta mejor dependiendo del tipo de proyecto y del personal involucrado.

En el caso del proyecto que nos atañe, debido a que el desarrollo realizado es lineal, ya que cada uno de los módulos depende de la finalización del anterior, se ha decidido que la metodología más adecuada es el ciclo de vida en cascada. Existen dos variaciones dentro de esta metodología, la clásica y la realimentada, ésta última permite realizar ajustes de una etapa previa a la que se esté desarrollando, cosa que la clásica no. Sin embargo, este modelo no es recomendable en equipos de varias personas, puesto que dificulta enormemente el

avance simultáneo del proyecto, pero para el caso de esta aplicación no supone ningún problema por haber sido desarrollada por una única persona. En la Figura 6.1 se pueden contemplar las distintas etapas que componen el modelo en cascada realimentado.

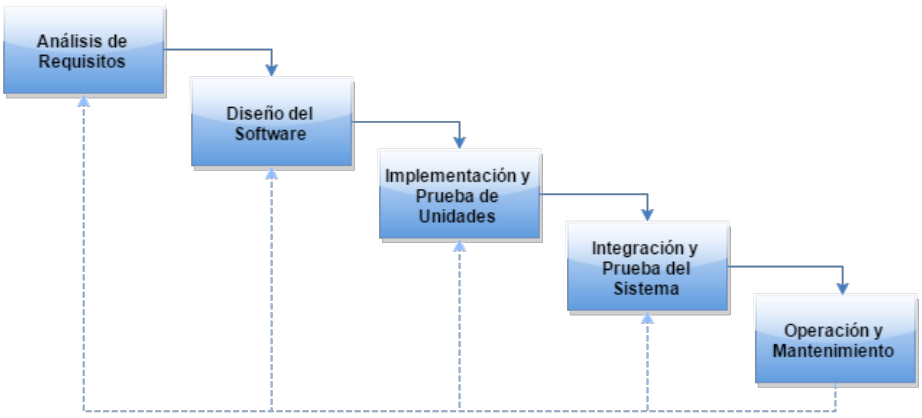


Figura 6.1: Ciclo de vida en cascada realimentado

## 6.2. Planificación del proyecto

La planificación del proyecto se ha basado en la metodología escogida en el apartado anterior, ciclo de vida en cascada realimentada. Por tanto, las distintas tareas que se podrán visualizar en dicha planificación representarán todos y cada uno de los ciclos que componen este modelo, además de un nuevo apartado en el que se verá reflejada la documentación del proyecto. Para construir esta planificación se ha utilizado la herramienta GanttProject, un programa orientado a la gestión de proyectos de software, capaz de reflejar la planificación de forma simple y bien estructurada. Como se puede apreciar en la Figura 6.2, las distintas tareas dependen de la finalización de su tarea anterior ya que se han ido realizando de forma secuencial para ser consecuente con la metodología aplicada, a excepción del apartado de documentación, cuyo desarrollo ha tenido una evolución constante desde el inicio del proyecto.

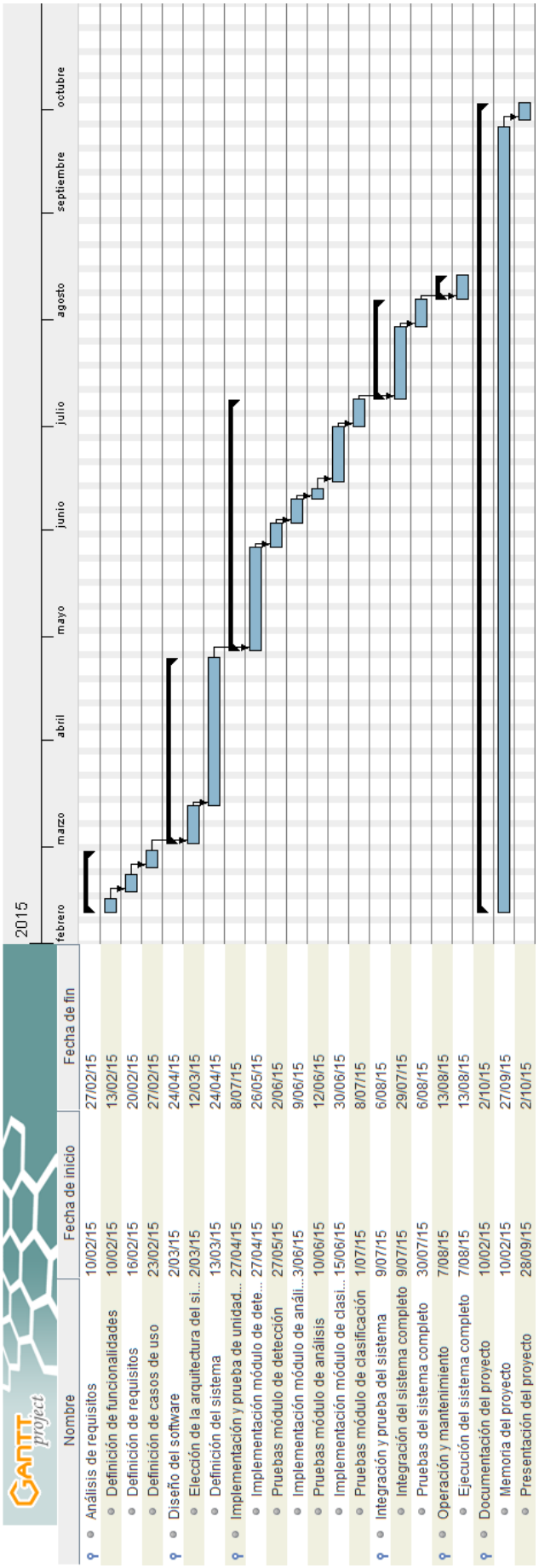


Figura 6.2: Diagrama de Gantt para la planificación del proyecto

## 6.3. Presupuesto del proyecto

Todos los gastos asociados al desarrollo del proyecto, ya sean costes de personal, infraestructuras o material, están reflejados en este apartado. El presupuesto de un proyecto debe contemplar también los costes indirectos asociados y el beneficio industrial que se pretende aplicar al mismo. A su vez, el total presupuestado debe encajar en el marco de una oferta atractiva para que el cliente se decante por ella, y bien consolidada para que el desarrollo no incurra en sobrecostos.

### 6.3.1. Costes de personal

El coste de personal asociado al proyecto se ha calculado en base a las tablas salariales dispuestas en el Convenio colectivo de ámbito estatal para los centros de educación universitaria e investigación, del BOE número 36 del 30 de enero del 2015 [22] , para el presente año. Para realizar el cálculo de los costes de personal es necesario especificar el cómputo aplicado correspondiente a la Seguridad Social [23] , que se ha realizado teniendo en cuenta los siguientes apartados y porcentajes:

- Contingencias comunes: 23,60 %
- Desempleo por contrato de duración determinada Tiempo Completo: 6,70 %
- Fondo de Garantía Salarial (FOGASA): 0,20 %
- Formación Profesional: 0,60 %
- Tarifa para la cotización por accidentes de trabajo y enfermedades profesionales para la categoría de Programación, consultoría y otras actividades relacionadas con la informática [24] : 1,35 %

| Costes de personal | | | | |
|---|---|---|---|---|
| **Cargo** | **Sueldo base** | **Número de meses** | **Coste total** | **Coste total con Seguridad Social** |
| **Jefe de proyecto** | 1.910,35€ | 1,00 | 1.910,35€ | 2.530,26€ |
| **Analista** | 1.548,37€ | 1,40 | 2.167,72€ | 2.871,14€ |
| **Programador** | 1.202,80€ | 2,33 | 2.806,53€ | 3.717,25€ |
| **Técnico de pruebas** | 1.202,80€ | 0,90 | 1.082,52€ | 1.433,80€ |
| | | | **TOTAL** | 10.552,45€ |

Tabla 6.1: Costes de personal asociados al proyecto

### 6.3.2. Costes de infraestructura

Puesto que en el desarrollo del proyecto todas los elementos utilizados son de código abierto, este apartado no supone ningún incremento en los costes finales del presupuesto.

Al no haber incrementos por esta parte hace que el proyecto sea mucho más interesante desde el punto de vista económico.

### 6.3.3. Costes de material

El desglose de los costes de material asociados se realiza mediante el cálculo de los recursos utilizados, los meses aplicados al proyecto y su dedicación. Con esto, en base a la amortización que se quiera obtener de cada elemento, se obtienen los cálculos de costes de material que inciden en el presupuesto total, quedando la fórmula de la siguiente forma:

$$\text{Coste de material} = \frac{\text{Precio}}{\text{Periodo de amortización}} \cdot \text{Número de meses de dedicación}$$

| Costes de material | | | | |
|---|---|---|---|---|
| **Material** | **Precio** | **Número de meses de dedicación** | **Periodo de amortización (meses)** | **Coste total** |
| **PC Intel Core i5-2500K** | 900,00€ | 6 | 48 | 112,50€ |
| **Portátil VAIO i5-2510M** | 600,00€ | 2 | 48 | 25,00€ |
| | | | **TOTAL** | 137,50€ |

Tabla 6.2: Costes de material asociados al proyecto

Se han incluido en este apartado dos equipos para el desarrollo: un PC de sobremesa y un portátil. Esto se debe a que para la mayor parte del desarrollo se ha utilizado el PC de sobremesa, pero para la realización de las pruebas se hizo uso del portátil.

### 6.3.4. Presupuesto total del proyecto

Finalmente se muestra el presupuesto total del proyecto, que integra los costes indicados en los apartados previos, añadiendo los costes indirectos que afectan al proyecto y el beneficio industrial correspondiente al desarrollo.

| Presupuesto total del proyecto | |
|---|---|
| **Costes de personal** | 10.552,45€ |
| **Costes de material** | 137,50€ |
| **Costes indirectos (10 %)** | 1.069,00€ |
| **Beneficio industrial (6 %)** | 641,40€ |
| **TOTAL** | **12.400,34€** |

Tabla 6.3: Presupuesto total del proyecto

El presupuesto total de este proyecto asciende a la cantidad de 12.400,34€ (doce mil cuatrocientos euros y treinta y cuatro céntimos), IVA no incluido.

# Capítulo 7

# Conclusiones y trabajos futuros

Este apartado se analizan y describen las conclusiones alcanzadas una vez finalizado el proyecto, se pretende plasmar los resultados obtenidos y su significado. También se incluyen desarrollos futuros derivados del trabajo aquí realizado.

## 7.1. Conclusiones

Como se ha podido observar a lo largo de la memoria, el principal objetivo del trabajo se ha cumplido: diseñar e implementar un sistema de reconocimiento vocal a través de información RGB-D. Sin embargo, cabe destacar que los resultados obtenidos durante la evaluación del sistema no han cumplido con las expectativas existentes tras el análisis con la herramienta WEKA, cuyos resultados de aciertos en la clasificación fueron superiores a un 80 %.

Es posible que estos resultados se deban a que el conjunto de perfiles (personas) que se utilizaron para el entrenamiento del clasificador no fuera lo suficientemente amplio como para generar un modelo genérico competente. Así como también haya influido el uso de más sujetos varones, dando lugar a un sobreentrenamiento del clasificador para estos casos, y por tanto en los procesos con mujeres se obtengan unos resultados peores.

En el ámbito de los reconocedores faciales, se puede afirmar que

el sistema es mejorable y por tanto capaz de otorgar unos valores más precisos para el posicionamiento correcto de la boca, mediante la manipulación, filtrado y análisis más exhaustivo de las imágenes recibidas, de tal forma que se podrían enriquecer los resultados de la detección y derivar en un análisis y clasificación más precisos. Aun así, el reconocedor utilizado ha cumplido con su cometido, pero habría sido interesante disponer del reconocedor IntraFace (sección 2.2.3) que tan buenas expectativas había generado en las demostraciones, para ver su desempeño en la solución desarrollada. Pero por retrasos en su publicación hicieron imposible su incorporación.

La detección de la boca se realiza en base a la información recibida en 2D, para posteriormente emparejarla con los valores correspondientes obtenidos de la nube de puntos 3D, y aunque no ha mostrado ser uno de los motivos que afectasen al rendimiento del sistema, no se descarta su implicación por la existencia de los valores *"nan"* descritos en los problemas de implementación (sección 4.1.4). Modificar el mecanismo de corrección también sería válido, para descartar fallos en ese sentido. Aun así, sería interesante visualizar los resultados mediante un detector de boca que opere únicamente con información en 3D para aprovechar todo el potencial de este tipo de sensores.

Todo esto indica que el sistema es viable y puede suponer una vía de estudio factible y mejorable para el desarrollo de las herramientas que requiere el proyecto RobAlz, para así alcanzar los objetivos deseados y resultar en una herramienta provechosa para las terapias con personas con Alheimer o demencia.

## 7.2. Trabajos futuros

La solución implementada puede ser mejorada o utilizada para el desarrollo de futuras líneas de trabajo. Parte de la motivación detallada en la introducción esta encaminada en este sentido, ya que se busca que el proyecto realizado sirva como base para implementaciones más complejas de sistemas que realicen terapias de logopedia para pacientes de Alzheimer o demencias.

El análisis de los movimientos de la boca a la hora de pronunciar una vocal, para que el sistema sea capaz de evaluar y corregir los gestos realizados por una persona al hablar, podría ser una de las evoluciones. Siendo ambiciosos, incluso el análisis de palabras

completas en lugar de vocales. También herramientas que permitan realizar y evaluar ejercicios de movilidad de los músculos de la boca, que ayudarían a los terapeutas con las terapias de vocalización que requieren los enfermos.

Para mejorar los resultados en el apartado de la detección, se podría incluir otro sistema más refinado como es IntraFace (véase sección 2.2.3), ya que su integración debería ser sencilla, gracias a la arquitectura modular utilizada. Puesto que, en base a las demostraciones de la herramienta, mejoran de forma significativa las detecciones faciales.

En el ámbito de la clasificación, sería recomendable realizar un estudio más amplio con igualdad de casos de ambos sexos, para así obtener un modelo mejor entrenado, y por tanto, capaz de obtener mejores resultados de clasificación. Al generar este nuevo conjunto de datos, se debería repetir el estudio de análisis de clasificadores para comprobar si este cambio supone en una variación del modelo de clasificador que se ajuste mejor al problema.

Por último, se podría hacer uso del sistema para combinarlo con un reconocimiento audiovisual de las palabras, obteniendo un sistema que puede completar y mejorar las soluciones únicamente basadas en el sonido. De esta forma se podrían desarrollar otros elementos que posibiliten tareas involucradas en el reconocimiento de la voz.

# Glosario

- RGB-D: imagen con profundidad.
- 2D: dos dimensiones.
- 3D: tres dimensiones.
- Hardware: elementos físicos que componen un sistema u ordenador.
- RGB: (Red Green Blue), hace referencia a la combinación de colores para formar una imagen.
- Framework: conjunto de herramientas para el desarrollo de software.
- C++: lenguaje de programación basado en C pero orientado a objetos.
- Xbox360: videoconsola creada por Microsoft.
- Controlador: programa informático que permite al sistema operativo interactuar con los elementos hardware conectados.
- Topic: mecanismo de comunicación entre nodos ROS a través del cual se envían mensajes.
- Java: lenguaje de programación orientado a objetos.
- Lisp: lenguaje de programación multiparadigma, siendo de los lenguajes más antiguos que se siguen usando hoy en día.
- Lua: lenguaje de programación imperativo, estructurado y ligero.
- Python: lenguaje de programación interpretado multiparadigma sencillo.
- C4.5: algoritmo de clasificación basado en árboles de decisión.

- LTS: (Long Term Support), soporte a largo plazo.
- Gant: herramienta gráfica para visualizar el tiempo de dedicación de las tareas o actividades de un proyecto.
- BOE: Boletín Oficial del Estado.

# Bibliografía


[1] "Stasm 4 user manual." http://www.milbo.org/stasm-files/stasm4.pdf. Accessed: 2015-02-10.

[2] F.-S. D. and M. M.J., "Defining socially assistive robotics," pp. 465–468, 2005.

[3] J. T. J. S. Ben P. Yuhas, Moise H. Goldstein, "Integration of acoustic and visual speech signals using neural networks,"

[4] A. C. Piotr Dalka, Piotr Bratoszewski, "Visual lip contour detection for the purpose of speech recognition,"

[5] I. A. Eldirawy, "Visual speech recognition,"

[6] A. K. J. Stan Z. Li, *Handbook of Face Recognition*.

[7] "Semantic vision technologies." http://www.semanticvisiontech.com/index.html. Accessed: 2015-02-10.

[8] J. Naruniec, "Discrete area filters in accurate detection of faces and facial features," pp. 979–993.

[9] "Active shape model library (asmlib)." https://github.com/cxcxcxcx/asmlib-opencv. Accessed: 2015-02-10.

[10] "Intraface." http://www.humansensing.cs.cmu.edu/intraface/. Accessed: 2015-04-14.

[11] "Stasm." http://www.milbo.users.sonic.net/stasm/. Accessed: 2015-02-10.

[12] S. Milborrow and F. Nicolls, "Active shape models with sift descriptors and mars," *VISAPP*, 2014.

[13] "Randomtrees." http://docs.opencv.org/modules/ml/doc/random_trees.html. Accessed: 2015-06-15.

[14] "random-forest." https://github.com/bjoern-andres/random-forest. Accessed: 2015-06-15.

[15] http://waffles.sourceforge.net/. Accessed: 2015-06-20.

[16] "Json - javascript object notation." http://www.json.org/json-es.html. Accessed: 2015-08-25.

[17] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications,"

[18] "Robotic operating system (ros)." http://wiki.ros.org/. Accessed: 2015-02-10.

[19] "Point cloud library." http://pointclouds.org/documentation/. Accessed: 2015-02-20.

[20] "Opencv." http://opencv.org/. Accessed: 2015-02-10.

[21] "Weka." http://www.cs.waikato.ac.nz/ml/weka/. Accessed: 2015-06-05.

[22] "Convenio colectivo de ámbito estatal para los centros de educación universitaria e investigación." https://www.boe.es/boe/dias/2015/02/11/pdfs/BOE-A-2015-1343.pdf. Accessed: 2015-08-25.

[23] "Bases y tipos de cotización 2015." http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm. Accessed: 2015-08-25.

[24] "Tarifa para la cotización por accidentes de trabajo y enfermedades profesionales." http://www.seg-social.es/prdi00/groups/public/documents/binario/113903.pdf. Accessed: 2015-08-25.

# Anexo A: Descripción de atributos de las tablas

## Descripción de los atributos de los casos de uso

La descripción textual de los casos de uso se va a realizar a través de tablas que reflejarán de forma más exhaustiva los casos de uso descritos en el apartado anterior. Por tanto, estas tablas se verán compuestas por los siguientes elementos que se pasan a describir:

- Código: elemento alfanumérico que identifica de forma unívoca los casos de uso desarrollados. El formato a usar será con las letras CU, un guión y tres dígitos para su idetificación, quedando algo así CU-000.
- Nombre: texto que refleja de forma más clara el caso de uso.
- Actor: conjunto de entidades que actúan o intervienen en el caso de uso descrito.
- Descripción: análisis breve de las funcionalidades que afectan al caso de uso.
- Precondiciones y efectos: se describen las condiciones que deben cumplirse para la realización de la operación, y el estado al que transita el sistema después de realizarla.
- Curso de eventos: descripción básica de las acciones que se ejecutarán paso a paso en el caso de uso.

## Descripción de los atributos de los requisitos

Al igual que con los casos de uso, en este apartado se describen los distintos elementos que forman parte de las tablas de requisitos, mejorando la comprensión de cada uno de los campos que serán incluidos en ellas. Los elementos que conforman la tabla de requisitos son:

- Código: elemento alfanumérico que identifica de forma unívoca los requisitos. El formato a usar será con las letras RF o RFN dependiendo de si es o no funcional, un guión y tres dígitos para su idetificación.
- Fuente: indicador que asocia el requisito con el caso de uso del que proviene o la fuente del cual se ha extraído.
- Nombre: texto que refleja de forma escueta el requisito.
- Descripción: explicación extensa del requisito, en la cual se indica el alcance del mismo y en qué afecta.
- Necesidad: determina el grado de implementación del requisito. Los valores aceptados son:
  - Esencial: el requisito debe ser implementado ya que afecta a la resolución del proyecto.
  - Deseable: el requisito no afecta a la resolución del proyecto, por tanto es opcional, pero mejoraría la calidad del mismo.
  - Opcional: este requisito no afecta al objetivo ni a la calidad final del proyecto.
- Prioridad: define la importancia del requisito, permitiendo ordenar su ejecución tanto para el diseño como para la implementación. Los valores que pueden aparecer en este atributo son:
  - Alta: el requisito debe ser implementado en las fases previas del proyecto.
  - Media: el requisito debe ser implementado una vez se hayan completado todos los requisitos de prioridad alta.
  - Baja: estos requisitos deben resolverse en la parte final del desarrollo ya que no influyen en el correcto funcionamiento de la solución.

- Estabilidad: indica las posibilidades de que el requisito sufra modificaciones durante el ciclo de vida del proyecto. Los valores utilizados para reflejar este campo son:
  - Estable: el requisito no puede variar durante el ciclo de vida del proyecto.
  - Inestable: el requisito puede variar a lo largo del ciclo de vida del proyecto.
- Verificabilidad: refleja la capacidad de comprobar que dicho requisito se satisface en el sistema. Esta capacidad se indica con los siguientes valores:
  - Alta: se puede verificar que el requisito ha sido introducido ya que se corresponde con las funcionalidades básicas del sistema.
  - Media: se puede verificar que el requisito forma parte del sistema mediante pruebas complejas o la visualización del código fuente.
  - Baja: es difícil o imposible verificar que el requisito forma parte del sistema.

## Descripción de los atributos de las pruebas del sistema

Para facilitar la comprensión de las pruebas realizadas sobre el sistema y su representación en este documento, se incluye la descripción de los distintos atributos que las componen, de forma que todos los elementos queden aclarados y especificando los valores que pueden albergar:

- Código: elemento alfanumérico que identifica de forma unívoca las pruebas realizadas. El formato a usar será con las letras PRU, un guión y tres dígitos para su idetificación, quedando algo así PRU-000.
- Nombre: texto que refleja de forma más clara la prueba.
- Requisito verificado: indicador que asocia el requisito comprobado.
- Descripción: descripción básica de la prueba realizada.

- Procedimiento: secuencia de acciones que deben realizarse en la prueba.
- Resultado esperado: detalla las condiciones que deben darse para aceptar la prueba como válida.

# Anexo B: Configuración del entorno

## Instalación de Groovy Galapagos

Los pasos a seguir para la instalación y configuración del entorno ROS en su versión Groovy Galapagos se encuentran disponibles y de forma clara en su web de ayuda para instalaciones en Ubuntu [1].

## Instalación de Stasm

En el caso de Stasm, los pasos a seguir son los siguientes para que el sistema funcione correctamente en un módulo de ROS.

En primer lugar, se debe descargar el paquete que contiene la herramienta:

```
$ wget http://www.milbo.org/stasm-files/4/stasm4.1.0.tar.gz
```

Se descomprime y se obtiene el directorio stasm4.1.0:

```
$ tar -xvzf stasm4.1.0.tar.gz
```

Se accede al directorio y se parchean los ficheros aquí indicados:

```
$ cd stasm4.1.0/apps
$ patch -p0 < appmisc.cpp.20140201.diff
$ cd stasm4.1.0/apps/shapefile
$ patch -p0 < shapefile.cpp.20140201.diff
```

[1] http://wiki.ros.org/groovy/Installation/Ubuntu

Se ejecuta el comando cmake para realizar la construcción de la librería en la carpeta que se desee (por ejemplo, stasm4.1.0/build). Se accede a la carpeta build y se ejecuta el comando make.

Con esto ya tenemos disponible para su uso la librería Stasm.

## Instalación de Waffles

La instalación de Waffles es muy sencilla, sólo requiere de la descarga del código, su compilación y ya se puede hacer uso de la librería.

La descarga se realiza de aquí:

```
http://sourceforge.net/projects/waffles/files/waffles
```

Al igual que en el caso de Stasm, se debe descomprimir el fichero:

```
$ tar -xvzf waffles_version.tar.gz
```

Se accede a la carpeta waffles/src y se ejecuta:

```
$ make install
```

Ya se puede hacer uso de la herramienta Waffles.

## Configuración del CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(vowel_identification_skill)

set(CMAKE_PREFIX_PATH ${CMAKE_PREFIX_PATH} "/path/to/stasm4.1.0/bui
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -I /path/to/waffles/src/")

find_package(catkin REQUIRED COMPONENTS
cv_bridge
rospy
roscpp
sensor_msgs
std_msgs
image_transport
```

```
pcl
pcl_ros
message_generation
geometry_msgs
)

find_package( OpenCV REQUIRED )
find_package( STASM REQUIRED )

...

include_directories(
${catkin_INCLUDE_DIRS}
${OpenCV_INCLUDE_DIRS}
${STASM_INCLUDE_DIRS}
)

## Declare a cpp library
# add_library(mouth_detection
#   src/${PROJECT_NAME}/mouth_detection.cpp
# )

LINK_DIRECTORIES( ${STASM_LINK_DIRS} )

target_link_libraries(detection
${OpenCV_LIBRARIES}
${catkin_LIBRARIES}
${STASM_LIBRARIES}
)
target_link_libraries(generation
${OpenCV_LIBRARIES}
${catkin_LIBRARIES}
)
target_link_libraries(classification
${catkin_LIBRARIES}
/path/to/waffles/lib/libGClassesDbg.a
)
```

# Anexo C: Pruebas realizadas con clasificadores

En este anexo se pretende dejar constancia de las pruebas realizadas en WEKA para la búsqueda del clasificador que mejor se adecuase al problema planteado. Se indicarán las distintas configuraciones de cada modelo para un mismo conjunto de datos de entrenamiento. También se mostrarán los tiempos de construcción y la matriz de confusión de los modelos.

El número de instancias analizadas para todos los casos es de 5971, con 54 atributos cada una (18 puntos de la boca, con sus correpsondientes coordenadas xyz) y un atributo más para definir la clase (a, e, i, o, u). Para el testeo de los clasificadores se ha utilizado el mecanismo de validación cruzada de 10 *folds*.

## J48

### J48 - configuración 1

```
Scheme:       weka.classifiers.trees.J48 -C 0.1 -M 2

Time taken to build model: 0.64 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4497
75.314  %
Incorrectly Classified Instances      1474
24.686  %
```

```
=== Confusion Matrix ===

a    b    c    d    e   <-- classified as
786  125   71   47   60 |    a = a
98   957  110   79   40 |    b = e
65   100 1113  79   31 |    c = i
61   92   84   888  97 |    d = o
53   53   33   96   753 |    e = u
```

**J48 - configuración 2**

```
Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2

Time taken to build model: 0.63 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4487
75.1465 %
Incorrectly Classified Instances      1484
24.8535 %

=== Confusion Matrix ===

a    b    c    d    e   <-- classified as
790  124   66   48   61 |    a = a
111  943  103   84   43 |    b = e
64   101 1110  81   32 |    c = i
65   82   90   889  96 |    d = o
60   48   34   91   755 |    e = u
```

**J48 - configuración 3**

```
Scheme:       weka.classifiers.trees.J48 -C 0.05 -M 2

Time taken to build model: 0.61 seconds

=== Stratified cross-validation ===
=== Summary ===
```

```
Correctly Classified Instances         4504
75.4313 %
Incorrectly Classified Instances       1467
24.5687 %

=== Confusion Matrix ===

a    b    c    d    e   <-- classified as
788  123   68   53   57 |    a = a
97   972   99   78   38 |    b = e
64   103 1110   81   30 |    c = i
61    99   79  888   95 |    d = o
53    59   34   96  746 |    e = u
```

## LibSVM

### LibSVM - configuración 1

```
Scheme:       weka.classifiers.functions.LibSVM -S 0 -K 2 -D
3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1
-model "C:\\Program Files\\Weka-3-7" -seed 1

Time taken to build model: 20.63 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         2818
47.1948 %
Incorrectly Classified Instances       3153
52.8052 %

=== Confusion Matrix ===

a   b   c   d   e   <-- classified as
483 175 165 260   6 |   a = a
266 453 343 199  23 |   b = e
186  34 835 269  64 |   c = i
87   83 204 731 117 |   d = o
74   77 282 239 316 |   e = u
```

**LibSVM - configuración 2**

```
Scheme:       weka.classifiers.functions.LibSVM -S 0 -K 2 -D
3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -B
-model "C:\\Program Files\\Weka-3-7" -seed 1

Time taken to build model: 42.11 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2821
47.245  %
Incorrectly Classified Instances      3150
52.755  %

=== Confusion Matrix ===

a   b   c   d   e   <-- classified as
434 215 168 258  14 |   a = a
220 492 347 192  33 |   b = e
132  46 853 250 107 |   c = i
74  86 206 713 143 |   d = o
66  96 268 229 329 |   e = u
```

**LibSVM - configuración 3**

```
Scheme:       weka.classifiers.functions.LibSVM -S 0 -K 2 -D
3 -G 0.0 -R 0.0 -N 0.5 -M 250.0 -C 1.0 -E 0.001 -P 0.1 -Z -B
-model "C:\\Program Files\\Weka-3-7" -seed 1

Time taken to build model: 44.49 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2234
37.4142 %
Incorrectly Classified Instances      3737
62.5858 %

=== Confusion Matrix ===
```

```
a   b   c   d   e   <-- classified as
334 366 121 256  12 |   a = a
269 521 268 189  37 |   b = e
145 298 648 233  64 |   c = i
33 313 250 586  40 |   d = o
40 353 256 194 145 |   e = u
```

## Multilayer Perceptron

### Multilayer Perceptron - configuración 1

```
Scheme:       weka.classifiers.functions.MultilayerPerceptron -L
0.2 -M 0.2 -N 1000 -V 0 -S 0 -E 20 -H a

Time taken to build model: 115.94 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4117
68.9499 %
Incorrectly Classified Instances      1854
31.0501 %

=== Confusion Matrix ===

a    b    c    d    e   <-- classified as
707  128  149   60   45 |    a = a
119  859  225   52   29 |    b = e
58   126 1123   73    8 |    c = i
53   120  196  773   80 |    d = o
55    61  127   90  655 |    e = u
```

### Multilayer Perceptron - configuración 2

```
Scheme:       weka.classifiers.functions.MultilayerPerceptron -L
0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Time taken to build model: 57.75 seconds

=== Stratified cross-validation ===
=== Summary ===
```

```
Correctly Classified Instances         3975
66.5718 %
Incorrectly Classified Instances       1996
33.4282 %

=== Confusion Matrix ===

a    b    c    d    e   <-- classified as
668  167  116   87   51 |    a = a
123  847  184   87   43 |    b = e
55   163 1037  110   23 |    c = i
57   145  137  795   88 |    d = o
41   91   93   135  628 |    e = u
```

(

NaiveBayes)

```
Scheme:       weka.classifiers.bayes.NaiveBayes

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         1717
28.7557 %
Incorrectly Classified Instances       4254
71.2443 %

=== Confusion Matrix ===

a   b   c   d   e   <-- classified as
457 407  14 206   5 |   a = a
475 656   2 139  12 |   b = e
220 649 284 207  28 |   c = i
211 588 149 235  39 |   d = o
137 544  37 185  85 |   e = u
```

## BayesNet

```
Scheme:       weka.classifiers.bayes.BayesNet -D -Q
weka.classifiers.bayes.net.search.local.K2 -- -P 1
-S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator
 -- -A 0.5

Time taken to build model: 0.13 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4135
69.2514 %
Incorrectly Classified Instances      1836
30.7486 %

=== Confusion Matrix ===

a    b    c    d    e   <-- classified as
684  109   96   54  146 |    a = a
170  854  102   74   84 |    b = e
76   138 1027   82   65 |    c = i
50   142   60  769  201 |    d = o
24   109   32   22  801 |    e = u
```

## OneR

```
Scheme:       weka.classifiers.rules.OneR -B 6

Time taken to build model: 0.03 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3223
53.9776 %
Incorrectly Classified Instances      2748
46.0224 %

=== Confusion Matrix ===
```

```
a   b   c   d   e   <-- classified as
565 155 144 143  82 |   a = a
190 686 187 148  73 |   b = e
156 183 836 137  76 |   c = i
135 156 116 681 134 |   d = o
101 116 134 182 455 |   e = u
```

## Random Forest

```
Scheme:       weka.classifiers.trees.RandomForest
-I 100 -K 0 -S 1 -num-slots 1

Time taken to build model: 2.35 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4913
82.281  %
Incorrectly Classified Instances      1058
17.719  %

=== Confusion Matrix ===

a    b    c    d    e   <-- classified as
877   82   47   43   40 |    a = a
80 1067   69   45   23 |    b = e
44   84 1174   70   16 |    c = i
40   63   57  987   75 |    d = o
28   52   31   69  808 |    e = u
```

## IBk

### IBk - configuración 1

```
Scheme:       weka.classifiers.lazy.IBk -K 1 -W 0 -A
 "weka.core.neighboursearch.LinearNNSearch -A
  \"weka.core.EuclideanDistance -R first-last\""

  Time taken to build model: 0 seconds
```

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4700
78.7138 %
Incorrectly Classified Instances      1271
21.2862 %

=== Confusion Matrix ===

   a    b    c    d    e   <-- classified as
 818   96   65   64   46 |    a = a
  99  993   86   72   34 |    b = e
  46   82 1143   82   35 |    c = i
  52   61   69  956   84 |    d = o
  46   33   43   76  790 |    e = u
```

**IBk - configuración 2**

```
Scheme:       weka.classifiers.lazy.IBk -K 2 -W 0 -A
 "weka.core.neighboursearch.LinearNNSearch -A
  \"weka.core.EuclideanDistance -R first-last\""

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4642
77.7424 %
Incorrectly Classified Instances      1329
22.2576 %

=== Confusion Matrix ===

   a    b    c    d    e   <-- classified as
 910   82   48   36   13 |    a = a
 172 1010   59   32   11 |    b = e
  84  126 1124   44   10 |    c = i
  94  102   92  898   36 |    d = o
  78   52   51  107  700 |    e = u
```

**IBk - configuración 3**

```
cheme:          weka.classifiers.lazy.IBk -K 4 -W 0 -A
 "weka.core.neighboursearch.LinearNNSearch -A
  \"weka.core.EuclideanDistance -R first-last\""

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4637
77.6587 %
Incorrectly Classified Instances      1334
22.3413 %

=== Confusion Matrix ===

   a    b    c    d    e   <-- classified as
 861   74   63   55   36 |    a = a
 149 1000   68   46   21 |    b = e
  68   93 1132   71   24 |    c = i
  65   94   87  897   79 |    d = o
  54   51   40   96  747 |    e = u
```