UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

SEGUIMIENTO DEL MARCAJE EN LA RUTA MEDIANTE CAPTURA DE VIDEO Y APRENDIZAJE AUTOMÁTICO

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERÍA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

NATASHA TERESA GARCÍA ROBLES

DIRECTOR: DR. DIEGO ORDÓÑEZ CAMACHO

Quito, Enero 2016

DERECHOS DE AUTOR

DECLARACIÓN

Yo **NATASHA TERESA GARCÍA ROBLES**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Natasha García Robles

C.I. 0401496302

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título **“Seguimiento del marcaje en la ruta mediante captura de video y aprendizaje automático**”, que, para aspirar al título de **Ingeniera en Informática y Ciencias de la Computación** fue desarrollado por **Natasha Teresa García Robles**, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 18 y 25.

_____

Dr. Diego Ordóñez Camacho

Director del trabajo de titulación

C.I. 1710449016

DEDICATORIA

Dedico la ejecución de este, uno de mis grandes sueños a mi abuelito Campo Elías Robles, porque bastaron 18 primaveras para guardar su amor, su ejemplo y su actitud ante cada estación de la vida por siempre.

A mis abuelitos Robert, Pachita y Teresita quienes adornan mi vida con sus enseñanzas y su amor infinito.

A mis maestros de vida, mis padres Jaime y Jackie, por su gran ejemplo de superación, perseverancia y bondad, con su apoyo siempre la vida es color de rosa. Especialmente a ese ser especial llamado mamá, porque en el mundo no he encontrado la perfección, sin embargo ella se le aproxima, porque me acompaña en las caídas y alegrías, y mis logros son suyos.

A mis hermanas Anggie y Melanie, mis amigas de siempre y para siempre, para que sigan mis pasos firmes, y en cambio los pasos flojos los esquiven.

A Christopher quien me enseña día tras día a ser feliz y aprender de las cosas simples, porque encontré el eco de mi risa y el acompañante de mis sueños. Este será uno de tantos logros que espero sean de inspiración para quien camina junto a mí, para que su vuelo sea sobre las estrellas.

AGRADECIMIENTO

Gracias Dios, cada logro sin duda es por ti. Contigo todo, sin ti nada.

A mi equipo incondicional, mi familia.

A la Universidad, mi segundo hogar de donde me llevo los mejores aprendizajes y amistades sinceras.

A mi tutor Dr. Diego Ordóñez, por sus consejos siempre acertados para la culminación del proyecto y su gran ejemplo de preparación y ética dentro y fuera de las aulas.

A mis profesores, en especial a Freddy un amigo que nos enseñó el significado de vivir, luchar y al mismo tiempo tener fe.

A mis amigos quienes compartieron el mismo camino y a quienes me ayudaron durante el camino.

TABLA DE CONTENIDO

Página

ÍNDICE DE TABLAS

Página

ÍNDICE DE FIGURAS

Página

ÍNDICE DE ANEXOS

Página

RESUMEN

El presente trabajo de titulación expone el análisis, diseño e implementación de una aplicación para dispositivos móviles, que sea capaz de dirigir de manera autónoma un robot en su desplazamiento. Utiliza inteligencia artificial para el reconocimiento de la ruta que se obtiene por medio de la cámara del dispositivo.

El documento contiene el análisis de requerimientos utilizando el estándar IEEE830, donde se presentan los procesos con la ayuda del lenguaje UML. Siguiendo la metodología OOHDM se expone el diseño navegacional e interfaz abstracta de la aplicación y por último se detalla el proceso para la implementación y pruebas realizadas durante el desarrollo de la aplicación y el software para obtener el modelo de inteligencia artificial.

Las herramientas utilizadas para el desarrollo del proyecto fueron basadas en software libre, siendo Netbeans y Eclipse los entornos de desarrollo; con lenguaje de programación Java y con las librerías de código libre OpenCV y Weka, para visión artificial y aprendizaje automático respectivamente.

Para el reconocimiento de la ruta se muestra una serie de pruebas realizadas hasta obtener el modelo adecuado para el aprendizaje automático del robot. Se sugiere además los principales métodos de visión artificial hasta obtener la aplicación para el seguimiento de marcaje de ruta para el robot.

ABSTRACT

This undergraduate thesis exposes the analysis, design and implementation of an application for a mobile devices, that will be able to direct, autonomously, a robot as it moves along a track. Artificial intelligence is used for the recognition of the route that is obtained with the device camera.

The thesis contains the analysis of the requirements using the standard IEEE830 where the processes are presented using the UML language Process. According to the OOHDM methodology, the navigational design and abstract interfaces of the application are exposed and finally the process for implementation and tests during the development of the application and the software are detailed to obtain the model of artificial intelligence.

The tools used for the development of the proyect were based on free software, where Netbeans and Eclipse were the development environment; with Java programming language and open source libraries OpenCV and Weka for computer vision and machine learning respectively.

For the recognition of the route test series are shown to obtain the appropiate model for the automatic learning of the robot. Finally the main methods of computer visión are suggested to obtain the application for monitoring the marking of the route for the robot.

INTRODUCCION

2. INTRODUCCIÓN

La utilización de la robótica es una alternativa para que el ser humano ya no realice tareas peligrosas, tediosas y que no necesiten un razonamiento complejo para ser ejecutadas. (Hidalgo & Haj, 2014)

En la actualidad existen varias aplicaciones para robots, Google por ejemplo introdujo al mercado el automóvil autónomo, conformado, entre otros elementos, por sensores mediante los cuales puede detectar obstáculos en el camino y señales de tránsito, y que mediante captura de video en tiempo real pueda dirigirse a cualquier sitio que Google Maps reconozca. Este proyecto teóricamente se dirige a personas invidentes además a personas con cierta discapacidad que le impida conducir un automóvil. (Itakura et al., 2008).

Hoy en día existen aplicaciones Android para controlar las actividades de un robot, mediante órdenes enviadas por el usuario. El robot es capaz de movilizarse y tomar decisiones de hacia dónde dirigirse sin salirse de la ruta, requiriendo el uso de las herramientas para aprendizaje automático (Alpaydin, 2009). Una de estas aplicaciones es la que se está utilizando en más de 50 bibliotecas en el medio oriente para transportar el material requerido y realizar inventario. (Thirumurugan, Vinoth, Kartheeswaran, & Vishwanathan, 2010)

Realizar una aplicación de seguimiento de ruta para un robot sería encontrar un conjunto de herramientas apropiadas para solucionar este tipo de problemas y que ayude para posteriores investigaciones en temas relacionados con visión por computador y aprendizaje automático en sistema operativo Android.

Considerando esto y los altos costos que implica el desarrollo de estos sistemas que involucran robots, procesamiento de imágenes, y aprendizaje automático, se ve la necesidad de buscar herramientas libres. Una opción son las tecnologías OpenCV (Cyganek, 2013) y Weka (Kaluza, 2013) librerías de visión computarizada y aprendizaje automático respectivamente.

Las librerías OpenCV permiten convertir una imagen en una matriz, para luego transformar esta matriz, con diferentes filtros, a diferentes tipos de imagen, y con esto poder instruir al robot para que realice una acción mediante aprendizaje automático. La librería que se ajusta es Weka. (Itakura et al., 2008)

El objetivo general del proyecto es diseñar e implementar una Aplicación Android-OpenCV-Weka que permita analizar en tiempo real un flujo continuo de imágenes en movimiento, procesar dichas imágenes para con ellas obtener la dirección a seguir por el robot.

Los objetivos específicos de la investigación son: a) encontrar el algoritmo indicado para realizar el entrenamiento de seguimiento de una ruta del robot, b) entrenar al robot para que sea capaz de seguir el marcaje de la ruta por sí solo, d) encontrar las técnicas, herramientas y métodos adecuados para la implementación de la aplicación Android para el robot.

MARCO TEÓRICO

3. MARCO TEÓRICO

Para la realización de la aplicación Android es necesario comprender algunos conceptos para entender el proceso que por medio del dispositivo móvil produce el movimiento del robot. Primero y como concepto global se tiene la Inteligencia Artificial que encapsula los temas de sistemas inteligentes, sistemas expertos, para que la máquina, en este caso el robot pueda tomar sus propias decisiones. Para el movimiento del robot se debe decidir con qué herramientas se puede trabajar, los algoritmos que se van a utilizar para el reconocimiento de patrones, y hacia que dispositivos móviles va dirigido el proyecto.

A continuación se explica todos los conceptos que hacen referencia al proyecto, así como también algunos algoritmos, herramientas, métodos y metodologías de utilidad.

2.1. INTELIGENCIA ARTIFICIAL

La inteligencia artificial es la ciencia que se encarga de la creación de sistemas de computación inteligentes que puedan actuar de manera similar a los seres humanos. También se la define como el conjunto de programas informáticos que realizan actividades o tareas, que al ser realizadas por el ser humano se consideran inteligentes(Henao, 2009).

Dichas actividades o tareas pueden constar dentro de áreas genéricas como aprendizaje, percepción, resolución de problemas y áreas específicas como jugar ajedrez, diagnosticar enfermedades, conducir coches, etc. (Russell, 2004).

Las áreas de la Inteligencia Artificial intentan simular las actividades complejas que caracterizan al ser humano mediante el empleo de sistemas informáticos, estos sistemas se denominan inteligentes, dentro de los cuales se encuentran los sistemas expertos.

2.1.1. SISTEMAS EXPERTOS

Uno de los desarrollos más importantes logrados por la Inteligencia Artificial, son los llamados sistemas expertos, o sistemas basados en el conocimiento. Estos son

programas computacionales que en un dominio específico ofrecen soluciones iguales a las que un experto humano daría. (Rosales, 2004)

También se define como un sistema experto a una máquina que piensa y razona como una persona experta en el campo de acción lo haría; que maneja una gran cantidad de datos y los manipula de manera inteligente, para que el resultado pueda resolver problemas incluso más rápido que un experto humano.

Estos sistemas son capaces de procesar y memorizar información, aprender y razonar en situaciones deterministas e inciertas para poder tomar las decisiones más certeras (Huete, 1998)

2.1.2. MACHINE LEARNING

Una subcategoría de lo que comprende la Inteligencia Artificial es Machine Learning, traducido al español como Aprendizaje Automático. Tiene como principal función crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos, ahora reflejados en sistemas de aprendizaje automático y aplicaciones en la industria, la medicina, la economía, entre otros (Sebe, Cohen, & Garg, 2006).

Machine Learning se utiliza para el análisis de datos y descubrimiento de conocimiento en bases de datos, en la generación automática de conocimiento para sistemas expertos, para la construcción de modelos numéricos y cualitativos, clasificación de texto, control dinámico y automático de procesos, para reconocer automáticamente la voz, escritura, imágenes, etc. (Kononenko & Kukar, 2007)

2.1.2.1. Algoritmos

Los algoritmos de visión artificial están bien establecidos hoy en el campo de la asistencia a la conducción. Limitaciones con estos métodos pueden ocurrir cuando las nuevas tareas como la detección de clases múltiples y / o múltiples puntos de vista deben ser abordados. (Jain, Duin, & Mao, 2000)

REDES NEURONALES

Una red neuronal artificial es un modelo computacional que consta de varias neuronas o unidades sencillas que trabajan en paralelo. Todas las neuronas tienen entradas que se unen para obtener una única salida por medio de la activación. La conexión entre cada par de neuronas tiene asociado un peso que representa el aprendizaje que el algoritmo o método matemático puede modificar. (Araujo, Ferrari Alve, & Mariño, 2012)

La estructura más popular de las redes neuronales es conocida como *feedforward*, relacionado con las conexiones hacia adelante. Esta estructura se caracteriza por la configuración en capas, una para la entrada, una para la salida y varias intermedias, como se muestra la Figura 1.

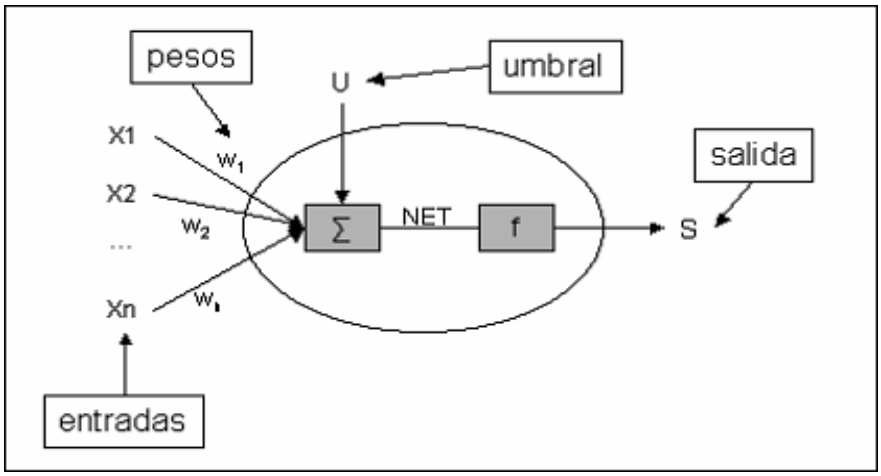

Figura 1. Revisión de algoritmos de redes neuronales (Araujo, Ferrari Alve, & Mariño, 2012)

En cada capa, excepto en la de entrada, existe un número de neuronas que reciben conexiones ponderadas por peso de la capa anterior. La información fluye desde las entradas hacia las capas intermedias para, finalmente, generar una o varias salidas en la capa de salida. (Rairán, Chiquiza, & Parra, 2012)

Se puede clasificar las Redes Neuronales según sean capaces de procesar información de distinto tipo en valores de entradas binarias, donde se procesa datos

discretos de entrada y valores de entrada continua, los cuales procesan datos de entrada con valores continuos, en su mayoría de veces acotados. (Araujo et al., 2012)

LOGICA DIFUSA

La lógica difusa ha sido probada en diversos campos de la ingeniería en la toma de decisiones, permite analizar observaciones en el tiempo clasificándolas en categorías lingüísticas, aproximadas a la calificación empleada o a la manera en que una persona describiría un fenómeno.

Como modelo de sistema inteligente, se considera la lógica difusa como una combinación de sistemas basados en reglas en conjunto con la inteligencia computacional que comprende las redes neuronales. Se puede utilizar en cualquier área de control inteligente o procesamiento de datos en tiempo real. (Segura, Obregón, & Neira, 2005)

ÁRBOL DE DECISIÓN

Una de las formas de aprendizaje de un conjunto de instancias independientes hacia un estilo de representación se conoce como árbol de decisión. Los nodos en un árbol de decisión involucran pruebas de atributos en particular. Con frecuencia, una prueba compara el valor del atributo con una constante. Esto conlleva a la clasificación de todas las instancias o la distribución de probabilidades de todas las posibles combinaciones. Para clasificar una instancia desconocida se acuerda para que el valor de los atributos realice pruebas en los nodos secuencialmente.

El problema de construir un árbol de decisión radica en poderlo expresar recursivamente. Lo que hace es primero seleccionar un atributo para el lugar donde se encuentra el nodo raíz y hacer una rama para cada valor posible. (Witten, Frank, & Hall, 2011)

Un árbol de decisión reduce significativamente el tiempo computacional y la simplicidad para realizar el análisis. Además, proporciona un conjunto de reglas muy fácilmente interpretables. (Stanislawski, Kotulska, & Unold, 2013)

VECINOS MÁS CERCANOS

Este algoritmo es de clasificación supervisada basado en reconocimiento de patrones y criterios de vecindad, también se le conoce como clasificador k-nn. El concepto del algoritmo es clasificar una muestra nueva dependiendo de cuál sea la mayor cantidad de vecinos más cercanos.

Al aplicar el algoritmo se toma en consideración todo el conjunto de entrenamiento para determinar la clase a la que pertenece la nueva muestra, en donde se toma en cuenta los k vecinos más cercanos. El k es una variable que se determina dependiendo de los vecinos más cercanos que se quieran tomar en cuenta. (Chapman & Hall, 2009)

Formalmente se define la vecindad como el conjunto de entrenamiento de N prototipos pertenecientes a M clases distintas que determina la clase a la cual pertenece una nueva muestra x la cual es la más votada por sus k vecinos más cercanos.

En la Figura 2 se tiene en el conjunto de entrenamiento de 21 prototipos, con 2 clases distintas para los 3 vecinos más cercanos.

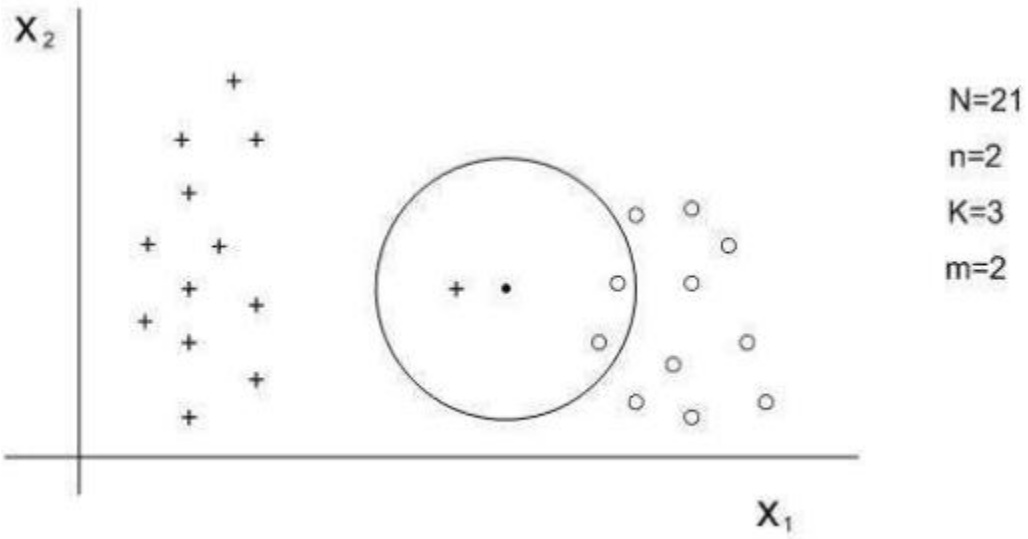

Figura 2. Clasificador basado en la regla de los k vecinos más próximos (Chapman & Hall, 2009)

2.1.2.2. Herramientas

Existe una diversidad de herramientas de software desarrollado bajo licencia GPL (Licencia Pública General), como: WEKA, RapidMiner, Matlab, Pentaho, Orange, SSPS Clementine, entre otros. A continuación se detalla las más utilizadas.

WEKA

La herramienta Weka es un conjunto de librerías Java para la extracción de conocimiento desde una base de datos, consta de algoritmos de aprendizaje automático para las tareas que realiza la minería de datos (The Univesity of Waikato, 2014). Estas herramientas son para el pre procesamiento de datos, clasificación, regresión, agrupamiento, reglas de asociación y visualización (Bouckaert et al., 2010).

Una vez que se procesa las imágenes, se obtiene un archivo de texto con la información de cada imagen en un vector binario, denominado vector de caracterización. Antes de alimentar la red, se debe procesar estos archivos que se presentan en un formato apto para la herramienta empleada. Una vez que se obtiene el archivo adecuado para Weka, se carga los datos y el correspondiente algoritmo de entrenamiento. (Esperón, Laudadio, Martínez, Serrano, & López, 2008)

Para clasificar las imágenes usando Weka es necesario extraer algunos parámetros que describan las imágenes para luego colocarlos en una base de datos y se pueda realizar el tratamiento respectivo y por último la clasificación. (Vera, 2013)

RAPIDMINER

RapidMiner es una solución para la minería de datos de código abierto. Trabaja con la construcción de bloques para los diferentes datos de transformación o tareas de análisis llamados operadores. Estos operadores son presentados en grupos, y se muestran a continuación:

- **Control de proceso:** Contiene operadores como bucles los cuales toman el control de un proceso.
- **Utilidad**: Operador auxiliar para agrupar subprocesos.

- **Acceso al repositorio:** Contiene dos operadores para leer y escribir acceso en los repositorios.
- **Importación**: Contiene operadores para leer datos y objetos de formatos externos como archivos, bases de datos, etc.
- **Exportación**: Contiene un gran número de operadores de escritura de datos y objetos para formatos externos como archivos, bases de datos, etc.
- **Transformación de datos**: Probablemente el grupo más importante en cuanto al análisis en términos de importancia y tamaño. Todos los operadores son localizados aquí para la transformación de los datos y los metadatos.
- **Modelamiento**: Contiene el proceso actual de los datos, como los métodos de clasificación, métodos de regresión, agrupamiento, ponderación, métodos para reglas de asociación, correlación y análisis de similitud como operadores, en orden para ser aplicados en un nuevo modelo de un conjunto de datos.
- **Evaluación**: Operadores que se usan cada uno para poder calcular la calidad del modelo y así para nuevos datos. (Hofmann & Klinkenberg, 2013)

MATLAB

MATLAB implementa y proporciona todas las funcionalidades que se espera que una fina calculadora científica lo pudiera hacer (Wallisch et al., 2014).

El procesamiento de imágenes no es una característica incorporada en MATLAB, sin embargo mientras se pueda cargar, guardar y realizar tareas básicas, el resto lo realizan las herramientas de procesamiento de imágenes adicionales. Estas herramientas son compatibles con formatos comunes de imágenes JPEG, TIFF, PNG y los formatos menos comunes como BIP y BIL. Además el tamaño del archivo no afecta para trabajar en MATLAB ya que incluye procesos de flujo de trabajo para manipular archivos de gran tamaño. (Jiménez, Navarro, & Cisneros, 2010)

2.2. RECONOCIMIENTO DE PATRONES

Un patrón se puede definir como una entidad. Dado un patrón el reconocimiento y clasificación consiste en una de las siguientes tareas:

- **Supervisado**

(Análisis de discriminación) en el cual el patrón de entrada está definido como un miembro de una clase predefinida.

- **No supervisado**

El patrón es asignado a cualquier agrupamiento de clase.

El problema del reconocimiento de patrones está dentro de las tareas de clasificación y categorización, donde las clases están definidas por el diseño del sistema. En el caso donde se supervisa la clasificación. Si no se supervisa la clasificación entonces aprende basado en patrones similares.

Uno de los principales enfoques es la utilización de una plantilla de comparación, donde se entiende a la comparación como una operación genérica dentro del reconocimiento de patrones, que se utiliza para determinar la similitud entre dos entidades del mismo tipo de dato. (Jain, Duin, & Mao, 2000)

2.3. VISIÓN POR COMPUTADOR

Visión por Computador, es la transformación de los datos de una cámara fotográfica o de video en cualquier decisión o nueva representación.

Una nueva representación puede significar convertir una imagen en color en una imagen en escala de grises o la eliminación de movimiento de la cámara a partir de una secuencia de imágenes. (Bradski & Kaehler, 2008)

2.3.1. ETAPAS

La visión por computador sirve para obtener una habilidad para reconocimiento de objetos dentro de la colección de una imagen, y luego pueda procesar e interpretar esta imagen. Las etapas de visión por computador son las siguientes:

2.3.1.1. Detección de la imagen

- Calibrar la cámara, donde se toma en cuenta la orientación, corrección de la imagen, etc.

- Iluminación, Se necesita para detectar y extraer objetos de una imagen, se establece una resolución adecuada para minimizar la complejidad
- Geometría de la imagen, Perspectivas

2.3.1.2. Preprocesamiento de la imagen

Como segundo aspecto está el preprocesamiento de la imagen, donde surgen dos metodologías:

- Dominio espacial, trabaja directamente con el arreglo de los pixeles de la imagen
- Dominio de frecuencia, transformada de Fourier, una de sus principales actividades es que reduce ruido en la imagen.

2.3.1.3. Segmentación de la Imagen

El siguiente paso es la segmentación de la imagen, donde la divide en sus partes constituyentes, los objetos se extraen de una escena y posteriormente se reconoce y analiza estos objetos, para ello existen algoritmos como:

- Discontinuidad, para detección de bordes
- Similitud, para la región creciente

2.3.1.4. Descripción de la Imagen

Luego se realiza una descripción de la imagen, donde se extrae las características de un objeto en particular que se lo quiera reconocer con algún fin.

2.3.1.5. Reconocimiento de la Imagen

Posteriormente se realiza el reconocimiento de la imagen, mediante metodologías como:

- Decisión teórica, existen dos formas por propiedad de decisión o funciones de discriminación para selección del patrón u objeto definido.
- Estructural, donde un objeto se descompone en un set de elementos primitivos por un tamaño y dirección ya definidas.

2.3.1.6. Interpretación de la Imagen

Por último se realiza la interpretación de la imagen. Es el nivel más alto de todo el proceso por el que la imagen pasa.

- Multiescala, base para buscar la imagen principal.
- Búsqueda secuencial, se examina pixel por pixel.

2.3.2. HERRAMIENTAS

2.3.2.1. OpenCV

OpenCV es una biblioteca multiplataforma de código abierto que proporciona bloques de construcción para los experimentos de visión por ordenador. Ofrece interfaces de alto nivel para captura, procesamiento, y presentación de datos de imagen. (Howse, 2013)

Esta herramienta facilita la creación de un entorno de desarrollo adecuado y la traducción de la amplia funcionalidad de las bibliotecas en las características de la aplicación son a la vez tareas de enormes proporciones. (Bradski & Kaehler, 2008)

Estructura de OpenCV

Estas librerías se dividen en cinco grupos, divididas las funciones dependiendo de su utilidad: CXCORE, contiene las estructuras de datos funciones de soporte para álgebra lineal; CV donde se encuentran las funciones de procesamiento de imágenes y algoritmos de visión; HighGUI, con todo lo relacionado a la interfaz gráfica de OpenCV y las funciones para importar imágenes y video; ML, que cuenta con algoritmos de aprendizaje, clasificadores y demás; y CvAux, que tiene las funciones experimentales.(Jurado, Asbusac, & EspaCursos, n.d.)

Patrones de Calibración para OpenCV

La calibración de la cámara determina los parámetros que explican la proyección de un objeto en 3 dimensiones al plano de la cámara. Estos parámetros se definen en escenarios conocidos llamados patrones de calibración. Cualquier objeto puede ser utilizado como un objeto de calibración, pero la opción más práctica es el de un patrón regular, como por ejemplo un tablero de ajedrez. (Coronado, 2014)

2.3.3. ALGORITMOS

2.3.3.1. Canny Edge Detector

El algoritmo de Canny permite la manipulación de la imagen, sirve para la detección de bordes y satisface los siguientes criterios:

- Baja tasa de error: buena detección de solo bordes existentes
- Buena localización: la distancia entre los pixeles del borde detectado y los pixeles de borde real debe ser mínima.
- Respuesta mínima: solo una respuesta del detector de borde

Este programa pide al usuario que introduzca un valor numérico para establecer el umbral más bajo para nuestro detector de Canny Edge, aplica el detector y genera una máscara (líneas brillantes que representan los bordes sobre un fondo negro). (Huamán, 2015)

Figura 3. Imagen sin filtro Canny (Huamán, 2015)

En la Figura 3 se indica la imagen captura sin el procesamiento de la imagen, el resultado después de utilizar el algoritmo de Canny será la misma imagen transformada en binaria y resaltando únicamente las partes donde había bordes, tal como muestra la Figura 4.

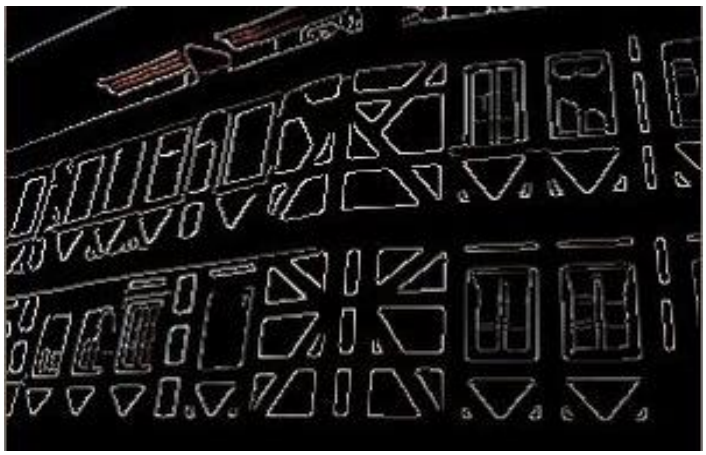

Figura 4. Imagen con filtro Canny (Huamán, 2015)

2.3.3.2. Hough line transform

Este algoritmo permite detectar líneas rectas. Para aplicar la trasformación es conveniente un preprocesamiento con el algoritmo de Canny (detección de bordes).

Para utilizar este algoritmo primero se debe realizar la carga de la imagen, luego aplicar la trasformación de línea estándar y por ultimo visualizar la imagen original y la detección de la línea en 2 ventanas. (Huamán, 2015)

Figura 5. Imagen sin filtro (Huamán, 2015)

Si tenemos como imagen de entrada la Figura 5 y se utiliza la función Hough Line Transform, se obtiene la Figura 6.

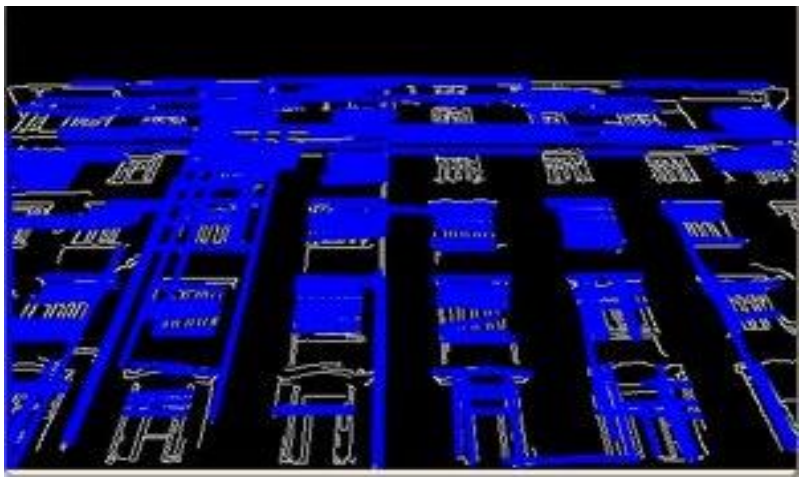

Figura 6. Imagen con filtro Canny (Huamán, 2015)

2.3.3.3. Cascade classifier

Se utiliza el algoritmo de Cascade Classifier para detectar objetos en una secuencia de video. Se utilizan las funciones de cargar, que permite cargar un archivo clasificador .xml y la función detectMultiScale para llevar a cabo la detección del objeto.

En la Figura 7 se indica un ejemplo donde se aplica el algoritmo de Cascade Classifier para la detección de la cara frontal, para esto dentro de la librería se carga el archivo en este caso haarcascade_forntalface_alt.xml y haarcascade_eye_tree_eyeglasses.xml en el directorio localizados dentro de opencv/data/cascades.

Figura 7. Imagen con filtro Cascade classifier (Huamán, 2015)

2.4. ROBÓTICA

El término robótica define a la rama de la tecnología que se ocupa del estudio, desarrollo y aplicaciones de los robots. Los robots son dispositivos compuestos de sensores que reciben datos de entrada y están conectados a una computadora o dispositivos móvil para por medio de estas entradas realice alguna acción. Puede ser que los propios robots dispongan de microprocesadores y estos ordenan para que el robot ejecute alguna determinada acción.

El nombre de robot se toma del vocablo “robota” que significa siervo y en ruso significa trabajo arduo, repetitivo, monótono. (Chong, 2009)

2.4.1. ROBOTS MÓVILES

El desarrollo de robots móviles responde a la necesidad de extender el campo de aplicación de la Robótica. Se trata también de incrementar la autonomía limitando todo lo posible la intervención humana. (Baturone, 2001)

Los robots móviles cubren muchos campos de aplicación, como son transporte de materiales, labores de limpieza, guiado de personas invidentes, éstos también se denominan robots autónomos pues se trata de que realicen por si solos cualquier actividad. (Simonite, 2013)

2.4.2. ROBOTS AUTÓNOMOS

Los robots autónomos son sistemas mecánicos que pueden interactuar en un ambiente, y tiene un comportamiento individual aunque éste sea socializado. Las dificultades que se les presenta son círculos impredecibles y dinámicamente cambiantes, así como si limitada autonomía y sobre todo su posicionamiento en un determinado ambiente. (Simonite, 2013)

2.5. DISPOSITIVOS MÓVILES

Los teléfonos celulares, así como todos los dispositivos móviles, son aparatos que en la actualidad utiliza la mayoría de la gente. En la actualidad ya no se considera que tener uno de estos aparatos sea un lujo, sino una necesidad. Dependiendo del usuario se desarrollan aplicaciones que satisfagan necesidades en común de la gente. (Becerril, 2008)

2.5.1. APLICACIÓN MÓVIL

Una aplicación móvil o app es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Por lo general se encuentran disponibles a través de plataformas de distribución, operadas por las compañías propietarias de los sistemas operativos móviles como Android, iOS, BlackBerry OS, Windows Phone, entre otros.

2.5.2. ANDROID

Android es un sistema operativo para dispositivos móviles, que está basado en Linux. La principal ventaja que adopta Android es que ofrece un enfoque unificado para el desarrollo de aplicaciones. La aplicación que se realice se podrá ejecutar en diferentes dispositivos móviles.

Las herramientas requeridas para el desarrollo de las aplicaciones Android son:

- **Eclipse**

El primer paso es seleccionar un entorno de desarrollo integrado (IDE), en el caso de Android el IDE recomendado es Eclipse para Java EE.

- **Android SDK**

Contiene el compilador, librerías, emulador, documentación, código simple y tutoriales.

- **Android Development Tools (ADT)**

Permite crear nuevos proyectos, aplicaciones para Android, acceder a las herramientas para emuladores y dispositivos Android, compilar y ejecutar aplicaciones, exportar una aplicación dentro de un paquete apk y crear certificados digitales para la firma del código del apk.

2.6. METODOLOGÍAS DE DESARROLLO

2.6.1. PROTOTIPADO

Una de las herramientas más funcionales que existen a la hora de desarrollar es la de prototipado. El proceso de desarrollo de aplicaciones, al igual que otros tipos de software, se ve sustancialmente mejorado, se ahorra tiempo y dinero, y sobre todo se puede tener al cliente bien informado sobre qué va a tener desde antes de programar una sola línea de código. (“Prototipado de aplicaciones moviles,” n.d.)

El prototipado consta de las siguientes fases:

Plan. Esta fase ayuda a decidir qué aspectos del software deben o no ser implementados en el prototipado del proyecto. Se divide en los siguientes pasos: Verifica los requerimientos de la aplicación, crea un flujo de las pantallas del sistema, se especifica el contenido.

Especificación. En esta fase se decide cómo hacer el prototipado, las características dan un mejor de entendimiento para definir la fidelidad en cuanto a qué necesita el prototipado. Se determina las características, se escoge los métodos y las herramientas que se van a utilizar.

Diseño. Se formula un criterio de diseño, y se crea el prototipo.

Resultados. Se revisa el prototipo con los usuarios, se valida el diseño, y se implementa el diseño.

2.6.2. METODOLOGÍA XP

La programación extrema XP usa un enfoque orientado a objetos y engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales: planeación, diseño, codificación y pruebas.

Planeación. Es la actividad para recabar requerimientos que permite que los miembros técnicos del equipo XP entiendan el contexto del negocio para el software y adquieran la sensibilidad de la salida y características principales y funcionalidad que se requieren, este proceso lleva a la creación de algunas historia, que describen la salida necesaria, características y funcionalidad del software que se va a elaborar.

A medida que avanza el trabajo el cliente puede agregar historias, cambiar la prioridad de una ya existente, descomponerlas o eliminarlas. Entonces, el equipo XP reconsidera todas las entradas faltantes y modifica sus planes en consecuencia.

El proceso de la metodología XP consta de las siguientes etapas:

Diseño. Es el proceso mediante el cual se realiza un diseño sobre una representación más compleja. A demás, el diseño guía una implementación de una historia conforme se escribe.

Estimula el rediseño, técnica de construcción que también es un método para la optimización del diseño en forma tal que no altere el comportamiento externo del código, pero si mejora la estructura interna. Rediseñar significa que el diseño se hace de manera continua según se construye el sistema.

Codificación. Después de que las historias han sido desarrolladas se aplica una serie de pruebas a cada una de la historias que se incluirán en la entrega en curso. Una vez que el código está terminado se le aplica de inmediato una prueba unitaria, con lo que se obtiene retro alimentación instantánea para los desarrolladores.

Pruebas. Se realizan pruebas unitarias, se centran en un componente de software individual, en un esfuerzo por descubrir errores locales del componente. A medida que se realizan las pruebas unitarias en un grupo de prueba universal, las pruebas de integración y validación del sistema pueden efectuarse a diario.

Las pruebas de aceptación XP, se centran en las características y funcionalidad generales del sistema que son visibles y revisables por parte del cliente.

2.6.3. METODOLOGÍA AUP

Es una metodología que aplica técnicas ágiles, por sus siglas Metodología Ágil de Procesos Unificados. Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software, realizando siempre iteraciones dentro del proyecto y descomponiendo en proyectos más pequeños para tener mayor control en cada iteración. (Gualotuña, Ramírez, Jaramillo, & Campiña, 2014)

Esta metodología se enfoca en la determinación de riesgos en etapas tempranas, para que la aplicación sea adaptable en la gestión de cambios y el proceso que sigue se resume en las siguientes fases:

Incepción. En esta etapa se identifica el alcance, la dimensión del proyecto y las limitaciones, revisando los requerimientos que fueron determinados en un inicio. Además se analizan los riesgos que podrían presentarse en el proyecto y la arquitectura a emplearse.

Elaboración. Es la construcción del sistema, además se confirma la idoneidad de la arquitectura.

Construcción. Se construye el software siguiendo una base iterativa e incremental dependiendo de las prioridades del cliente.

Transición. Se valida e implementa el sistema.

2.6.4. METODOLOGIA OOHDM

La metodología OOHDM o también conocida como metodología de diseño hipermedia orientada a objetos, es ampliamente aceptada para el desarrollo de aplicaciones hipermedia, un conjunto de métodos y procedimientos que integren soportes como: texto, imagen, video, audio, mapas, entre otros soportes de información. (Céspedes & González, s.f)

Esta metodología es muy utilizada en el desarrollo de aplicaciones móviles, y tiene las siguientes etapas:

Determinación de Requerimientos: Se recopila la información necesaria para detallar la funcionalidad, actividades y procesos que conforman la creación del portal.

Diseño Conceptual. Se construye un esquema conceptual representando las clases y las relaciones entre las mismas.

Diseño de Navegación. Se define los enlaces y estructuras de acceso.

Diseño de Interfaz Abstracta. Se define la forma en la que aparecen los contextos de navegación. Se definen todas las funcionalidades de la aplicación.

Implementación. Dedicada a la puesta en marcha de la aplicación.

2.6.5. LENGUAJE UML

El lenguaje UML es un estándar diseñado para visualizar, especificar, construir y documentar software orientado a objetos. El modelo simplifica la realidad y ayuda en la comunicación de la estructura del sistema, comprender mejor el sistema y descubrir oportunidades de simplificación y reutilización. (Booch & Rumbaugh, 2004)

Diagramas de clases. Muestran el conjunto de clases y sus relaciones, representan el diseño estructural del sistema.

Diagramas de secuencia. Muestran la secuencia de mensajes entre objetos durante un escenario concreto.

- En la parte superior están los objetos que intervienen, ya sea usuarios, clases, interfaces
- El tiempo se lo representa con líneas verticales, y determinan el tiempo de vida de cada objeto
- El paso de mensajes se lo representa con flechas horizontales
- La realización de una acción se representa con rectángulos sobre la línea de actividad

Diagramas de casos de uso.

Los diagramas de casos de uso suelen ser utilizados en el modelado del sistema para representar las acciones que realiza cada tipo de usuario.

- Los casos de uso se representan mediante elipses con el nombre del caso y los actores mediante un monigote.
- Las asociaciones entre actores y casos de uso se representan mediante una línea continua, que significa la participación del actor en el caso de uso.
- Entre casos de uso se puede dar relación de extensión, inclusión generalización.
- Una relación de inclusión significa que se heredan comportamientos, mientras que una relación de extensión es cuando se podría incorporar algún comportamiento especificado. (Vega, 2010)

2.7. ESTADO DEL ARTE

Sistema de gestión de información de una biblioteca por medio de un robot seguidor de línea

En Pakistán y el Medio Oriente están utilizando en más de 50 bibliotecas, para que el manejo del sistema no lo realice una persona se implementó el sistema en un robot seguidor de línea utilizando aprendizaje automático. Este robot está diseñado utilizando motores de sensores operados para mantener un seguimiento de la ruta de la línea predeterminada para los arreglos de estantes de libros de la biblioteca.

El robot lleva un lector de código de barras que recoge los datos de código de los libros dispuestos de manera y compara los datos de código de barras descodificados con la entrada de búsqueda.

Si el robot alcanza el libro entonces busca la ubicación del libro, en caso de cualquier obstáculo que enfrenta el robot, el robot se detiene y envía una alarma. (Thirumurugan et al., 2010)

Perro guía robótico para personas invidentes

Los ingenieros japoneses de la compañía NSK, con la ayuda de la Universidad de Electro-Comunicaciones de Tokio, han desarrollado un perro guía robótico para personas invidentes, que utiliza una Kinect (controlador de juego libre y entretenimiento desarrollado por Microsoft) para evaluar y comprender el entorno que rodea al robot, con el fin de que el usuario pueda caminar con seguridad por cualquier lugar. Las nuevas versiones incluirán comandos de voz, y GPS para una mejor navegación. (Castro, 2011)

Sistema de visión del vehículo no tripulado Stanley

Otro proyecto con visión computarizada está el Sistema de visión del vehículo no tripulado Stanley de la Universidad de Stanford, el ganador en el año 2005 del Gran desafío DARPA. Este proyecto utilizaba sistema de vigilancia de vídeo y programa Swistrack, una herramienta de seguimiento distribuida.

Este auto tiene sensores con láser, radares y un sistema de visión monocular que le permiten superar los obstáculos del camino. En 2005, consiguió recorrer en 6 horas y 53 minutos la ruta de 212 kilómetros a través del desierto de Mojave, en los Estados Unidos. (Rosario, 2014)

METODOLOGÍA

3. METODOLOGÍA

Se realizó el diseño e implementación de la aplicación Android para un robot utilizando herramientas OpenCV y Weka que permiten analizar en tiempo real un flujo continuo de imágenes en movimiento y detectar la dirección a seguir por el robot para mantenerse en la ruta. A continuación se detalla el alcance, las herramientas y metodología que se utilizó para concluir con el proyecto.

3.1. ALCANCE

La aplicación Android es capaz de reconocer la ruta por la cámara del dispositivo móvil una vez que se conecta con el robot mediante la tecnología Bluetooth que recibe la dirección a través de comandos para que el robot los recepte y se movilice, en este proceso se tomó en cuenta los siguientes aspectos:

- Se evaluó algoritmos disponibles en la librería OpenCV para visión computarizada en sistemas operativos Android.
- Se elaboró modelos de aprendizaje automático en lenguaje Java para escoger el mejor modelo utilizando la librería Weka.
- El patrón de reconocimiento, es decir la ruta a seguir por el robot, será una línea negra con mínimo 5 cm de ancho y el ambiente propicio con la luz del día para evitar el ruido dentro de las imágenes capturadas.
- Los caminos por donde pueda atravesar el robot dependen directamente de las capacidades del robot, en este caso los caminos deben ser planos.
- La aplicación no toma en consideración rutas o caminos complicados como por ejemplo caminos cruzados o caminos en Y.
- La aplicación es para dispositivos con sistema operativo Android, y la comunicación de la aplicación con el robot se la realizará mediante la tecnología Bluetooth.

3.2. EQUIPO

El equipo que se utilizó para la realización del proyecto fue:

- Laptop, para realizar la aplicación Android
- Smartphone, para ejecutar la aplicación y realizar las pruebas
- Robot, el cual realiza todas las actividades programadas

3.3. HERRAMIENTAS Y TÉCNICAS

Para la elaboración de la aplicación Android se utilizó un computador con sistema operativo Windows 7 con 6 GB de memoria RAM, procesador Intel Core i7 de 2.4 GHz, adicionalmente se instaló el siguiente software:

- Netbeans
- OpenCV (Para sistema operativo: Windows – Android)
- Weka (Para sistema operativo: Windows – Android)
- Eclipse Juno
- Android SDK

Para la implementación y pruebas de la aplicación se necesitará un dispositivo móvil que cumpla un mínimo de características:

- Sistema operativo Android v4.0
- 512 MB de memoria RAM
- Procesador 1.0 GHz
- Cámara 5 megapíxeles
- Incorporada tecnología Bluetooth

3.4. METODOLOGÍA

Para la realización del proyecto se utilizó la metodología de Prototipado para detallar el proceso interno y externo de la aplicación. En complemento con la metodología de Prototipado se utilizó la metodología de Diseño de hipermedia orientado a objetos OOHDM, la cual proporciona un proceso adecuado para el desarrollo de aplicaciones móviles y fue de gran apoyo durante la evolución del proyecto.

3.4.1. METODOLOGÍA DE PROTOTIPADO

Para detallar el proceso interno y externo de la aplicación Android se utilizó la metodología de Prototipado permitiendo al usuario visualizar el proceso de todas las actividades que va a realizar la aplicación para que el robot reconozca y se movilice por la ruta definida.

Se realizó un documento donde se incluye la especificación de requisitos del software para la aplicación Android que permitirá al robot seguir una ruta determinada por el usuario. Esta especificación se ha estructurado de acuerdo al estándar “IEEE Práctica Recomendada para Especificaciones de Requisitos Software ANSI/IEEE 830 1998”.

3.4.2. METODOLOGÍA DE DISEÑO HIPERMEDIA ORIENTADO A OBJETOS (OOHDM)

Para el desarrollo de la aplicación se utilizó la metodología OOHDM, que está basada en prototipos, desarrollo interactivo y desarrollo para facilitar el desarrollo de aplicaciones Web a través de cinco etapas donde se combinan notaciones gráficas UML (Lenguaje de Modelado Unificado) con otras propias de la metodología. (Céspedes & González, s.f)

A continuación se muestra en la Figura 8 las fases para la elaboración de la metodología OOHDM.

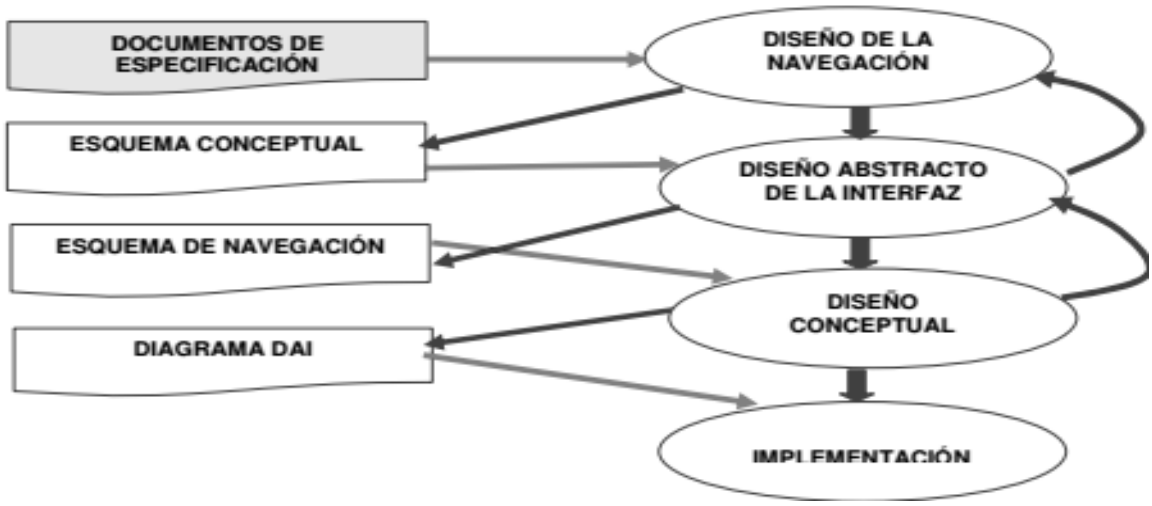

Figura 8. Fases de OOHDM (Céspedes & González, s.f)

Determinación de Requerimientos

El primero paso para el desarrollo de la aplicación fue la determinación de los requerimientos, en esta fase se definen los objetivos del proyecto, alcance, factibilidad de la aplicación y requerimientos funcionales y no funcionales.

Se identificó actores y tareas que cada uno de ellos realiza dentro del proceso además escenarios para cada tarea y actor. Con esta información se realizó casos de uso y el manual de requerimientos basado en la IEEE 830 especificado en la metodología de prototipado.

Diseño Conceptual

El diseño conceptual consta de esquemas conceptuales representando las clases y las relaciones entre las mismas utilizando el lenguaje unificado de modelado UML y diagramas de secuencia. Existen objetos que no se muestran en el modelo conceptual, pues son exclusivos para manejo de roles, pues no son parte conceptual del sistema o sus entidades.

Diseño de Navegación

En la metodología OOHDM la aplicación se ve a través de un sistema de navegación, en esta fase se realizó el diseño de la aplicación teniendo en cuenta las tareas que el usuario va a realizar sobre el sistema para lo cual se partió del diseño conceptual.

En OOHDM hay una serie de clases especiales predefinidas, que se conocen como clases navegacionales: nodos, enlaces y estructuras de acceso, que se organizan dentro de un contexto navegacional, siendo los nodos los contenedores básicos, los enlaces reflejan las relaciones de navegación por donde puede explorar el usuario y el contexto navegacional son los posibles caminos por donde el usuario pueda navegar, es el conjunto de nodos, enlaces y otros contextos navegacionales. En esta etapa mediante un esquema general se explicó la navegación de la aplicación.

Diseño de Interfaz Abstracta

En esta etapa se definieron las funcionalidades de la aplicación y la arquitectura del sistema. Esto significa definir la forma en la cual los objetos navegacionales pueden aparecer, cómo se activa la navegación y resto de funcionalidades de la aplicación.

Implementación

La última etapa es la implementación de la aplicación. Se implementó el diseño establecido en todos los pasos anteriores, el ambiente donde se ejecutará la aplicación y el tipo de arquitectura que se empleó para el desarrollo. Además se incluye las pruebas realizadas hasta obtener la aplicación final.

ANÁLISIS DE RESULTADOS

4. ANÁLISIS DE RESULTADOS

El proyecto tuvo como objetivo diseñar e implementar una aplicación Android que pueda detectar un tipo de marcaje definido en el camino, y que según el entrenamiento del robot siga esa ruta. Para la planificación y ejecución del proyecto se utilizó la metodología OOHDM combinando con la metodología de prototipado para un mejor entendimiento del usuario.

A continuación se detalla el proceso del desarrollo del proyecto con cada una de las fases descritas en la metodología OOHDM.

4.1. ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE

4.1.1. INTRODUCCIÓN

La determinación de requerimientos es una de las fases más importantes para el desarrollo de una aplicación ya que contiene las necesidades del usuario. Como se indicó en la metodología, se elaboró el documento de requerimientos siguiendo el estándar IEEE 830.

“La IEEE indica que un buen documento de requisitos debe contemplar toda la información presentada en dicho estándar, aunque propone una organización de la información, no exige estrictamente el formado de dicha información” (Specifying Security Requirements Improvement for IEEE Standard 830, 2007)

4.1.1.1. Propósito

La especificación de requerimientos se hizo con el fin de definir de manera clara y precisa todos los requerimientos de la aplicación, para el funcionamiento de los mismos.

4.1.1.2. Ámbito del Sistema

El proyecto se realizó con el fin de ayudar a posteriores investigaciones que estén relacionadas con aplicaciones móviles que utilicen visión computarizada y además se incorpore inteligencia artificial. El sistema se lo realizó para Android, el sistema operativo más utilizado en teléfonos celulares en el mundo (Univision, 2013),

utilizando librerías OpenCV y para el entrenamiento del robot mediante aprendizaje automático utilizando Weka.

4.1.1.3. Definiciones y Acrónimos

En la Tabla 1,2 y 3, se indican las definiciones, acrónimos y abreviaturas de palabras que se van a utilizar en el documento respectivamente.

Definiciones

Tabla 1. Definiciones

Robot móvil	Máquina automática que es capaz de movilizarse en cualquier ambiente dado.
Aplicación móvil (app)	Aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles.
Patrón de Diseño	Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software
Iracer	Nombre del robot con el cual se va a conectar la aplicación.
Assets	Carpeta propia de Android donde se almacenan las imágenes de prueba y los archivos necesarios en el proyecto.
Arff	Archivos propios de Weka, utilizado para la creación y lectura del modelo.
Bitmap	Imagen matricial, se utiliza para procesar la imagen.
Cvs	Archivos para representar datos en forma de tabla, utilizados para la creación del modelo.

Fuente: (WordReference.com, 2015), (Real Academia Española, 2015)

Acrónimos

Tabla 2. Acrónimos

ERS	Especificación de Requisitos de Software
IA	Inteligencia artificial
GUI	Interfaz gráfica de usuario
API	Interfaz de aplicación de programación
SDK	Kit de desarrollo de software
UML	Lenguaje de modelado unificado
CU	Caso de uso
ARFF	Formato de archivo atributo-relación
CVS	Valores separados por coma

Abreviaturas

Tabla 3. Abreviaturas

APP	Aplicación
S.O	Sistema operativo

4.1.1.4. Referencias

En la Tabla 4 se muestra las referencias para la realización del documento de requerimientos.

Tabla 4 Referencias

Referencia	Titulo	Ruta	Fecha	Autor
IEEE	Especificación de Requisitos según el estándar de IEEE 830.	kovachi.sel.inf.uc3m.es/@api/deki/files/56/=formato_ieee830.doc	2001	Raúl Monferrer Agut

4.1.1.5. Visión general del ERS

El presente ERS está conformado por tres secciones.

-Introducción. Se proporciona una visión general del ERS.

-Descripción general. Con el fin de conocer las principales funciones que debe realizar, los datos asociados y los factores, restricciones, supuestos y dependencias que afectan al desarrollo, sin entrar en excesivos detalles.

-Requisitos. La definición detallada de los requisitos que debe satisfacer la aplicación.

4.1.2. DESCRIPCIÓN GENERAL

En esta sección se presenta una descripción a alto nivel del sistema que indica las funciones que el sistema debe realizar, la información utilizada, las restricciones y otros factores que afecten al desarrollo del mismo.

4.1.2.1. Perspectiva del Producto

- El sistema se desarrollará para la plataforma Android que se ejecutará desde la versión 4.0
- La aplicación tendrá una interfaz gráfica amigable para el usuario y de fácil acceso.
- El software se conecta mediante bluetooth con el robot.
- La aplicación no almacenará ninguna información.

4.1.2.2. Características de los usuarios

Los usuarios que utilizarán la aplicación deberán estar familiarizados con el manejo de aplicaciones en dispositivos móviles, además deberá contar con un robot que tenga conexión bluetooth. El sistema desarrollado es de gran ayuda para usuarios interesados en la realización de nuevas aplicaciones que contengan procesamiento de imágenes e inteligencia artificial dentro de la plataforma Android.

4.1.2.3. Restricciones

- La aplicación estará disponible solo para dispositivos móviles con sistema operativo Android v4.0 en adelante.

- El software no almacenará ninguna información.
- La aplicación utiliza la cámara propia de OpenCV, por lo que es necesario la utilización de la misma.
- La ruta del robot es una línea continua negra.
- La ruta no tiene cruces, obstáculos ni tampoco varios caminos alternos.

4.1.2.4. Suposiciones y Dependencias

- El software se implementará con lenguaje de programación java en la plataforma Eclipse Juno y la aplicación podrá ser ejecutada en cualquier dispositivo Android desde la versión 4.0.
- El dispositivo móvil debe estar conectado con el robot mediante Bluetooth y debe tener instalada la librería OpenCV – Manager.
- La ruta deberá tener las proporciones adecuadas según el tamaño del robot.

4.1.3. REQUISITOS

En este apartado se presentan los requisitos: funcionales, no funcionales, de rendimiento y futuros que deberá satisfacer la aplicación. Todos los requisitos expuestos a continuación se han tomado en cuenta para su desarrollo.

4.1.3.1. Funciones del Sistema

La aplicación móvil tiene como función el seguimiento de marcaje de ruta del robot que se lo realizará mediante el reconocimiento de patrones a partir de cámara de video en tiempo real utilizando procesamiento de imágenes para tener resultados óptimos en diferentes ambientes.

4.1.3.2. Requisitos funcionales

Los requerimientos funcionales describen las funciones que realiza la aplicación, sus entradas y salidas, los cuales se pueden representar de diferentes formas, en este caso se utilizó casos de uso, “una técnica que permite modelar las funciones de un sistema en términos de eventos, de quién inicia los eventos y de cómo responde el sistema a estos eventos.” (Alarcón, 2010)

A continuación en la Figura 9 se indica la relación que existe entre los actores y la aplicación.

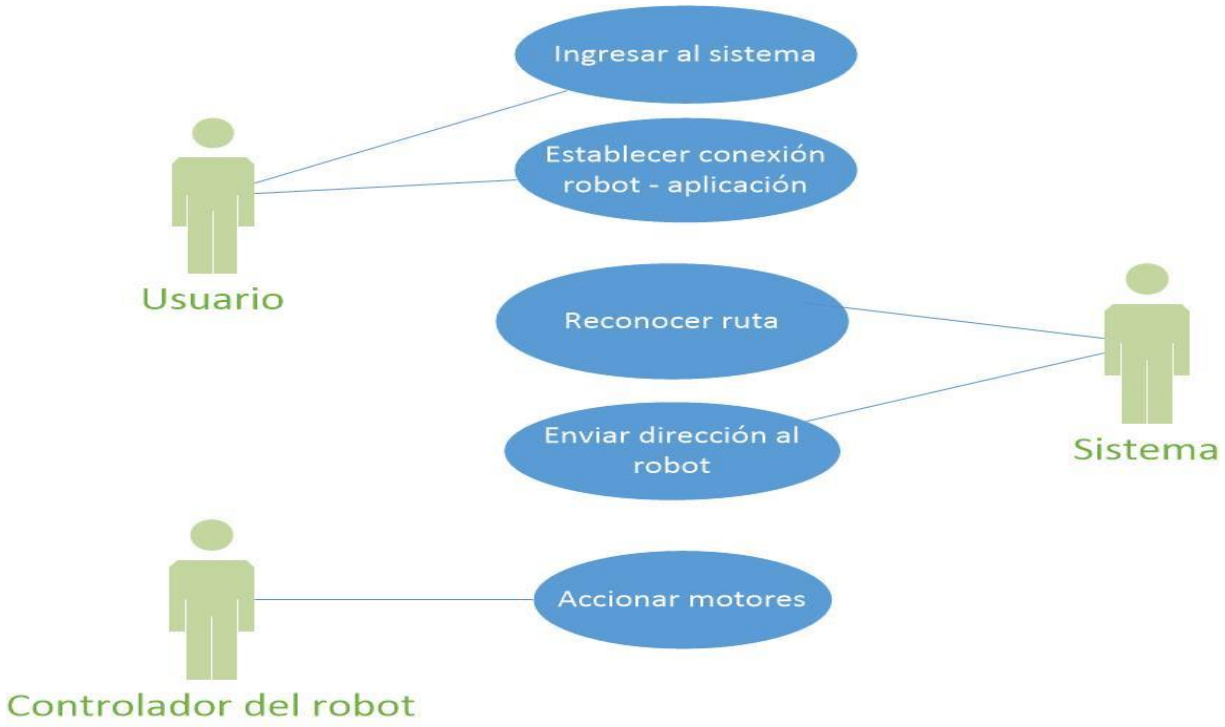

Figura 9. CU General del Sistema

El sistema es para un usuario, y su interfaz gráfica es muy simple ya que consta de un único botón que conecta al robot con la aplicación y enseguida se conecta empieza a seguir la ruta. En la Tabla 5 se presenta el detalle de actores que participan en el sistema.

Tabla 5. Detalle de actores del sistema

DETALLE DE ACTORES	
Actor	**Descripción**
Usuario	Persona que va a interactuar con la aplicación y el robot.
Sistema	Aplicación Android instalada en el dispositivo móvil
Robot	Ejecutor del seguimiento de la ruta

4.1.3.3. Caso de uso Ingresar al sistema

En la Figura 10 se indica el proceso de ingreso al sistema, luego el usuario encuentra un botón Conectar robot para realizar la conexión mediante bluetooth.

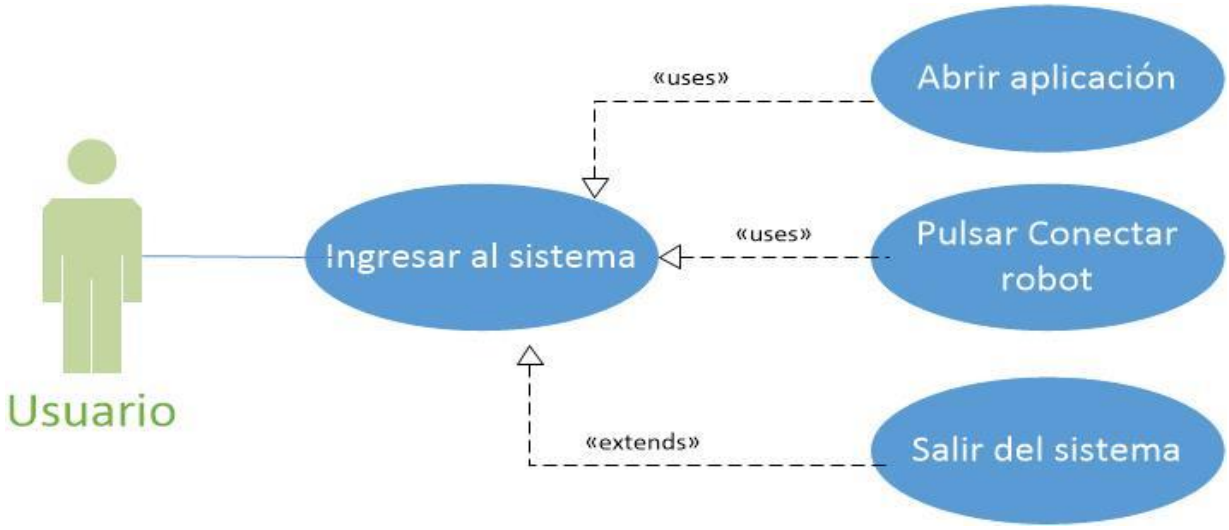

Figura 10. CU Ingresar al Sistema

En la Tabla 6 se indica la especificación del caso de uso Ingresar al sistema.

Tabla 6. Especificación CU Ingresar al sistema

Nombre del caso de uso	**Ingresar el sistema**	
Descripción	Se abre la aplicación y se procede a realizar la conexión bluetooth.	
Actor	Usuario	
Flujo Básico		
Paso	**Actor**	**Sistema**

1	Abrir aplicación	Se presenta la pantalla principal de la aplicación
2	Pulsa botón Conectar Iracer	Ingresa a la pantalla para realizar la conexión bluetooth
Flujo alterno		
2.1	Salir	Cierra la aplicación
Pre-condición	_____	
Post-condición	_____	

4.1.3.4. Caso de uso Conectar robot - aplicación

En la Figura 11, se indica el proceso que implica conectar el robot con la aplicación Android.

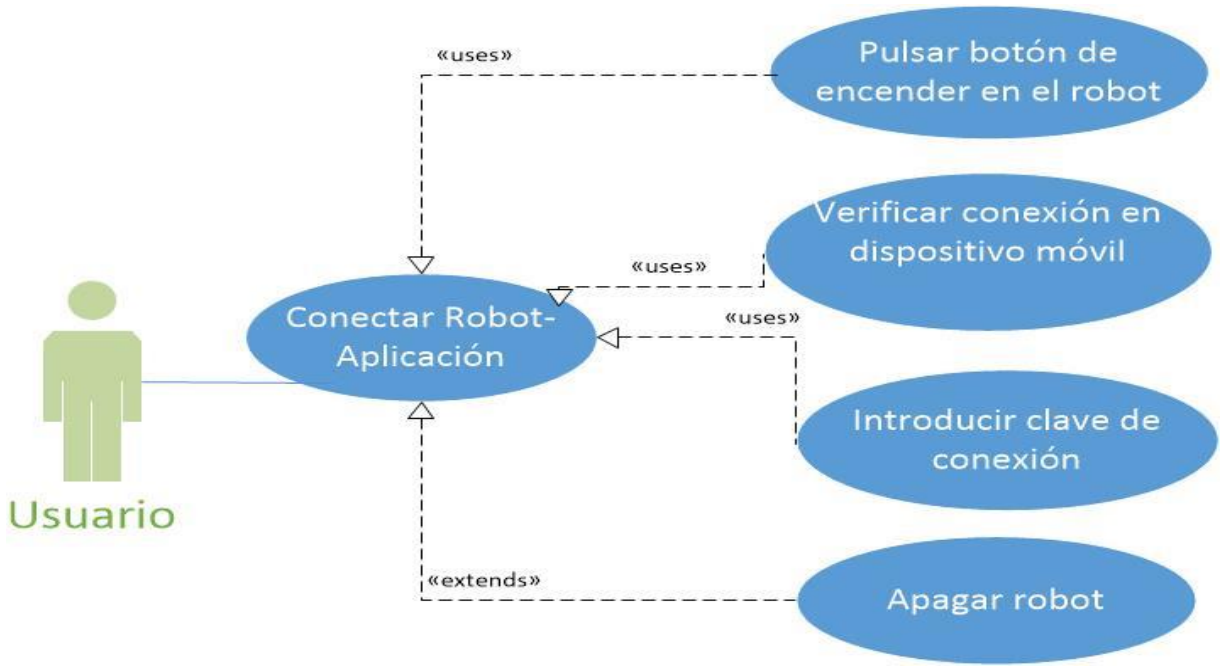

Figura 11. CU Encender Robot

En la Tabla 7 se presenta la especificación del caso de uso Conectar robot – aplicación.

Tabla 7. Especificación CU Conectar robot - aplicación

Nombre del caso de uso	**Conectar robot - aplicación**	
Descripción	Se verifica la conexión entre el robot y la aplicación.	
Actor	Usuario	
Flujo Básico		
Paso	**Actor**	**Sistema**
1	Pulsar botón de encender en el robot	Presenta un mensaje de conexiones disponibles
2	Verificar conexión en dispositivo móvil	Presenta un mensaje de si encontró o no el robot
3	Introducir la clave de conexión bluetooth	Pide ingreso de clave
Flujo alterno		
2.1	Apagar robot	Presenta un mensaje de que no encuentra el robot para la conexión
Pre Condición	_____	
Post Condición	_____	

4.1.3.5. Casos de uso Reconocer ruta

En la Figura 12, se indica el proceso de reconocer la ruta que lo realiza el sistema.

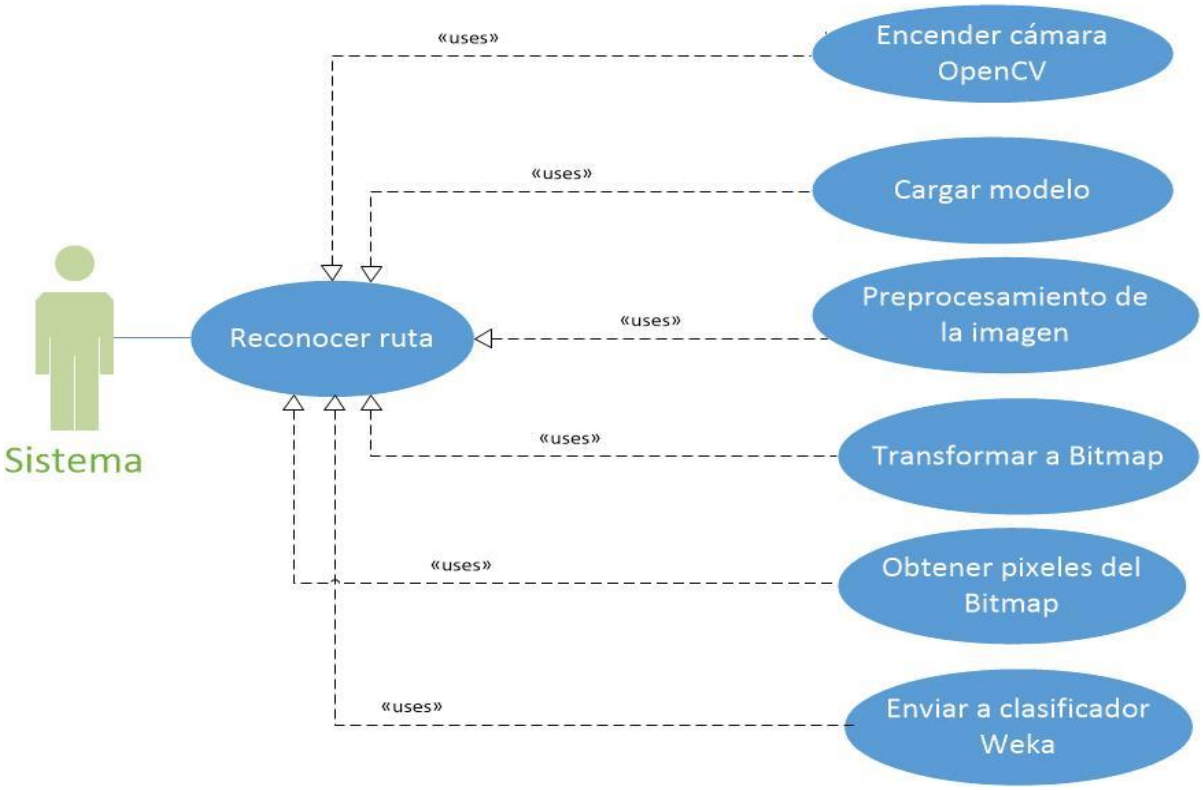

Figura 12. CU Reconocer ruta

En la Tabla 8 se indica la especificación del caso de uso reconocer ruta.

Tabla 8. Especificación CU Reconocer ruta

Nombre del caso de uso	**Reconocer ruta**
Descripción	Se realiza el tratamiento de la imagen necesario hasta obtener el Bitmap que se envía para la clasificación de las nuevas imágenes
Actor	Sistema

Flujo Básico		
Paso	**Sistema**	**Descripción**
1	Encender cámara OpenCV	Utilizando la librería OpenCV se enciende la cámara
2	Cargar modelo	Se carga el modelo de Weka una sola vez, cuando la cámara se enciende
3	Preprocesamiento de la imagen	Se realiza el tratamiento de la imagen
4	Transformar a Bitmap	Se trasforma la imagen a tipo Bitmap para poder obtener los pixeles
5	Obtener pixeles del Bitmap	Se extrae los pixeles de la imagen
6	Enviar a clasificador Weka	Se clasifica la nueva instancia utilizando la librería Weka
Pre Condición	———	
Post Condición	———	

4.1.3.6. Caso de uso Enviar dirección al robot

En la Figura 13, se muestra el proceso para enviar la dirección al robot

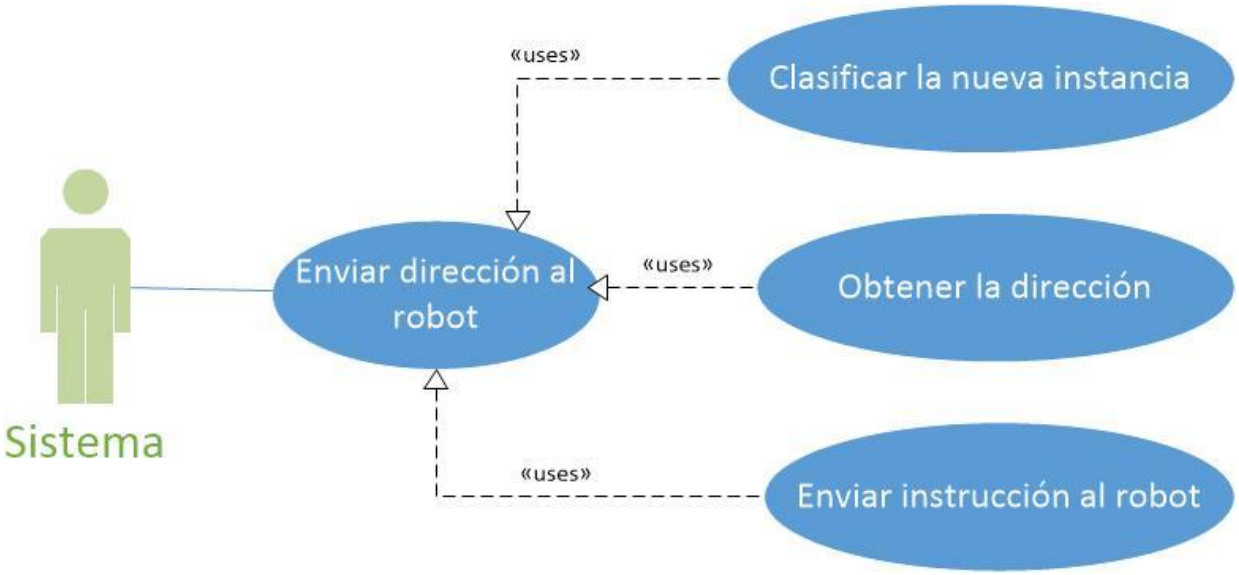

Figura 13. CU Enviar dirección

En la Tabla 9 se detalla la especificación del caso de uso enviar dirección.

Tabla 9. Especificación CU Enviar dirección

Nombre del caso de uso	**Enviar dirección**	
Descripción	Se clasifica la nueva imagen utilizando la librería Weka, se obtiene la dirección y se envía como una instrucción al robot.	
Actor	Sistema	
Flujo Básico		
Paso	**Sistema**	**Descripción**
1	Clasificar la nueva instancia	Se clasifica la nueva instancia.

2	Obtener la dirección	Se obtiene la dirección y cada dirección se la transforma a hexadecimal
3	Enviar instrucción al robot	Se envía la dirección y la velocidad al robot
Pre Condición	———	
Post Condición	———	

4.1.3.7. Caso de uso Accionar actuadores

En la Figura 14 se indica el proceso que realiza el controlador del robot para accionar los actuadores mediante las instrucciones recibidas del sistema Android.

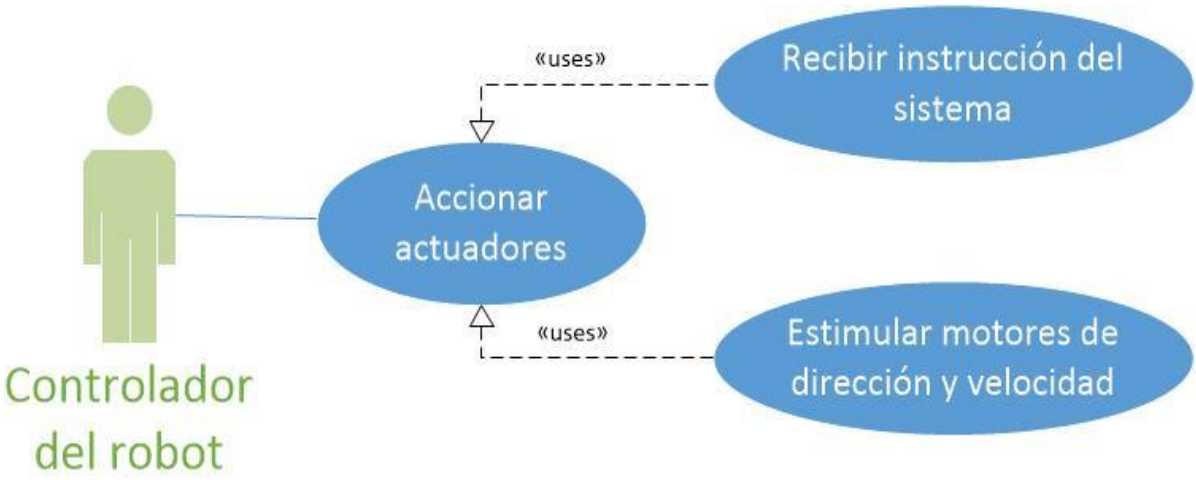

Figura 14. CU Accionar actuadores

Tabla 10. Especificación CU Accionar actuadores

Nombre del caso de uso	**Accionar actuadores**	
Descripción	Se recibe la instrucción del sistema para accionar los actuadores del robot que son sensores, motores de dirección y velocidad, entre otros componentes.	
Actor	Controlador del robot	
Flujo Básico		
Paso	**Sistema**	**Descripción**
1	Recibir instrucción	Mediante conexión bluetooth el controlador del robot recepta instrucciones de dirección y velocidad
2	Estimular motores de dirección y velocidad	Con la instrucción recibida acciona a todos los actuadores para realizar el seguimiento de la ruta
Pre Condición	———	
Post Condición	———	

4.1.4. REQUISITOS NO FUNCIONALES

4.1.4.1. Requisitos de hardware

La aplicación se puede ejecutar en dispositivos móviles de aproximadamente 4.3 pulgadas debido al soporte que tiene el robot para ubicar el dispositivo.

4.1.4.2. Requisitos de software

La aplicación permite ser ejecutada en dispositivos móviles con sistema operativo Android versión 4 como mínimo, que tengan previamente instalada la librería OpenCV – Manager 2.4.X disponible en google Play store.

4.1.4.3. Requisitos de comunicación

La aplicación es para un robot, por lo tanto se necesita un medio para comunicar el robot y la aplicación que es mediante tecnología Bluetooth.

4.1.4.4. Atributos del Sistema

El sistema debe cumplir con los siguientes atributos:

Fiabilidad. La aplicación presenta las validaciones de acuerdo a las funciones que debe realizar.

Mantenible. La aplicación puede ser modificada de acuerdo a los nuevos requerimientos, e incluso aumentar varias funcionalidades en nuevos entornos y rutas más complejas.

Eficiencia. El tiempo que tarda la aplicación en obtener la dirección y enviarla en forma de comandos al robot debe ser mínimo de una dirección por segundo.

Portabilidad. La aplicación podrá ser ejecutada en cualquier dispositivo con bluetooth y con una versión de Android superior a la 4.0

Reusable. El código de la aplicación se puede utilizar para el desarrollo de nuevas aplicaciones a fines con las librerías OpenCV y Weka, y cualquier aplicación para S.O Android.

Usabilidad. La interfaz del usuario es amigable y fácil de ejecutar.

4.1.4.5. Requisitos de Interfaces

- La interfaz de usuario es sencilla, pues consta de un único botón que conecta al robot con la aplicación y ya empieza a seguir el patrón siempre que se encienda antes al robot.
- Está orientado a pantallas táctiles, se ajusta a diversos tamaños de pantallas, y orientación.

4.1.4.6. Requisitos de Rendimiento

- Es necesario el S.O Android igual o superior a la versión 4.0, para poder ejecutar en cualquier dispositivo móvil.
- Debido a que el tiempo de respuesta de la cámara integrada en el dispositivo móvil era lento, y causó varios inconvenientes se optó por utiliza la cámara propia de la librería OpenCV.
- El tiempo en clasificar una nueva instancia, es decir el que indica la dirección por donde va a ir el robot, es aproximadamente 3 direcciones por segundo, se debe ajustar tomando en cuenta el proceso en cada nueva imagen que realiza la aplicación, en este caso se tuvo que redimensionar a 28x28 pixeles aun cuando el resultado más óptimo inicialmente fue de 50x50 pixeles, debido al tiempo de respuesta.
- Se debe verificar cuántos fotogramas (imágenes) por segundo se obtienen a través de la cámara, que debe ser similar al tiempo de respuesta del robot, para que una vez que el robot reciba la dirección avance siguiendo la ruta especificada.

4.1.4.7. Restricciones de Desarrollo

- Incompatibilidad de las herramientas para generación del modelo clasificador de Weka en la plataforma Android y la utilización de las librerías OpenCV en sistemas operativos Windows y Android que finalmente se pudieron solucionar adaptando todo en función de OpenCV.

4.1.4.8. Requisitos futuros

- Generar nuevos modelos en Weka para que el robot sea capaz de movilizarse en escenarios complejos, como por ejemplo rutas con formas irregulares y se los pueda implementar para cualquier robot incluso los robots aéreos que están siendo de mucha utilidad actualmente.

4.1.5. FACTIBILIDAD DEL SISTEMA

4.1.5.1. Factibilidad técnica

Para el desarrollo del proyecto se tuvo los recursos informáticos necesarios tanto en hardware como software. En la Tabla 11 se detallan los aspectos más importantes.

Tabla 11. Factibilidad Técnica

Descripción	Si	No
Computador con sistema operativo Windows 7 core i7	X	
Robot con tecnología Bluetooth	X	
Manejo Eclipse Juno SDK Android	X	
Manejo de Java IDE NetBeans	X	
Manejo de librerías OpenCV		X
Manejo de librería Weka		X

4.1.5.2. Factibilidad Económica

La inversión para desarrollar el proyecto es mínima, ya que dispone de la mayoría de los recursos necesarios. En la Tabla 12 se lista el presupuesto general.

Tabla 12. Presupuesto del Proyecto

Presupuesto		
Rubro	**US$**	**Fuente Financiamiento**
Dispositivo móvil	300	Propio
Papel	50	Propio
Cd´s	5	Propio
Internet	90	Propio
Robot	150	Laboratorio
Total:	**$595**	

4.1.6. PROTOTIPADO DIGITAL

Para explicar el proceso interno de la Aplicación Android y el proceso externo es decir lo que realiza el robot, se utilizó la herramienta de Office Power Point utilizando una serie de imágenes que facilitan el entendimiento para el usuario y se presentaron antes del desarrollo de la aplicación.

4.1.6.1. Proceso

La Aplicación tiene un botón Conectar Iracer como se indica en la Figura 15, que conecta la aplicación Android con el robot mediante tecnología Bluetooth y automáticamente enciende la cámara del dispositivo móvil que recibe varias imágenes por segundo para enviar al robot la dirección por donde éste debe movilizarse.

Figura 15. Prototipado Interfaz Gráfica

Para que el robot sea capaz de tomar decisiones en caso de nuevas rutas sigue el siguiente proceso:

- Captura de la imagen mediante la cámara propia de la librería OpenCV como indica la Figura 16.

Figura 16. Prototipado Captura de Imagen

- Tratamiento de la imagen con librerías OpenCV, donde se realizó las debidas modificaciones para que el patrón que va a seguir el robot no se pierda en todo el escenario que presenta la imagen completa. (Ver Figura 17)

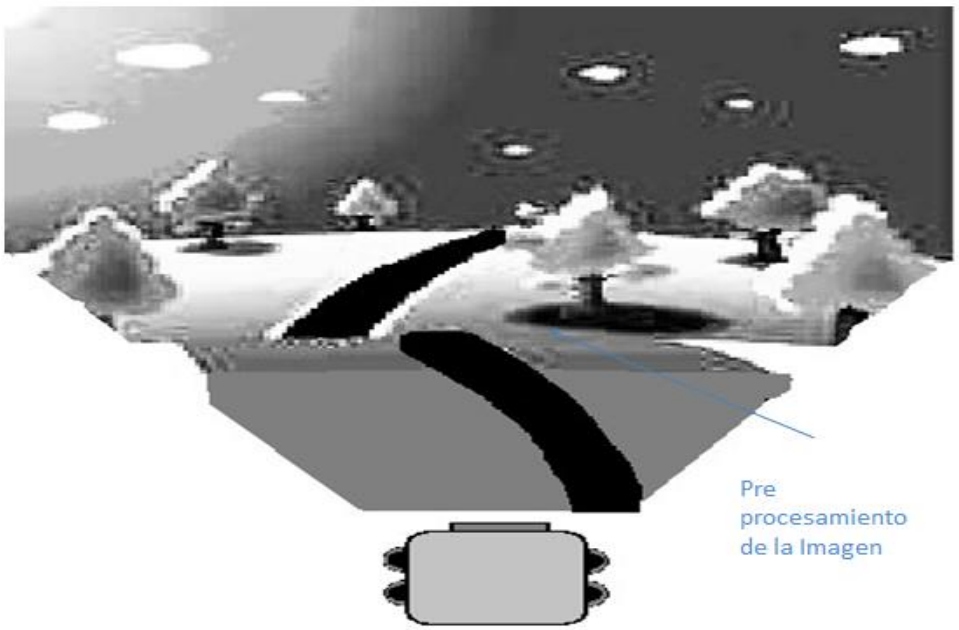

Figura 17. Prototipado Pre procesamiento de imagen

- Se cortó la imagen debido a la cantidad de ruido en la parte superior como se muestra en la Figura 18.

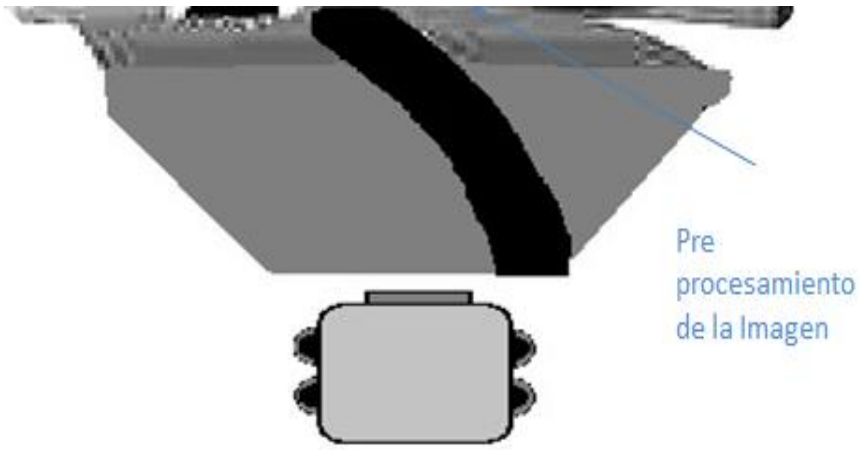

Figura 18. Prototipado Procesamiento de imagen

- Redimensionar la imagen, facilita el trabajo ya que no toma todos los pixeles de la imagen. (Ver Figura 19)

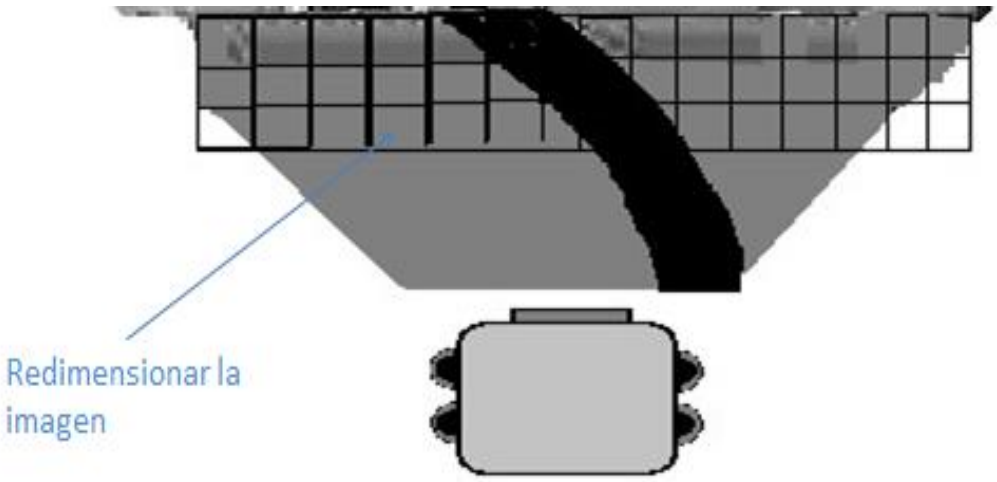

Figura 19. Prototipado Redimensionar imagen

- Extracción de características, para lo cual se utiliza Weka, para la detección de la línea. Al final se enciende el robot y luego el dispositivo móvil como indica la Figura 20.

Figura 20. Prototipado Extracción de características

4.2. DISEÑO CONCEPTUAL

El diseño conceptual es la segunda fase de la metodología OOHDM, equivale a los modelos de la aplicación, los diagramas de secuencia, diagramas de clase, diseño navegacional, diseño de interfaz abstracta y arquitectura de la aplicación.

4.2.1. DIAGRAMAS DE SECUENCIA

Estos diagramas se hicieron de acuerdo a cada caso de uso, con el fin de modelar la interacción entre objetos de la aplicación. Se los realizó en la herramienta Microsoft Visio 2010. En la Figura 21 se presenta el diagrama secuencial del caso de uso Ingresar al sistema.

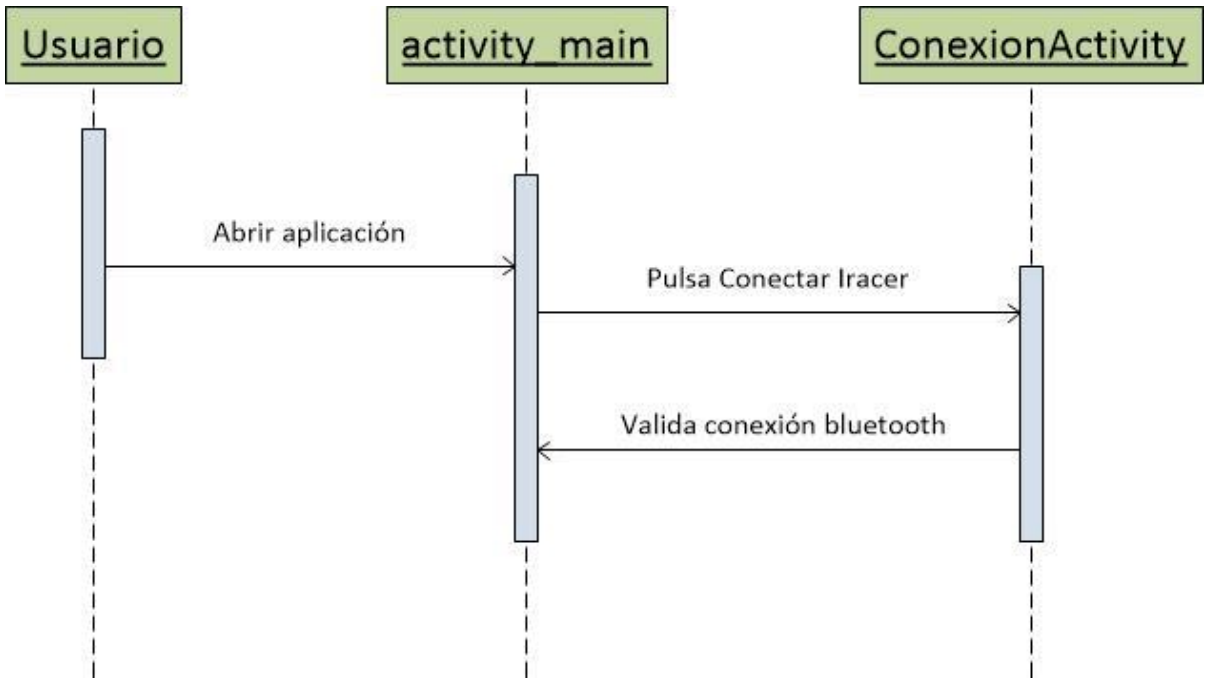

Figura 21. Diagrama secuencial Ingresar al sistema

En la Figura 22 se presenta el diagrama secuencial del caso de uso Conectar robot – aplicación, cuando el usuario conecta al robot con la aplicación.

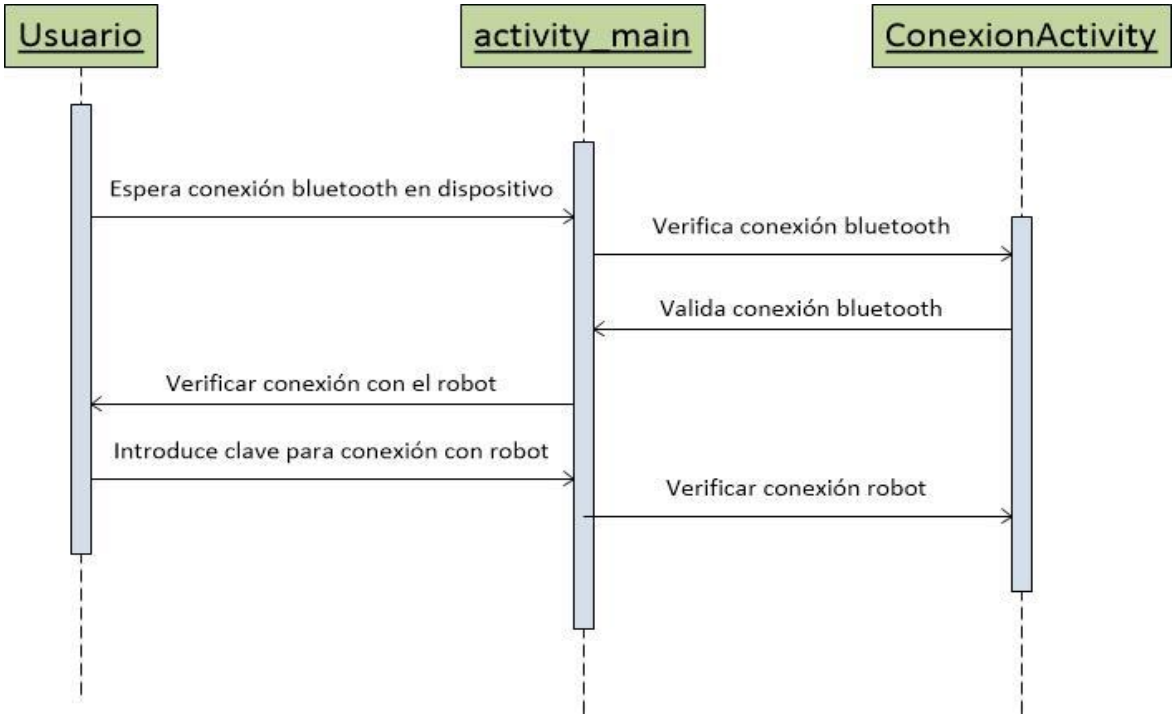

Figura 22. Diagrama secuencial Conectar robot- aplicación

En la Figura 23 se indica el diagrama secuencial del caso de uso reconocer ruta.

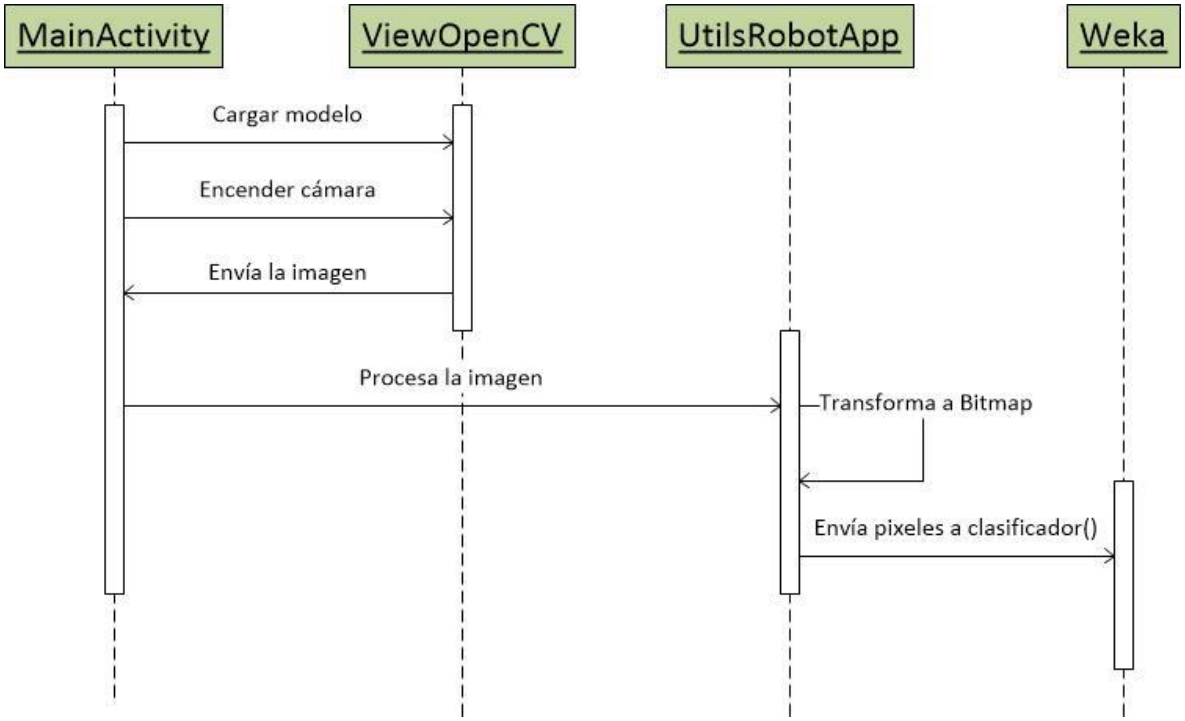

Figura 23. Diagrama secuencial Reconocer ruta

En la Figura 24 se presenta el diagrama secuencial del caso de uso enviar dirección al robot.

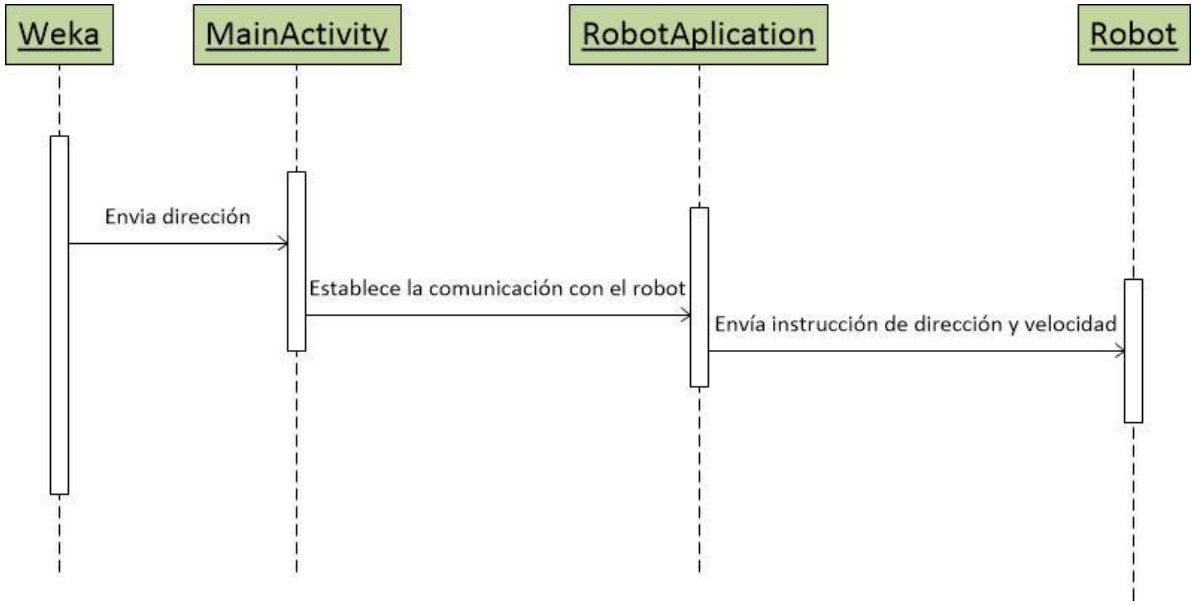

Figura 24. Diagrama secuencial Enviar dirección al robot

4.2.2. DIAGRAMAS DE CLASES

En la Figura 25 se presenta el diagrama de clases generado automáticamente con la herramienta ObjectAid, un plugin para la plataforma Eclipse, que permite mostrar el código Java (incluido librerías) como diagrama de clases UML y como diagrama de secuencia. (Gracia, 2012)

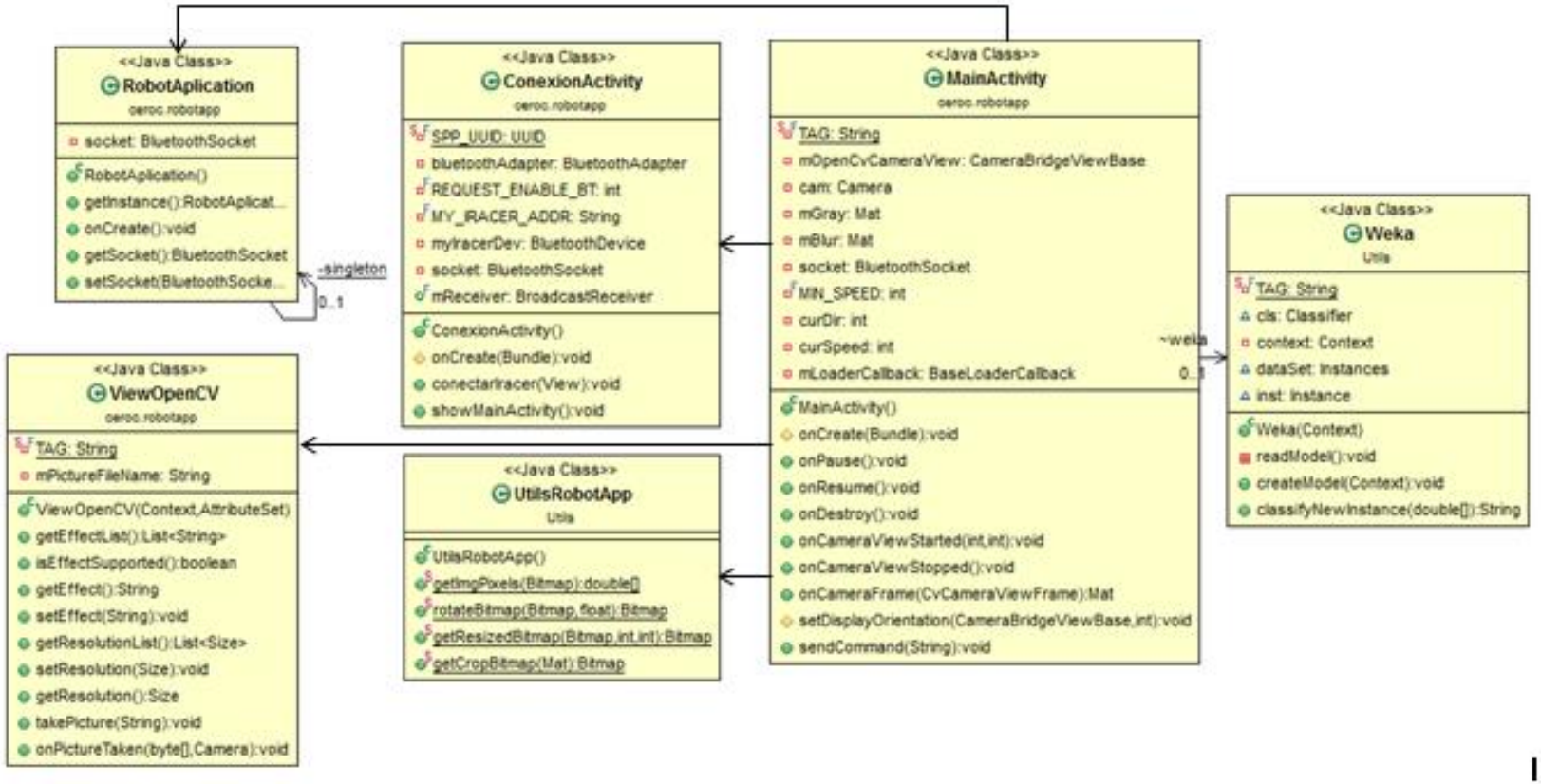

Figura 25. Diagrama de clases de la aplicación

4.2.3. DISEÑO NAVEGACIONAL

La aplicación es dirigida para un robot, utilizando inteligencia artificial por lo tanto no se necesitan mucha interacción con el usuario, aun así se maneja una interfaz para establecer la conexión con el robot.

4.2.3.1. Esquema navegacional

La aplicación tiene una interfaz muy sencilla, pues una vez que se enciende el robot el usuario ingresa al sistema y da clic en el botón conectar Iracer, luego ubica el teléfono en la base del robot y automáticamente empieza a funcionar.

Objetos Navegacionales

En la Figura 26 se presenta los objetos navegacionales, que son las interfaces gráficas de la aplicación.

- Pantalla inicial
- Conectar Iracer

Contextos Navegacionales

- Inicia la aplicación

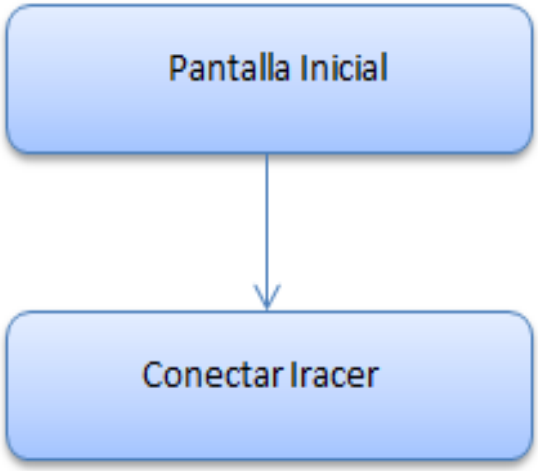

Figura 26. Contextos navegacionales

4.2.4. DISEÑO DE INTERFAZ ABSTRACTA

Los modelos abstractos especifican la organización y el comportamiento de la interfaz (Pressman, 2006), a continuación se detallan las vistas abstractas de la aplicación.

En las figuras 27 y 28 se indican las vistas de datos abstractas.

Nodo Pantalla Inicial

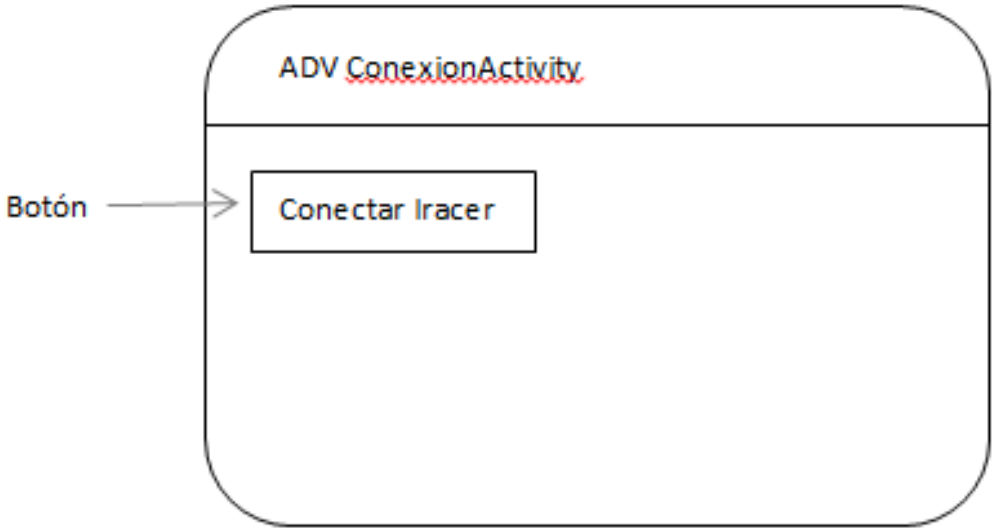

Figura 27. Nodo pantalla inicial

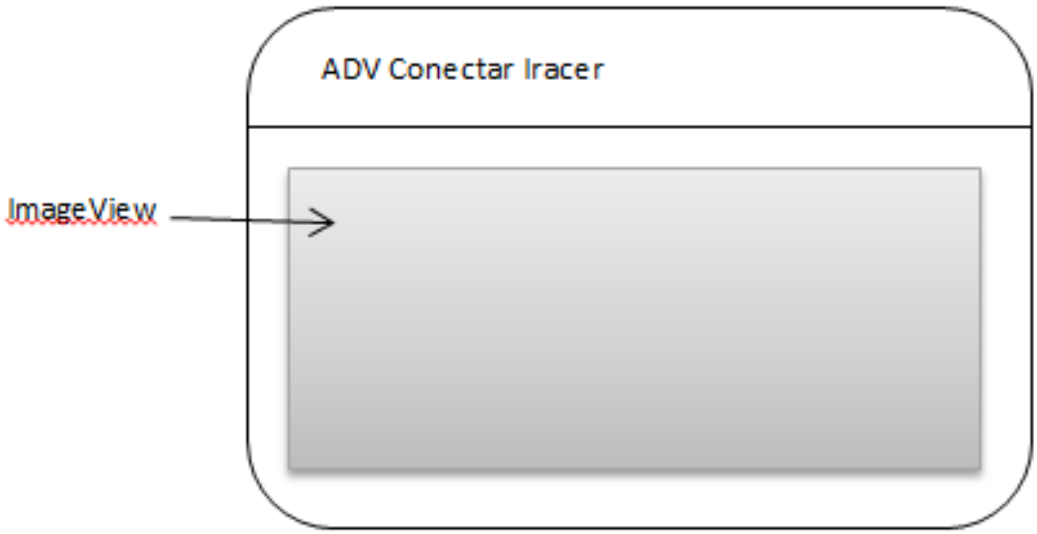

Figura 28. Nodo Conectar Robot

4.3. IMPLEMENTACIÓN

En la fase de la implementación se indica cómo se desarrolló la aplicación, las herramientas que se utilizó, y las pruebas que se realizaron hasta obtener el producto final.

Se realizaron dos sistemas en diferentes plataformas, uno para la construcción del modelo y para el entrenamiento del robot utilizando las librerías weka para Windows, y la aplicación final del robot utilizando S.O Android.

4.3.1. SISTEMA: MODELO WEKA

El sistema modelo weka, es un sistema que ayuda en la construcción y generación de nuevos modelos con un entrenamiento que se da a través de todas las imágenes que se cargan y el tratamiento que se les da para encontrar el modelo con mayor porcentaje de exactitud.

4.3.1.1. Herramientas

- Java - Netbeans IDE 7.4
- OpenCV (para Windows)
- Weka (para Windows)

4.3.1.2. Instalación

NETBEANS

- Ingresar en www.netbeans.org
- Descargar Netbeans a partir de la versión 7.4 para Windows (ver Figura 29)

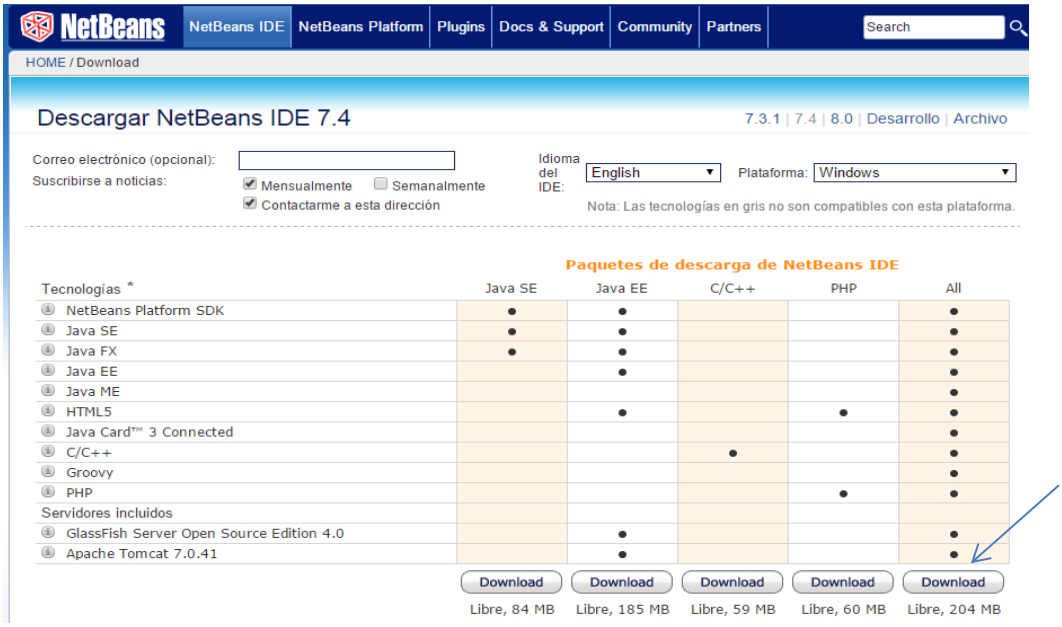

Figura 29. Descarga Netbeans IDE 7.4

- Una vez se finaliza la instalación se sigue con el desarrollo del sistema, para lo cual es necesario instalar las librerías OpenCV y Weka.

OPENCV – WINDOWS

- Ingresar en www.opencv.org
- Descargar OpenCV para Windows, como se indica en la Figura 30.

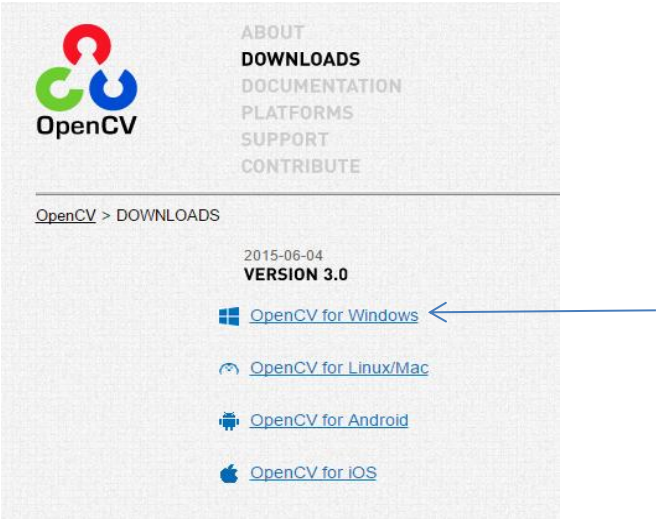

Figura 30. Descarga de OpenCV para Windows

- En las propiedades del proyecto se deben cargar las librerías, como se muestra en la Figura 31.

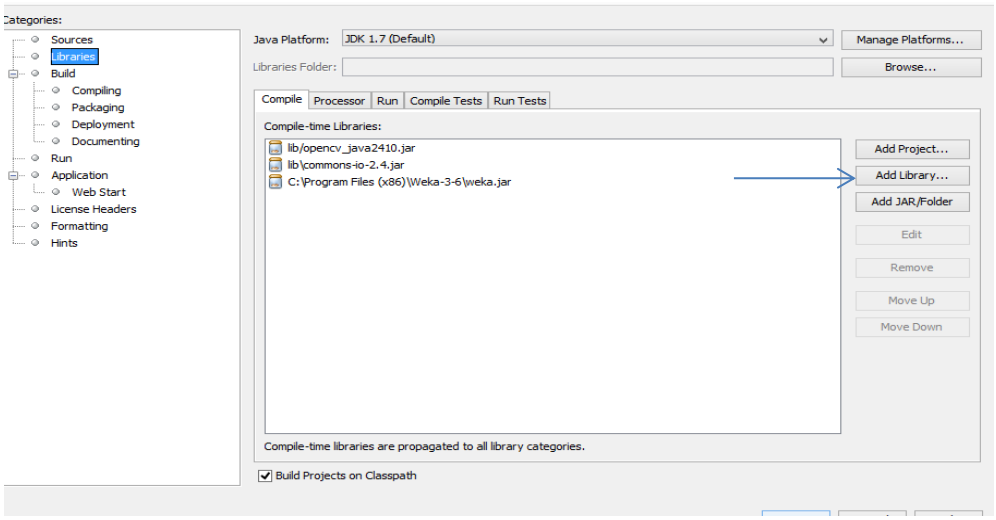

Figura 31. Configuración de librería OpenCV

WEKA – WINDOWS

- Ingresar en www.cs.waikato.ac.nz
- Descargar Weka para Windows, como se indica en la Figura 32.

Figura 32. Descarga librería Weka para Windows

- En las propiedades del proyecto se deben cargar la librería weka, de la misma manera en que se cargó la librería OpenCV.

4.3.1.3. Diseño y código fuente

Para la obtención del modelo se realizó un programa utilizando la plataforma Netbeans, aquí se selecciona la carpeta que contiene todas las imágenes que entrenarán al robot organizadas por dirección como muestra la Figura 33.

El programa tiene un botón que selecciona la carpeta con las imágenes previamente clasificadas por el usuario con la dirección. Una vez que se ejecuta el programa el usuario selecciona primero la dirección ya sea izquierda, centro o derecha como indica la Figura 34. Luego busca la carpeta correspondiente, tomando en cuenta que las imágenes deben estar en formato JPEG.

Figura 33. Imágenes clasificadas por dirección

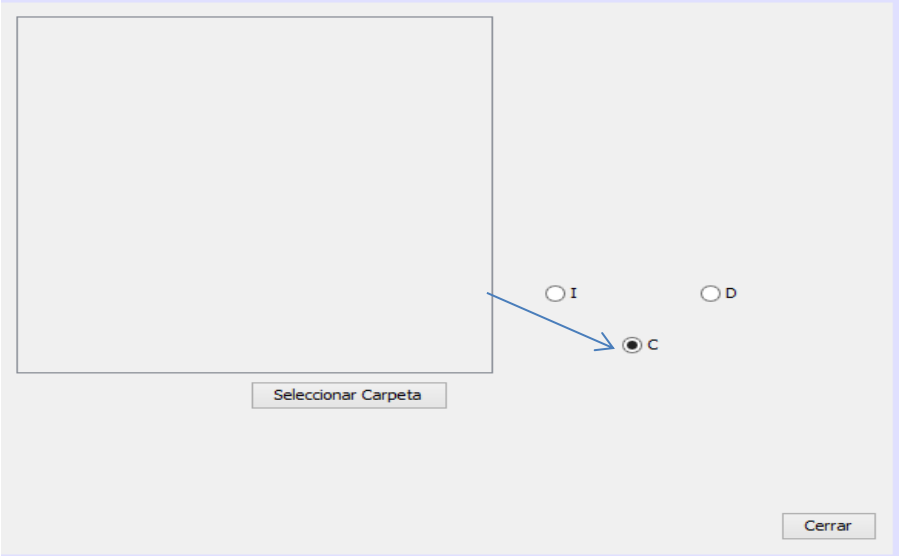

Figura 34. Interfaz gráfica Netbeans

El proyecto consta de varias clases, agrupadas en dos paquetes como se indica en la Figura 35. El primer paquete es cargarimagen donde se tiene:

- **CargarImagen.java** donde está la interfaz gráfica
- **ImageLibrary.java** que contiene métodos para obtener los pixeles de la imagen
- **ImageObject.java** que contiene métodos para procesar la imagen y transformarla en un arreglo de imágenes con distintos tamaños
- **Modelo.java** es la clase que utiliza la librería Weka para la creación del modelo y la clasificación de nuevas instancias.

El otro paquete es utilities, que contiene los métodos de procesamiento de imágenes que utilizan las librerías OpenCV.

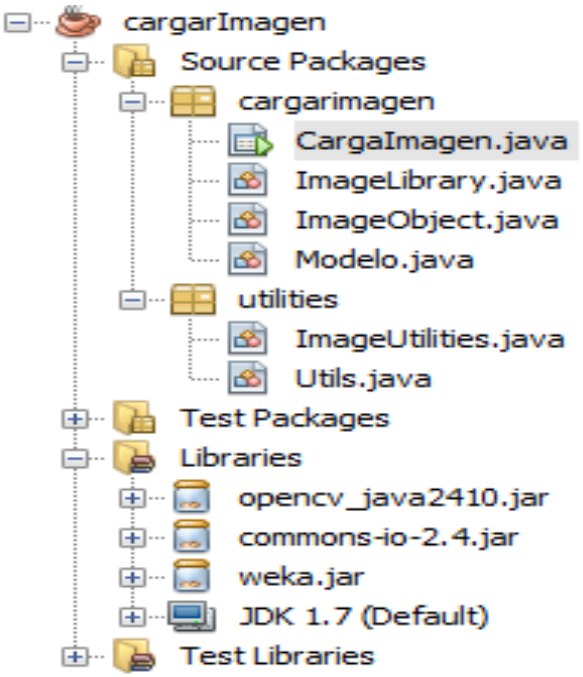

Figura 35. Vista navegacional del proyecto cargarImagen de Netbeans

A continuación se muestra el código fuente de los métodos y botones principales para la creación del modelo, con el cual se realizaron varias pruebas hasta obtener el apropiado.

Botón Cargar Imágenes

En la Figura 36 se indica el método principal que carga la carpeta con las imágenes y llama a los métodos para procesarlas.

```
private void btnSeleccionarFolderActionPerformed(java.awt.event.ActionEvent evt) {
    //Creamos un nuevo cuadro de diálogo para seleccionar imagen
    JFileChooser selector = new JFileChooser("C:\\Users\\Natasha Garcia\\Desktop\\ImagenesTest");
    //Le damos un título
    selector.setDialogTitle("Seleccione una carpeta");
    //Filtramos los tipos de archivos
    selector.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    //Abrimos el cuadro de diálog
    int flag = selector.showOpenDialog(null);
    //Comprobamos que pulse en aceptar
    if (flag == JFileChooser.APPROVE_OPTION) {
        try {
            System.out.println(selector.getSelectedFile());

            File folder = selector.getSelectedFile();
            final File[] listOfFiles = folder.listFiles();

            Thread thrSetupFiles = new Thread() {
                public void run() {
                    setupImages(listOfFiles); //lectura de imagenes
                    procesarImagenes(); //filtros de imagenes
                }
            };

            thrSetupFiles.start();
        } catch (Exception e) {
        }
    }
}
```

Figura 36. Botón Cargar imágenes

Método procesarImagenes

El método procesarImagenes, recoge la lista de imágenes generadas en los distintos tamaños y se realiza el procesamiento de imágenes. (Ver Figura 37)

```
private void procesarImagenes() {
      //List<Thread> threads = new ArrayList<Thread>();

       for (final ImageObject imagen : listaImagenes) {
                    System.out.println("Inicia lectura blur");
                    procesarEliminarRuido(imagen);
                    procesarCanny(imagen);
                }

       System.out.println("finalizó");
   }
```

Figura 37. Método procesar imágenes

Método procesarEliminarRuido

En la Figura 38 se indica el método procesarEliminarRuido, el cual verifica la dirección que el usuario seleccionó y escribe en un archivo plano los pixeles de la imagen y al último pixel le asigna la dirección.

```
public void procesarEliminarRuido(ImageObject imagen) {
   synchronized (this) {
       try {

           BufferedImage[] images = ImageUtilities.eliminarRuido(imagen);
           //se asigna la dirección del panel y se la escribe en el último pixel
           String direccion = Utils.getSelectedButtonText(buttonGroup1);
           //escribe las imagenes en el archivo plano
           ImageUtilities.saveInFile(images, direccion,2);
       } catch (IOException ex) {
           Logger.getLogger(CargaImagen.class.getName()).log(Level.SEVERE, null, ex);

       }
   }
}
```

Figura 38. Método procesar eliminar ruido

Método eliminarRuido

Este método recibe un arreglo de imágenes y las procesa utilizando librerías OpenCV, como se indica en la Figura 39.

```
public static BufferedImage[] eliminarRuido(ImageObject image2) throws IOException
{
    BufferedImage[] target = image2.arrayImages.clone();
    BufferedImage[] result = new BufferedImage[target.length];
    try      {
     System.loadLibrary( Core.NATIVE_LIBRARY_NAME );
        for (int k = 0; k < target.length; k++)
        {
            BufferedImage image = target[k];
            byte[] data = ((DataBufferByte) image.getRaster().getDataBuffer()).getData();
            Mat mRgb = new Mat(image.getHeight(), image.getWidth(), CvType.CV_8UC3);
            mRgb.put(0, 0, data);
            Mat mGray = new Mat(image.getHeight(),image.getWidth(),CvType.CV_8UC1);
            Imgproc.cvtColor(mRgb, mGray, Imgproc.COLOR_RGB2GRAY);

            Mat mBlur = new Mat(image.getHeight(),image.getWidth(),CvType.CV_8UC1);
            Size s = new Size(5, 5);
            Imgproc.GaussianBlur(mGray, mBlur, s,0);
            byte[] data1 = new byte[mBlur.rows() * mBlur.cols() * (int)(mBlur.elemSize())];
            mBlur.get(0, 0, data1);
            BufferedImage image1 = new BufferedImage(mBlur.cols(),mBlur.rows(), BufferedImage.TYPE_BYTE_GRAY);
            image1.getRaster().setDataElements(0, 0, mBlur.cols(), mBlur.rows(), data1);
              result[k] = image1;
        }

  } catch (Exception e) {
     System.out.println("Error: " + e.getMessage());
  }

    return result;
}
```

Figura 39. Método eliminar ruido

Método Modelo

El método modelo que se indica en la Figura 40, carga la estructura del modelo que es un arff, el último atributo le asigna como índice para crear el modelo.

```
public Modelo(String model, String structure) {
    try {
        cls = (Classifier) SerializationHelper.read(model); //Carga el modelo
        //Carga la estructura de los datos que hallará en el archivo
        // "structure.arff"
        dataSet = new ConverterUtils.DataSource(structure).getDataSet();
        //Asigna el índice de la clase al dataset (es el último atributo)
        dataSet.setClassIndex(dataSet.numAttributes() - 1);
    } catch (Exception ex) {
    }
}
```

Figura 40. Método modelo

Método ClasificarInstancia

En la Figura 41 se presenta el método ClasificarInstancia, que clasifica la nueva instancia, es decir recibe la nueva imagen y dice a qué dirección pertenece.

```
public String clasificarInstancia(double[] data) {
    try {
        //Crea una instancia vacía y le asigna la estructura (dataset) cargada
        Instance inst = new Instance(dataSet.numAttributes());
        inst.setDataset(dataSet);
        //Incluye los datos que acaba de cargar en la instancia vacía creada
        for (int i = 0; i < data.length; i++) {
            inst.setValue(i, data[i]);
        }
        //Clasifica la nueva instancia según el modelo cargado
        double pred = cls.classifyInstance(inst);
        return inst.classAttribute().value((int) pred); //Devuelve la clasificación obtenida
    } catch (Exception e) {
        System.out.println(e);
        return null;
     }
   }
}
```

Figura 41. Método clasificar instancia

4.3.2. APLICACIÓN ANDROID

A continuación se presenta las herramientas y el proceso que se siguió para el desarrollo de la aplicación Android.

4.3.2.1. Herramientas

Para la aplicación Android que realiza el seguimiento de la ruta se utilizó las siguientes herramientas:

- Java - Eclipse Juno SDK Android 4.4.2
- OpenCV (para Android)
- Weka (para Windows)
- Weka (para Android)
- Robot con conexión bluetooth
- Dispositivo móvil

4.3.2.2. Instalación

ANDROID SDK

- Se descarga el instalador del SDK de Android desde la página https://developer.android.com/sdk/index.html
- Instalar el SDK en la ruta por defecto. (Ver Figura 42)
- Descargar y actualizar los paquetes necesarios del SDK

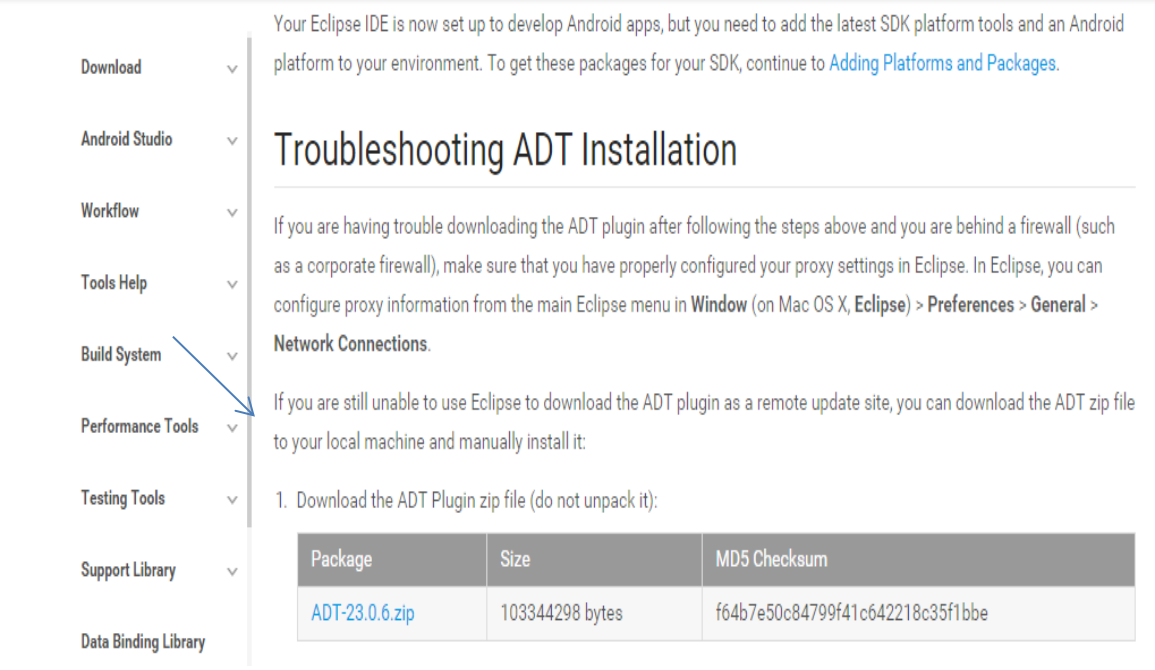

Figura 42. Descarga SDK Android

OPENCV – ANDROID

- Se descarga OpenCV para Android desde la página www.opencv.org (ver Figura 43).

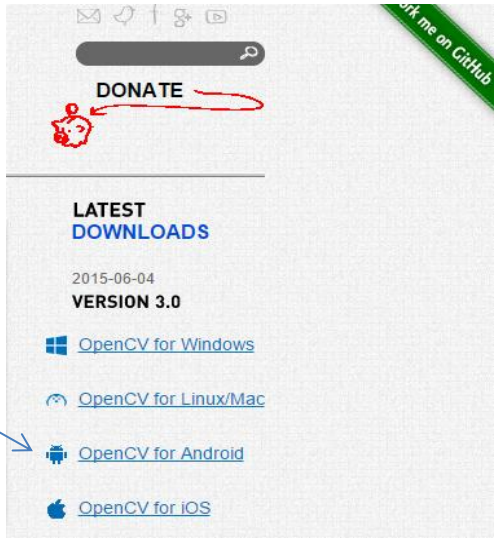

Figura 43. Descarga OpenCV para Android

- Se debe abrir desde eclipse la librería OpenCV, y en el proyecto que se va a utiliza la librería se debe cargar como se muestra en la Figura 44.

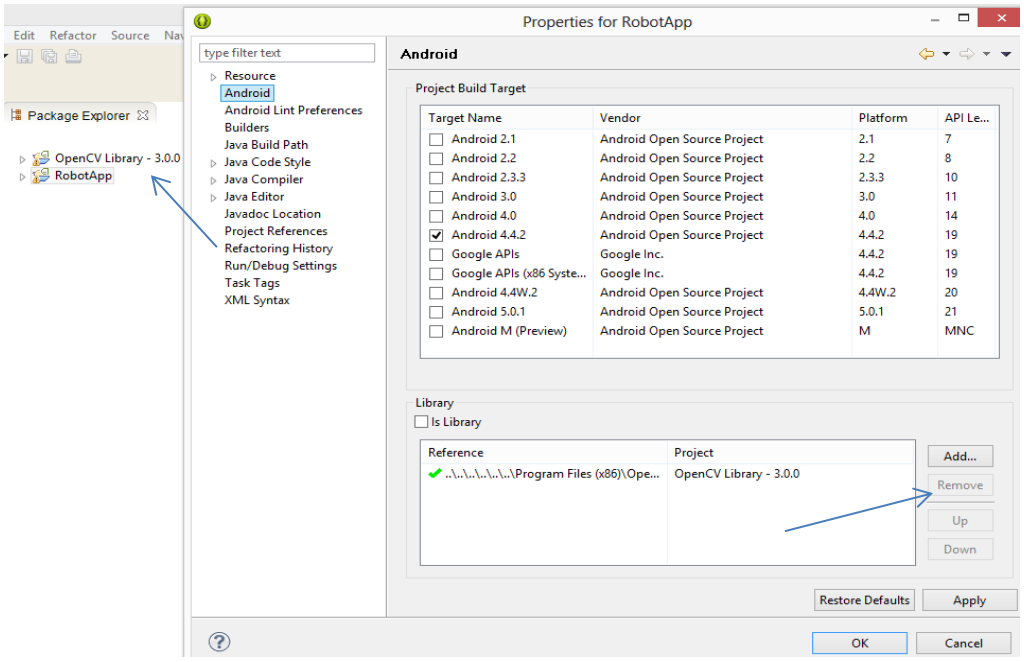

Figura 44. Configuración librería OpenCV para Android

WEKA

En caso de weka se utilizó el mismo ya instalado en Windows.

- Se carga la librería Weka en la plataforma Android como se indica en la Figura 45.

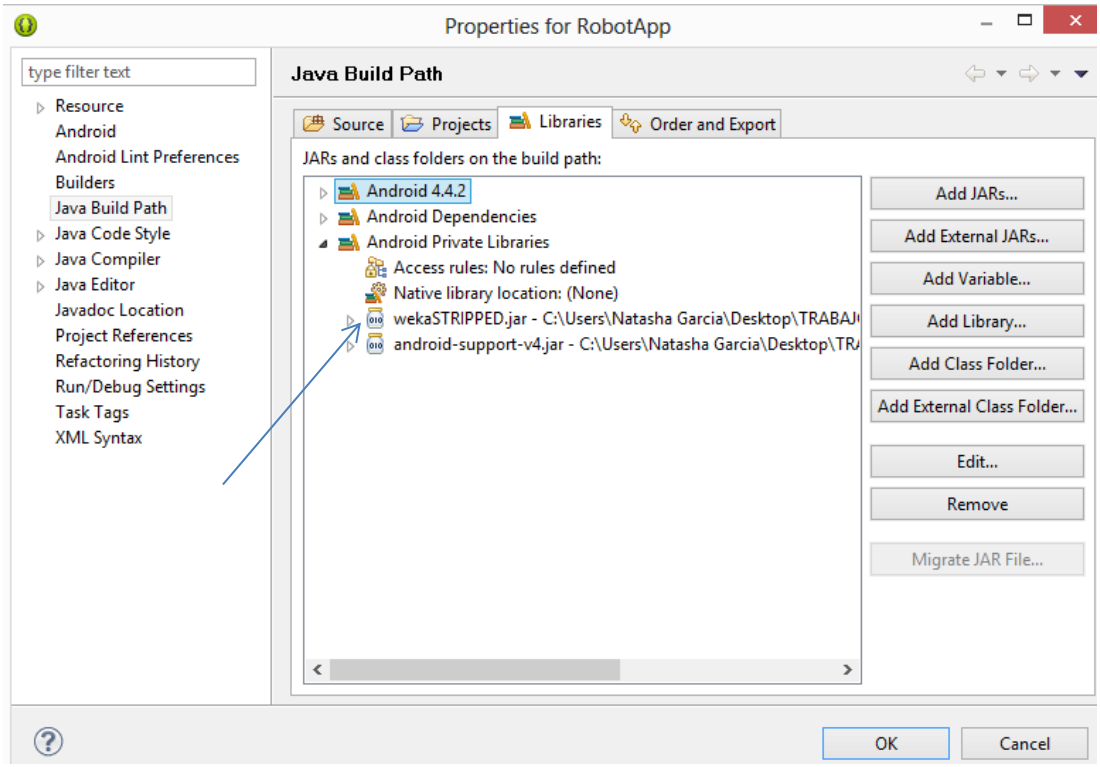

Figura 45. Configuración librería Weka para Android

4.3.2.3. Elección del Modelo

Para la obtención del modelo con mayor exactitud se requiere seguir un proceso, el primer paso es la obtención de las imágenes y el siguiente es el preprocesamiento, el cual se lo implementó en java Netbeans. (Ver Figura 46)

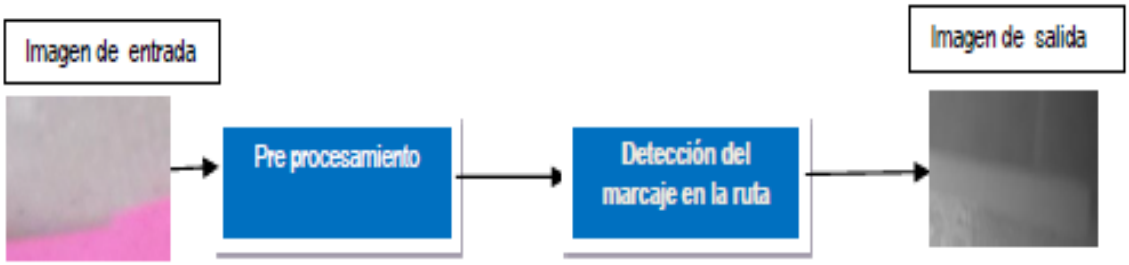

Figura 46. Proceso del programa Netbeans

El objetivo es presentar cuál es el procesamiento de imagen que permite obtener mejores resultados en conjunto con los algoritmos de Weka. Para esto se realizó paquetes de 70 imágenes por cada clasificación, es decir izquierda, centro y derecha las direcciones que el robot seguirá, dando un total de 210 imágenes iniciales. Cada una de estas imágenes genera 3 imágenes de tamaños 5x5, 28x28 y 50x50.

Luego de ejecutar el programa de Netbeans para crear el modelo se seguirá el siguiente proceso:

- Abrir la carpeta con los archivos generados por el programa de Netbeans como se muestra en la Figura 47.

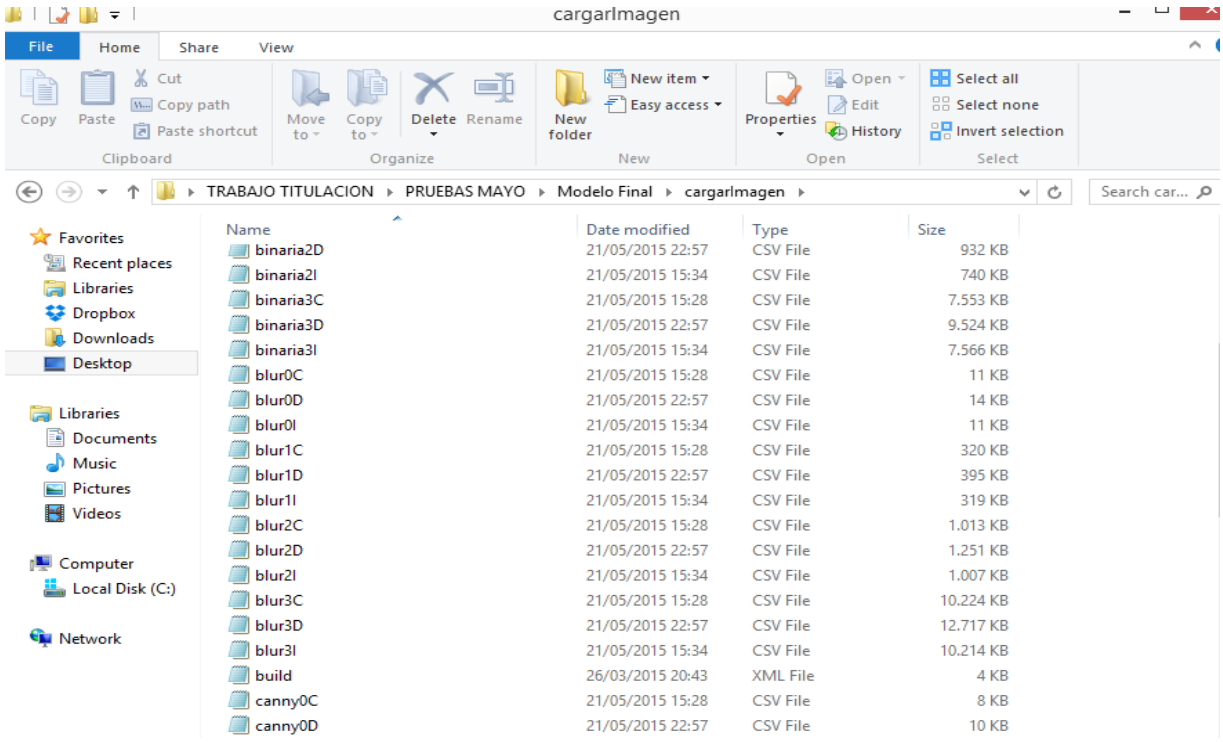

Figura 47. Archivos resultantes del programa Netbeans

- Se debe unir en un solo archivo todos separándolos por procesamiento de imágenes y tamaño, por ejemplo blur1c + blur1d + blur1i. Estas son las imágenes blur con tamaño 28x28, como se observa en la figura 47 son de tipo CVS.

- Para probar el modelo en Weka, es necesario convertir estos archivos en ARFF, para esto se abre el archivo, se aumenta @data antes de todos los datos de las imágenes y se guarda como arff como indica la Figura 48.

```
File  Edit  Format  View  Help
@attribute pixel27.16   real
@attribute pixel27.17   real
@attribute pixel27.18   real
@attribute pixel27.19   real
@attribute pixel27.20   real
@attribute pixel27.21   real
@attribute pixel27.22   real
@attribute pixel27.23   real
@attribute pixel27.24   real
@attribute pixel27.25   real
@attribute pixel27.26   real
@attribute pixel27.27   real
@attribute class        {I,C,D}

@data
139.0,148.0,164.0,178.0,190.0,199.0,205.0,208.0,209.0,209.0,211.0,214.0,219.0
,224.0,231.0,236.0,241.0,244.0,246.0,246.0,244.0,239.0,233.0,225.0,218.0,210.
0,204.0,202.0,139.0,148.0,165.0,178.0,190.0,199.0,204.0,207.0,208.0,209.0,211
.0,214.0,219.0,224.0,231.0,237.0,241.0,245.0,246.0,246.0,244.0,239.0,233.0,22
6.0,218.0,210.0,205.0,202.0,140.0,148.0,165.0,178.0,190.0,199.0,204.0,206.0,2
```

Figura 48. Creación de archivos arff

- Se abre Weka versión 3.6 y se obtiene la pantalla reflejada en la Figura 49, donde se selecciona la opción Experimenter

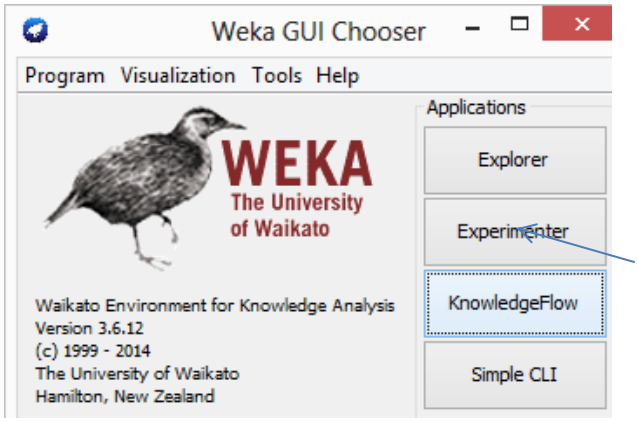

Figura 49. Weka pantalla inicial

- En la pantalla que se muestra en la Figura 50 se cargan todos los archivos arff y todos los algoritmos que se deseen probar, para ver qué modelo es el más apropiado para este sistema.

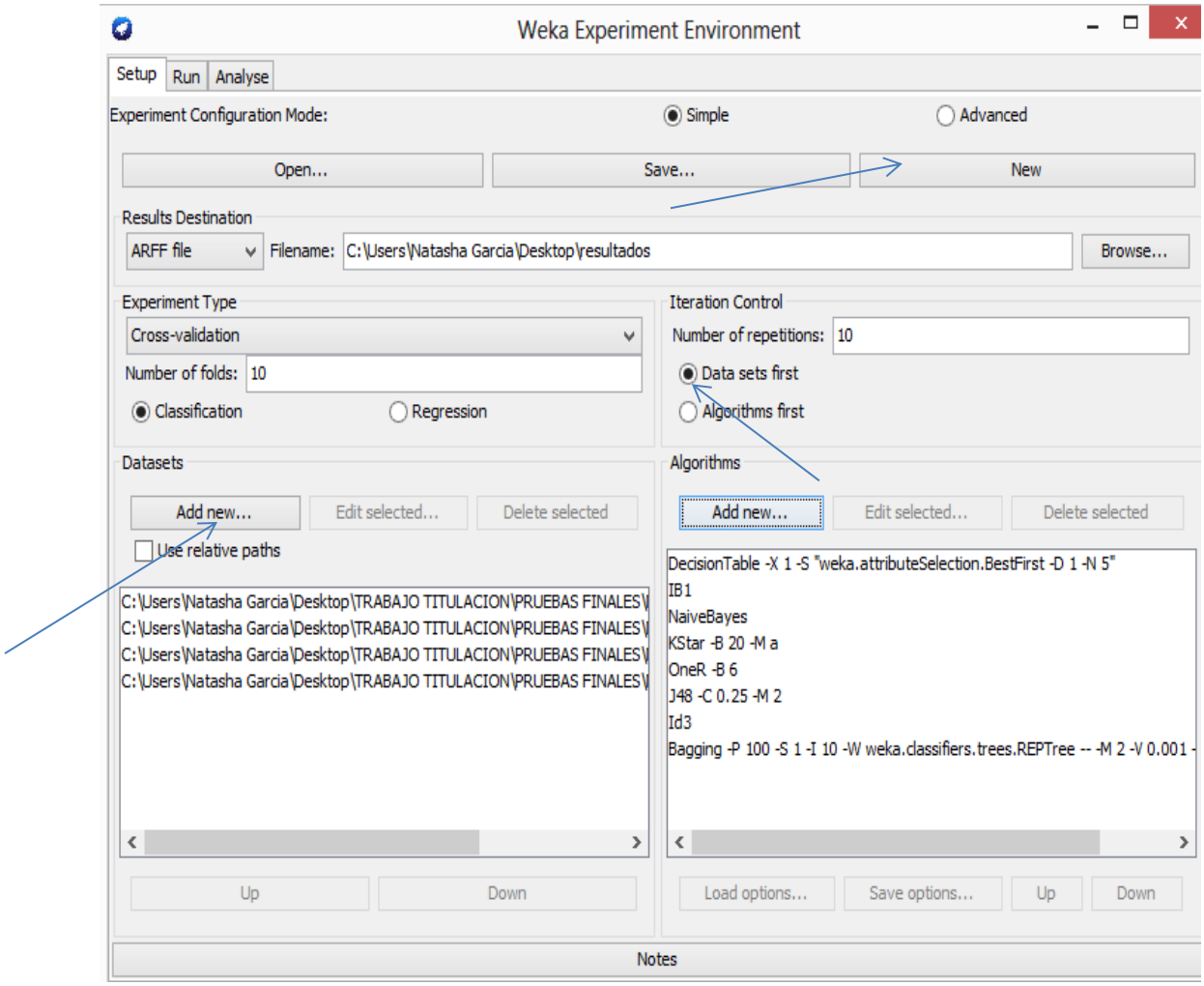

Figura 50. Weka Carga de algoritmos y arff

- Al finalizar se da clic en la pestaña analizar, y se presentan los resultados como se indica en la Figura 51.
- Aquí se genera el archivo .model que se deberá cargar en la aplicación Android creada en la plataforma Eclipse.

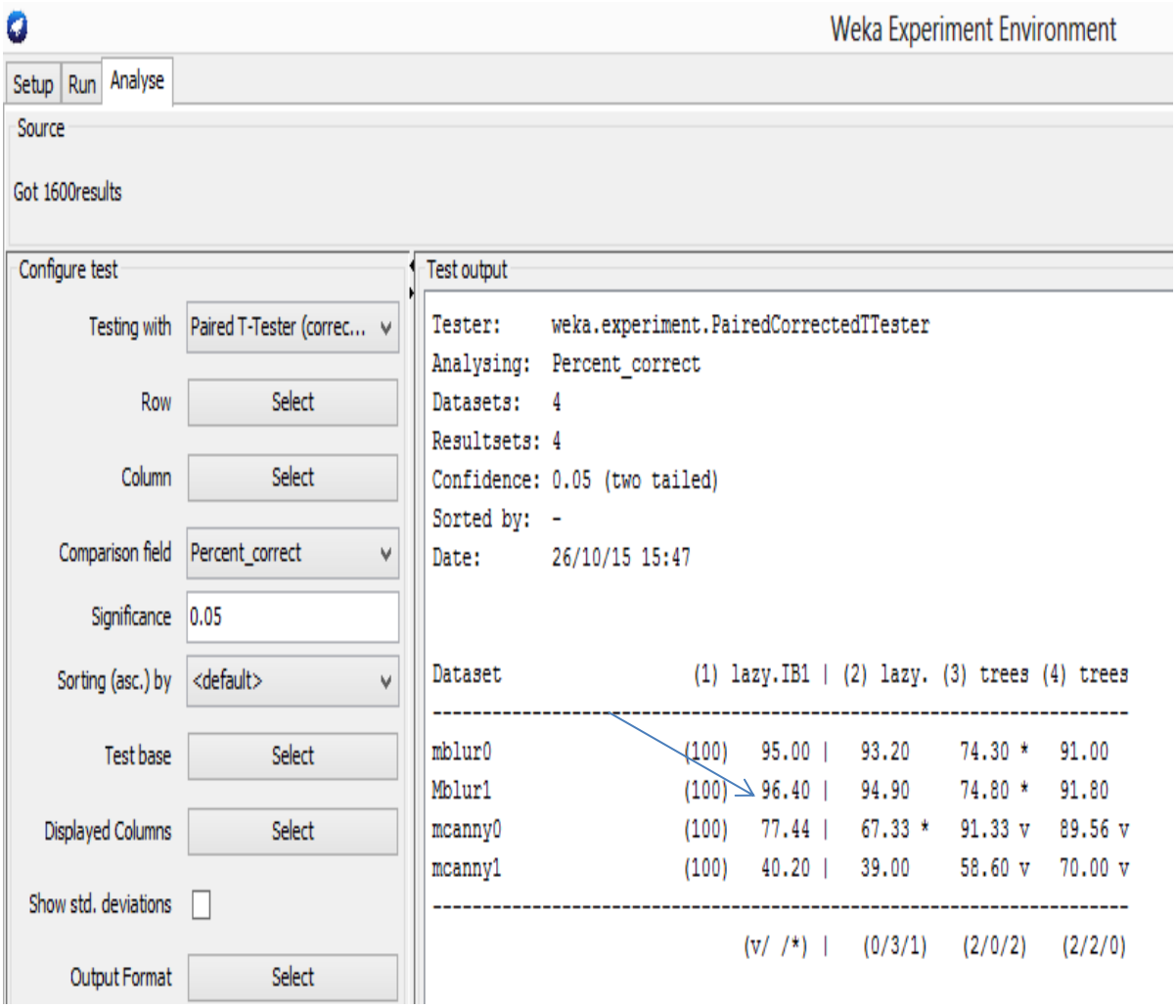

Figura 51. Mejor resultado analizador Weka

- Se verifica el mejor resultado para luego implementarlo en la aplicación Android.

4.3.2.4. Diseño y Código fuente

Para la realización de la aplicación Android se diseñó una interfaz muy sencilla como se muestra en la Figura 52, ya que a través de un único boton la aplicación empieza a ejecutarse por sí sola.

Figura 52. Interfaz gráfica aplicación Android

El proyecto consta de varias clases, agrupadas en dos paquetes que se muestra en la Figura 53. El primer paquete es ceroc.robotapp donde se tiene:

- **ConexionActivity.java**, es la clase principal, donde se establece la conexión entre el robot y la aplicación para luego si procesar la imagen.
- **MainActivity.java**, luego de establecer conexión con el robot, aquí están los métodos donde se procesa la imagen.
- **RobotAplication.java**, clase necesaria para establecer conexión bluetooth con el robot.
- **ViewOpenCV.java**, clase necesaria para la utilización de la cámara OpenCV y sus métodos propios.

El segundo paquete es Utils que contiene dos clases:

- **UtilsRobotApp.java**, contiene los métodos de cortar imagen, redimensionar y pasar la imagen a un archivo de pixeles.
- **Weka.java**, contiene los métodos para crear el modelo, leerlo y clasificar nuevas instancias.

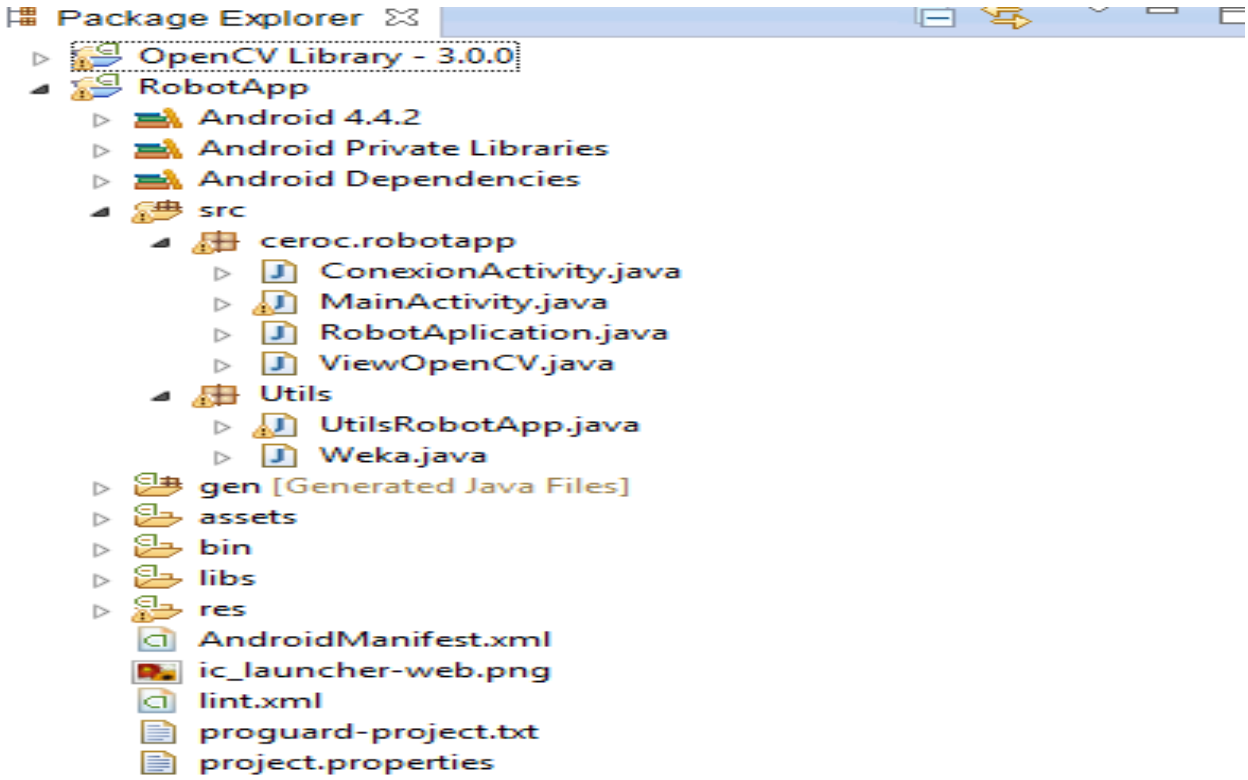

Figura 53. Vista del proyecto RobotApp

A continuación se muestra el código fuente de los métodos y botones principales que se utilizaron en el desarrollo de la aplicación Android, tomando en cuenta que la clase principal es la ConexionActivity.java ya que una vez que se conecta con el robot pasa a la siguiente clase que es la que ejecuta todos los métodos para la detección de la ruta.

Método onCreate ConexionActivity.java

El método onCreate es el primero en ejecutarse, donde aparece la pantalla inicial y se muestra en la Figura 54.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_conexion);
}
```

Figura 54. Método onCreate

Método conectarIracer

Este método se encarga de realizar la conexión por medio de bluetooth con el robot., el código se visualiza en la Figura 55.

```
public void conectarIracer(View v){
    //verifica si puede utilizar Bluetooth
    bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if (bluetoothAdapter == null)   {
        Toast.makeText(this, "Bluetooth is not available", Toast.LENGTH_SHORT).show();
        finish();
        return;
    }
    //verifica si esta apagado
    if (!bluetoothAdapter.isEnabled())    {
        // enable bluetooth
        Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
    }
    //si existe alguna busqueda, cancela cualquier búsqueda actual
    if(bluetoothAdapter.isDiscovering())
        bluetoothAdapter.cancelDiscovery();
    // Register for broadcasts when a device is discovered
    IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
    this.registerReceiver(mReceiver, filter);
    // Register for broadcasts when discovery has started
    filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
    this.registerReceiver(mReceiver, filter);
    // Register for broadcasts when discovery has finished
    filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
    this.registerReceiver(mReceiver, filter);
    //inicia la búsqueda del iRacer
    bluetoothAdapter.startDiscovery();
}
```

Figura 55. Método conectar robot

Método onReceiver

El método onReceiver es el que acepta las conexiones bluetooth, busca nuevas conexiones, y verifica la conexión con el robot. (Ver Figura 56)

```
public final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();    //detecta nuevo dispositivo
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            Toast.makeText(getApplicationContext(),
                "Otro BT: " + device.getName() + ":" + device.getAddress(), Toast.LENGTH_SHORT).show();
            if (device.getAddress().equals(MY_IRACER_ADDR)) {
                myIracerDev = device;
                Toast.makeText(getApplicationContext(), "iRacer encontrado", Toast.LENGTH_SHORT).show();
                 try {
                        socket = myIracerDev.createRfcommSocketToServiceRecord(SPP_UUID);
                        socket.connect();
                        RobotAplication app = (RobotAplication) getApplicationContext();
                        app.setSocket(socket);
                        Toast.makeText(getApplicationContext(), "Conectado !!!", Toast.LENGTH_SHORT).show();
                        bluetoothAdapter.cancelDiscovery();
                        unregisterReceiver(mReceiver);
                        showMainActivity();
                 } catch (Exception e) {
                        Toast.makeText(getApplicationContext(), "No se puede conectar a Bluetooth", Toast.LENGTH_SHORT).show();
                        socket = null;
                        e.printStackTrace();
                    }
                }
        } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
            Toast.makeText(getApplicationContext(), "Iniciada búsqueda Bluetooth", Toast.LENGTH_SHORT).show();
        // When discovery is finished
        } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
            Toast.makeText(getApplicationContext(), "Finalizada búsqueda Bluetooth", Toast.LENGTH_SHORT).show();
        }}
```

Figura 56. Método onReceiver

Método onCreate MainActivity.java

El método que se ejecuta primero es el onCreate, que muestra la pantalla inicial, donde se instancia Weka y se lee el modelo una sola vez como se indica en la Figura 57.

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    setContentView(R.layout.activity_main);
    weka = new Weka(this);
    mOpenCvCameraView = (ViewOpenCV) findViewById(R.id.viewOpencv);
    mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
    mOpenCvCameraView.setCvCameraViewListener(this);
    socket = ((RobotAplication) getApplicationContext()).getSocket();
}
```

Figura 57. Método onCreate

Método onCameraFrame

Este método es el principal, pues aquí se corta la imagen, para eliminar el mayor ruido posible, luego se redimensiona para convertir la imagen a pixeles, y enviar esos pixeles al método de clasificar nueva instancia, y posterior a eso enviar la dirección al robot. (Ver Figura 58)

```
@Override
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {

    Imgproc.cvtColor(inputFrame.rgba(), mGray, Imgproc.COLOR_RGBA2GRAY);
    Imgproc.GaussianBlur(mGray, mBlur, new org.opencv.core.Size (5,5), 0);
    Thread thrProcesar = new Thread(new Runnable() {

        @Override
        public void run() {

            Bitmap bmp= UtilsRobotApp.getCropBitmap(mBlur);
            bmp = UtilsRobotApp.getResizedBitmap(bmp, 28, 28);
            double[] pixels = UtilsRobotApp.getImgPixels(bmp);
            String direccion = weka.classifyNewInstance(pixels);
            //Log.i("dirección",  direccion);
            sendCommand(direccion);
        }

    });
    thrProcesar.start();

        return mBlur;

}
```

Figura 58. Método onCameraFrame

Método sendCommand

Este método envía la dirección que se obtiene a partir de weka al robot por medio de códigos hexadecimales, el robot tiene tres direcciones que se representaron por sus iniciales, izquierda, centro y derecha. (Ver Figura 59)

```
 //Para enviar comando al robot
public void sendCommand(String direccion) {
    try
    {
        if(direccion.equalsIgnoreCase("C"))
            curDir = Integer.parseInt("60", 16);//0x50

        else if(direccion.equalsIgnoreCase("I"))
            curDir = Integer.parseInt("50", 16);//0x50

        else if(direccion.equalsIgnoreCase("D"))
            curDir = Integer.parseInt("10", 16);//0x50
        else
            curDir = 0;//0x00;
            curSpeed = 0;//0x00;


        socket.getOutputStream().write(curDir+curSpeed+MIN_SPEED );
        socket.getOutputStream().flush();
        //Log.i("iRacerCommand", "::" + Integer.toHexString(curDir + curSpeed + MIN_SPEED));
    } catch (Exception e) {
        Toast.makeText(getApplicationContext(), "No se puede enviar comando Bluetooth", Toast.LENGT
        e.printStackTrace();
    }
}
```

Figura 59. Método sendCommand

Dentro del segundo paquete Utils están los métodos de procesamiento y obtención de pixeles de la imagen y clasificación de nuevas instancias con el modelo de Weka, que se presentan a continuación.

Método getImgPixels

En la Figura 60 se indica el método getImgPixels que recibe la imagen y obtiene un arreglo de los pixeles de dicha imagen.

```
public static double[] getImgPixels(Bitmap img) {
    double[] pixeles = new double[img.getWidth() * img.getHeight()];
    for(int row = 0; row < img.getHeight(); row++) {
        for(int col = 0; col < img.getWidth(); col++) {
            int pixel = img.getPixel(col, row);
            pixeles[row * img.getHeight() + col] = Color.red(pixel);

        }
    }
    return pixeles;
}
```

Figura 60. Método getImgPixels

Método rotateBitmap

Este método permite rotar la imagen, recibe como entrada la imagen y el ángulo de rotación. (Ver Figura 61)

```
public static Bitmap rotateBitmap(Bitmap bmp, float angulo) {
    Matrix matrix = new Matrix();
    matrix.postRotate(angulo);
    return  Bitmap.createBitmap(bmp, 0, 0, bmp.getWidth(), bmp.getHeight(), matrix, true);
}
```

Figura 61. Método rotateBitmap

Método getRisizedBitmap

El método getRisizedBitmap que se indica en la Figura 62, permite redimensionar la imagen, es muy importante para no trabajar con demasiada cantidad de pixeles, ya que para clasificar una instancia se trabaja con una dimensión de imagen especifica.

```
public static Bitmap getResizedBitmap(Bitmap bm, int newHeight, int newWidth) {
    int width = bm.getWidth();
    int height = bm.getHeight();
    float scaleWidth = ((float) newWidth) / width;
    float scaleHeight = ((float) newHeight) / height;
    // CREATE A MATRIX FOR THE MANIPULATION
    Matrix matrix = new Matrix();
    // RESIZE THE BIT MAP
    matrix.postScale(scaleWidth, scaleHeight);
    // RECREATE THE NEW BITMAP
    Bitmap resizedBitmap = Bitmap.createBitmap(bm, 0, 0, width, height, matrix, false);
    return resizedBitmap;
    }
```

Figura 62. Método getResizedBitmap

Método getCropBitmap

Este método corta el bitmap, ese se lo utiliza para eliminar una parte de la imagen ya que existe demasiado ruido. (Ver Figura 63)

```
public static Bitmap getCropBitmap( Mat uncropped) {

    int y= uncropped.rows()/3;
    Rect roi = new Rect(0,y , uncropped.cols(), uncropped.rows()-y-1);
    //Log.e("TAG","valores"+y+"col"+uncropped.cols()+"filas"+uncropped.rows());
    Mat cropped = new Mat(uncropped, roi);
    Bitmap bmp = Bitmap.createBitmap(cropped.cols(), cropped.rows(),Bitmap.Config.ARGB_8888);
    Utils.matToBitmap(cropped, bmp);
    return bmp;
}
```

Figura 63. Método getCropBitmap

Método createModel

Este método que se muestra en la Figura 64, recibe el modelo generado en Netbeans, y vuelve a crearlo a partir del código Android. Solo se lo genera una vez, ya que el modelo creado en Netbeans no es compatible, luego únicamente se lo lee para clasificar la instancia nueva.

```
public void createModel(Context context)
{
    try
    {
        Log.i(TAG, "Inicia la creacion del modelo");
        //Carga el dataset del archivo "iris.arff"
        InputStream isdata = context.getAssets().open("mblur0.arff");
        DataSource datasource = new DataSource(isdata);
        Instances data = datasource.getDataSet();
        data.setClassIndex(data.numAttributes() - 1);

        //Constuye el modelo usando IBk
        IBk ibk = new IBk(1);
        ibk.buildClassifier(data);
        //Guarda el modelo en el archivo "model.android.ibk.blur50"
        File file = new File(Environment.getExternalStorageDirectory(), "model.android.ib1.mblur0");
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file));
        oos.writeObject(ibk);
        oos.flush();
        oos.close();

        Log.i(TAG, "Modelo creado. Path: " + file.getAbsolutePath());

    }
    catch (Exception e)
    {
        Log.e(TAG, "Error en la creacion del modelo: ", e);
    }
}
```

Figura 64. Método createModel

Método readModel

Este método lee el modelo generado en Android, se lee una sola vez el modelo y luego se llama a clasificar instancia. (Ver Figura 65)

```
private void readModel()
{
    try {
        //Carga el modelo que se encuentra en el archivo "model.android.j48.iris"
        //InputStream ismodel = this.getAssets().open("model.android.j48.iris");
        InputStream ismodel = context.getAssets().open("model.android.ib1.mblur0");
        ObjectInputStream ois = new ObjectInputStream(ismodel);
        cls =  (Classifier) ois.readObject();
        ois.close();
        //Carga la estructura de los datos que se halla en el archivo "iris.structure.arff"
        InputStream isstructure = context.getAssets().open("mblur0structure.arff");
        dataSet = new ConverterUtils.DataSource(isstructure).getDataSet();
        isstructure.close();
        //Asigna el índice de la clase al dataset (es el último atributo)
        dataSet.setClassIndex(dataSet.numAttributes() - 1);
        inst = new DenseInstance(dataSet.numAttributes());
        inst.setDataset(dataSet);
    }
    catch (Exception e) {
        Log.e(TAG, "Error leyendo el modelo: ", e);
    }
}
```

Figura 65. Método readModel

Método classifyNewInstance

En la Figura 66 se indica el método classifyNewInstance, que lee el modelo generado en Android, se lee una sola vez el modelo y luego se llama a clasificar instancia.

```
public String classifyNewInstance(double[] pixels)
{
    String direccion = null;
    try
    {
        //Crea una instancia vacía y le asigna la estructura (dataset) cargada

        for(int i = 0; i < pixels.length; i++)
            inst.setValue(i, pixels[i]);
        //Clasifica la nueva instancia según el modelo cargado
        double pred = cls.classifyInstance(inst);
        //Devuelve la clasificación obtenida

        direccion = inst.classAttribute().value((int) pred);
        //Log.i(TAG, "Instancia clasificada: " + direccion);
    }
    catch (Exception e) {
        Log.e(TAG, "Error clasificando la instancia: ", e);
    }
    return direccion;
}
```

Figura 66. Método classifyNewInstance

Dentro del proyecto es importante dar los permisos dependiendo la aplicación que se esté desarrollando, en este caso se necesita:

- Permisos para utilizar la cámara
- Permisos de bluetooth
- Permisos para almacenamiento de archivos externo, este último es solo inicialmente para guardar las imágenes y verificar si el proceso se está haciendo correctamente.

El Android Manifest es un archivo de configuración de la aplicación Android que se está desarrollando, donde los permisos juegan un papel importante, y depende de las herramientas que se estén utilizando.

Android Manifest Permissions

En la Figura 67 se presenta el archivo de configuración con los permisos respectivos.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ceroc.robotapp"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk       android:minSdkVersion="17"
                    android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:name="RobotAplication"
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".ConexionActivity"
            android:label="@string/title_activity_conexion" >
             <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

4.3.3. PRUEBAS

Las pruebas de la aplicación se realizaron en el dispositivo móvil con sistema operativo Android, las mismas que ayudaron a optimizar el proceso para la detección de la dirección y envío de la misma al robot.

- Uno de los primeros problemas que se presentaron durante la ejecución del proyecto fue el tiempo de respuesta de clasificación de la nueva instancia, ese problema se generó cuando se realizaba el procesamiento de la imagen utilizando las librerías OpenCV, ya que inicialmente se estaba utilizando la cámara propia del dispositivo móvil y para obtener la imagen de la cámara era necesario utilizar un SurfaceView, los resultados que se obtuvieron se reflejan en la Figura 67.

L...	Time	PID	Application	Tag	Text
I	09-28 10:03:06.933	28966	com.android.gra...	t	total direcciones:17
I	09-28 10:03:07.934	28966	com.android.gra...	t	total direcciones:16
I	09-28 10:03:08.925	28966	com.android.gra...	t	total direcciones:16
I	09-28 10:03:09.926	28966	com.android.gra...	t	total direcciones:12

Figura 67: Tiempo de respuesta (SurfaceView)

- Para dar solución al problema se utilizó la cámara propia de las librerías OpenCV, donde se obtuvo entre 4-6 direcciones por segundo, como se indica en la Figura 68.

L...	Time	PID	Application	Tag
E	10-02 15:24:56.993	9211	ceroc.r...	DIRECCION
E	10-02 15:24:56.993	9211	ceroc.r...	DIRECCION
E	10-02 15:24:57.274	9211	ceroc.r...	DIRECCION
E	10-02 15:24:57.414	9211	ceroc.r...	DIRECCION
E	10-02 15:24:57.624	9211	ceroc.r...	DIRECCION
E	10-02 15:24:57.664	9211	ceroc.r...	DIRECCION
E	10-02 15:24:57.784	9211	ceroc.r...	DIRECCION
E	10-02 15:24:57.834	9211	ceroc.r...	DIRECCION
E	10-02 15:24:58.195	9211	ceroc.r...	DIRECCION
E	10-02 15:24:58.215	9211	ceroc.r...	DIRECCION
E	10-02 15:24:58.315	9211	ceroc.r...	DIRECCION
E	10-02 15:24:58.645	9211	ceroc.r...	DIRECCION

Figura 68: Tiempo de respuesta (OpenCV Camera)

- La prueba para obtener el modelo con mejor exactitud se la realizó utilizando la herramienta Weka para distintos algoritmos, se obtuvo la siguiente tabla de resultados reflejada en la Figura 69.

Dataset	(1) bayes.Na	(2) lazy.	(3) lazy.	(4) rules	(5) trees	(6) trees	(7) trees	(8) meta.	(9) meta.	(10) meta
binario0	(100) 40.43	38.90	38.48	33.33 *	40.48	38.10	38.14	41.19	38.76	32.05 *
binario1	(100) 42.81	58.43 v	33.33 *	33.33 *	46.62	54.19 v	44.67	49.90	49.48	35.67 *
binario2	(100) 43.38	52.24 v	33.33 *	33.33 *	47.00	56.14 v	44.62	50.81	51.05	33.24 *
blur0	(100) 38.29	45.71	46.29	33.33	36.29	46.19	43.14	38.62	45.43	34.71
blur1	(100) 61.86	83.57 v	33.33 *	33.33 *	67.57	81.76 v	72.48 v	72.05 v	79.62 v	45.62 *
blur2	(100) 63.43	88.76 v	33.33 *	33.33 *	68.57	83.67 v	72.24 v	71.14 v	81.90 v	45.57 *
canny0	(100) 51.57	53.71	54.67	33.33 *	53.52	53.62	53.19	56.05	53.76	47.62
canny1	(100) 41.10	45.48	33.33 *	33.33 *	35.67	44.95	36.00	38.62	43.14	32.14 *
canny2	(100) 43.00	36.10	33.33 *	33.33 *	39.81	45.86	38.43	43.38	41.67	34.90 *
grises0	(100) 54.57	67.62 v	61.19	33.33 *	56.05	67.57 v	55.38	62.33	65.71 v	47.14
grises1	(100) 55.14	69.95 v	33.33 *	33.33 *	53.67	68.29 v	55.95	61.90 v	67.90 v	41.52 *
grises2	(100) 55.05	71.05 v	33.33 *	33.33 *	55.19	68.29 v	55.19	60.57	66.76 v	43.43 *
(v/ /*)		(7/5/0)	(0/4/8)	(0/1/11)	(0/12/0)	(7/5/0)	(2/10/0)	(3/9/0)	(5/7/0)	(0/3/9)

Key:
(1) bayes.NaiveBayes '' 5995231201785697655
(2) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -30801
(3) lazy.KStar '-B 20 -M a' 332458330800479083
(4) rules.ZeroR '' 48055541465867954
(5) trees.J48 '-C 0.25 -M 2' -217733168393644444
(6) trees.RandomForest '-I 100 -K 0 -S 1' -2260823972777004705
(7) trees.RandomTree '-K 0 -M 1.0 -S 1' 8934314652175299374
(8) meta.Bagging '-P 100 -S 1 -I 10 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1' -5178288489778728847
(9) meta.RandomCommittee '-S 1 -I 10 -W trees.RandomTree -- -K 0 -M 1.0 -S 1' -9204394360557300092
(10) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.DecisionStump' -7378107808933117974

Figura 69: Resultados Iniciales en analizador Weka

Se obtuvo como mejor resultado 88.76v que representa el dataset blur2 con el tamaño de las imágenes 28x28 aplicando el algoritmo de los vecinos más cercanos lazy.Ibk con k = 1.

- La imagen que mira a continuación es cuando el robot captaba las direcciones, y se realizó la prueba de conectividad con el robot, fue muy óptima. En la Figura 70 podrá ver los comandos en hexadecimal que se envía desde la aplicación al robot.

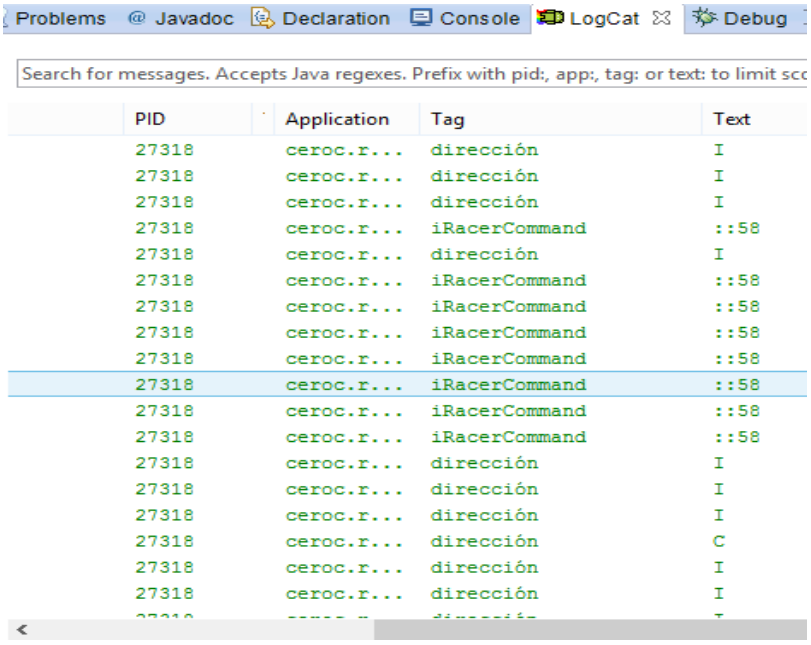

Figura 70: Conexión de la aplicación con el robot

- Se realizó la primera prueba ya en el robot y no se obtuvo buenos resultados, pues el set de datos con los cuales se entrenó al modelo no estaban en iguales condiciones de claridad, el patrón era distinto, entre otras cosas que se presentaron.
- Se realizó varios modelos nuevamente utilizando el programa de Netbeans, es decir se generaron 8 modelos nuevos tomando nuevas fotografías. Se tomaron varios set de imágenes hasta obtener el de mejor resultado como se indica la figura 71.

Figura 71: Set de imágenes para crear el modelo

- En la generación de los nuevos modelos, el último con mayor porcentaje de efectividad fue aplicando el algoritmo Ib1 para Blur(5,5) con dimensiones de la imagen de 50x50. Sin embargo para la generación del modelo debido a los tiempos de respuesta se tuvo que utilizar las dimensiones de 28x28. (ver Figura 72)

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   4
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       26/10/15 15:47


Dataset                   (1) lazy.IB1 | (2) lazy. (3) trees (4) trees
----------------------------------------------------------------------
mblur0                  (100)   95.00 |   93.20     74.30 *   91.00
Mblur1                  (100)   96.40 |   94.90     74.80 *   91.80
mcanny0                 (100)   77.44 |   67.33 *   91.33 v   89.56 v
mcanny1                 (100)   40.20 |   39.00     58.60 v   70.00 v
----------------------------------------------------------------------
                             (v/ /*) |   (0/3/1)   (2/0/2)   (2/2/0)


Key:
(1) lazy.IB1 '' -6152184127304895851
(2) lazy.IBk '-K 3 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(3) trees.J48 '-C 0.25 -M 2' -217733168393644444
(4) trees.RandomForest '-I 100 -K 0 -S 1' -2260823972777004705
```

Figura 72: Resultado Final del analizador Weka

- Se realizó las pruebas de procesamiento de la imagen, para ver si se están aplicando bien los filtros, en un inicio debido al tipo de dato que utiliza OpenCV se obtuvo resultados no esperados reflejados en la Figura 73.

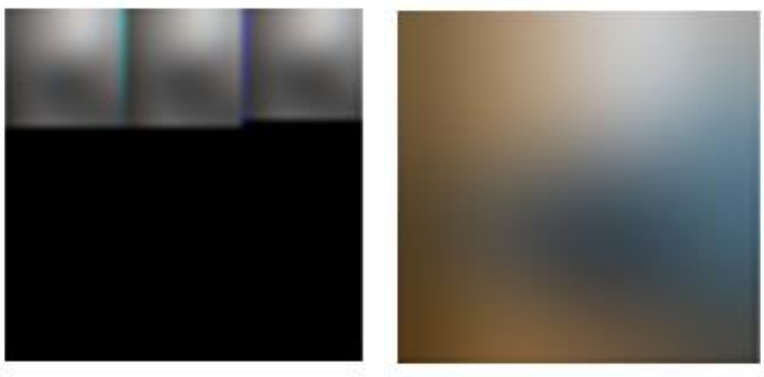

Figura 73: Procesamiento de la imagen con OpenCV

- Se cambió el método que transformaba de tipo bitmap a Mat, necesarios para el procesamiento de la imagen aplicando blur(45,45), obteniendo la Figura 74 como resultado.

Figura 74: Blur(45,45)

- Las pruebas finales se realizaron en el robot y para rutas no tan complejas, donde controlando las condiciones el robot respondió bien. Para solucionar esto se tuvo que cortar la imagen que la cámara recibía pues en la parte superior debido a la luz y sombras del entorno la ruta se perdía. (Ver Figura 75)

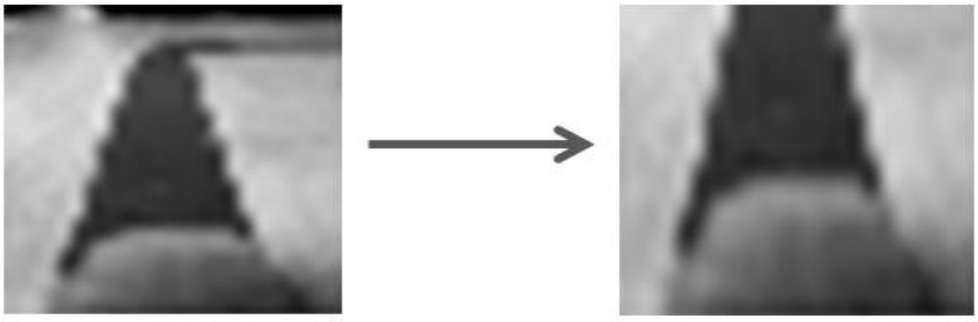

Figura 75: Imagen cortada

4.3.3.1. Características de las imágenes

- Las dimensiones de la imagen en su tamaño original son: ancho 3264, Largo 1836
- Las imágenes originales han sido tomadas mediante una cámara previamente calibrada y tienen formato .jpg.
- La cámara utilizada es de celular, marca Samsung GT-I9195.
- El set de imágenes que se toma para la construcción debe estar lo más aproximado al entorno donde el robot va a encontrarse para el seguimiento de ruta.

4.3.4. MANEJO DE LA APLICACIÓN

Para una correcta instalación seguir los pasos descritos en el manual de instalación expuesto en el Anexo 1. A continuación se muestra la forma de utilizar la aplicación de seguimiento de ruta.

- El robot cuenta con un soporte para ubicar el dispositivo móvil, y con un elástico para sujetarlo como se muestra en la Figura 76, ahí se ubica el dispositivo móvil.

Figura 76: Ubicación del dispositivo móvil en el robot

- Se enciende el robot como se indica en la Figura 77.

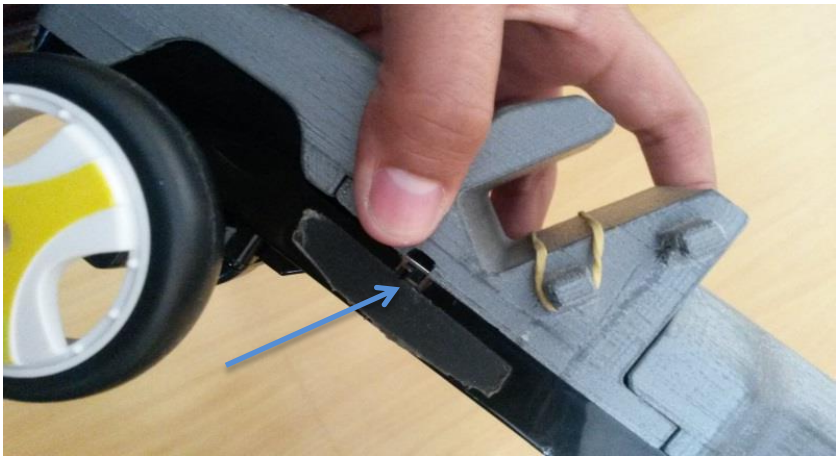

Figura 77: Encender robot

- Se ubica al robot en la ruta diseñada para hacer las pruebas como muestra la Figura 78.

Figura 78: Prueba de la aplicación en la ruta diseñada

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

A continuación se presenta las conclusiones y recomendaciones que se obtuvieron en el desarrollo del proyecto.

5.1. CONCLUSIONES

- Cuando se utilizan aplicaciones de reconocimiento de patrones dentro de una imagen es necesario tomar en cuenta la iluminación del entorno de trabajo, de lo contrario se puede perder las características que se van a extraer de la imagen.
- Se utilizó la metodología OOHDM en complemento con la metodología de prototipado, la primera plantea un proceso adecuado para el desarrollo de aplicaciones móviles y la segunda presenta un documento completo de requerimientos siguiendo el estándar IEEE 830 y un prototipo realizado en Microsoft Power Point para la explicación del proceso al usuario.
- El pre-procesamiento, la clasificación y la visualización previa de las imágenes de entrada permite obtener con mayor facilidad los resultados deseados con OpenCV.
- El uso de librerías de visión artificial como OpenCV provee varios algoritmos para el tratamiento de imágenes para poder trabajar con detección y reconocimiento de patrones.
- Para utilizar librerías OpenCV es necesario utilizar la cámara propia de OpenCV y transformar las imágenes que se obtienen a objetos Mat.
- Weka presenta variedad de algoritmos que por medio del analizador se puede realizar las pruebas para ver que algoritmo se ajusta más al problema que se está resolviendo.
- Las pruebas se realizaron para imágenes con distintos tamaños, y diferentes procesamientos de imagen, el resultado más óptimo fue con tamaño 50x50, sin embargo por el tiempo de respuesta de clasificación se utilizó el tamaño 28x28 si variar mucho la probabilidad de exactitud.

- Los modelos creados en el programa de Netbeans tienen que ser entrenados con imágenes en el ambiente donde se va a someter a pruebas al robot, además se debe tomar en cuenta el ángulo con el que va a estar la cámara del dispositivo móvil.
- En cada nueva prueba los porcentajes de exactitud se han ido acercando a lo óptimo, hasta obtener un 95% de exactitud.

5.2. RECOMENDACIONES

- Se recomienda hacer un sistema para la creación del modelo en la plataforma Netbeans por la velocidad de procesamiento al momento de ejecutar el programa para hacer los cambios, y en especial porque se deben crear varios modelos y probarlos hasta obtener los resultados deseados.
- Se recomienda hacer el procesamiento de las imágenes antes de clasificarla para obtener una respuesta precisa y en menor tiempo, pues la mayoría de pixeles de una imagen son ruido y no se debería tomar en cuenta.
- Antes de desarrollar aplicaciones para reconocimiento de líneas u objetos se debe investigar los algoritmos dependiendo del problema para facilitar el trabajo.
- Se recomienda entrenar la aplicación móvil con un set de imágenes no menores a 70, pues la probabilidad de exactitud varía y entre mayor cantidad de imágenes de entrenamiento se tenga el sistema será más preciso.

BIBLIOGRAFÍA

Alarcón, V. F. (2010). *Desarrollo de sistemas de información: una metodología basada en el modelado*. Univ. Politèc. de Catalunya.

Alpaydin, E. (2009). *Introduction to Machine Learning* (second edition edition). Cambridge, Mass: The MIT Press.

Araujo, M. I., Ferrari Alve, S. I., & Mariño, S. I. (2012). Revisión de algoritmos de Redes Neuronales en dos herramientas de Minería de Datos. Retrieved May 21, 2014, from http://www.cyta.com.ar/ta1104/v11n4a1.htm

Becerril, I. (2008). Dispositivos móviles, análisis forense y sus futuros riesgos. Retrieved from http://www.revista.unam.mx/vol.9/num4/art26/int26.htm#a

Booch, G., & Rumbaugh, J. (2004). *El lenguaje unificado de modelado*.

Bouckaert, R. R., Frank, E., Hall, M. A., Holmes, G., Pfahringer, B., Reutemann, P., & H.Witten, I. (2010). WEKA--Experiences with a Java Open-Source Project. *Journal of Machine Learning Research*, *11*(9), 2533–2541.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV : Computer Vision with the OpenCV Library* (1st ed.). Sebastopol: O'Reilly Media.

Castro, A. (2011, November 10). Perro guía robótico para personas invidentes. Retrieved from http://www.fierasdelaingenieria.com/perro-guia-robotico-para-personas-invidentes/

Céspedes, E., & González, M. (s.f). *Desarrollo e implementación de un sistema de generación de tests on-line para dispositivos móviles* (Desarrollo aplicaciones móviles). Escuela Politécnica del Ejército, Quito. Retrieved from http://repositorio.espe.edu.ec/bitstream/21000/5394/4/AC-SISTEMAS-ESPE-033274.pdf

Chapman & Hall. (2009). *The Top Ten Algorithms in Data Mining*. Retrieved from https://www.crcpress.com/The-Top-Ten-Algorithms-in-Data-Mining/Wu-Kumar/9781420089646

Chong, M. (2009). Robótica e inteligencia artificial. Retrieved from http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10316298

Coronado, E. (2014). Mecatrónica UASLP. Retrieved June 17, 2014, from https://mecatronicauaslp.wordpress.com/

Cyganek, B. (2013). *Object Detection and Recognition in Digital Images : Theory and Practice* (1st ed.). Hoboken: Wiley.

Esperón, G., Laudadio, A., Martínez, A., Serrano, E., & López, D. (2008). Redes Neuronales y Series de Fourier aplicados al procesamiento de imágenes en problemas de aprendizaje.

Gracia, L. M. (2012). ObjectAid UML Explorer for Eclipse. Retrieved from https://unpocodejava.wordpress.com/2012/02/08/objectaid-uml-explorer-for-eclipse/

Gualotuña, T., Ramírez, V., Jaramillo, V., & Campiña, M. (2014). *Sistema web con comunicación hacia dispositivos móviles para la gestión contable y tributaria con tecnología icefaces* (Informática). ESPE, Quito. Retrieved from http://repositorio.espe.edu.ec/bitstream/21000/5499/1/AC-SISTEMAS-ESPE-033591.pdf

Henao, D. (2009). Inteligencia artificial. Retrieved November 22, 2013, from http://site.ebrary.com/lib/utesp/docDetail.action?docID=10316885&p00=inteligencia%20artificial

Hidalgo, M. A., & Haj, M. (2014). Artículo Científico-Diseño y construcción de un robot móvil experimental 8x8 articulado para salvar obstáculos. Retrieved from http://repositorio.espe.edu.ec/handle/21000/7662

Hofmann, M., & Klinkenberg, R. (2013). *RapidMiner : Data Mining Use Cases and Business Analytics Applications*. Hoboken: Taylor and Francis.

Howse, J. (2013). *Android Application Programming with OpenCV*. Birmingham: Packt Publishing.

Huamán, A. (2015). OpenCV. Retrieved from http://docs.opencv.org/doc/tutorials/tutorials.html

Huete, J. F. (1998). *Sistemas expertos probabilísticos*. Univ de Castilla La Mancha.

Itakura, S., Ishida, H., Kanda, T., Shimada, Y., Ishiguro, H., & Kang Lee. (2008). How to Build an Intentional Android: Infants' Imitation of a Robot's Goal-Directed Actions. *Infancy*, *13*(5), 519–532. http://doi.org/10.1080/15250000802329503

Jain, A. K., Duin, R. P. W., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE IN TEL LIGENCE*, *22*(1), 4–37.

Jimenez, E. V. C., Navarro, D. Z., & Cisneros, M. A. P. (2010). *Procesamiento digital de imágenes con MATLAB y simulación*. RA-MA S.A. Editorial y Publicaciones.

Jurado, F., Asbusac, J., & EspaCursos. (n.d.). *Desarrollo de Videojuegos: Desarrollo de Componentes*. Cursos en Español.

Kaluza, B. (2013). *Instant Weka How-to* (1st ed.). Birmingham: Packt Publishing.

Kononenko, I., & Kukar, M. (2007). *Machine Learning and Data Mining*. Burlington: Elsevier Science.

Prototipado de aplicaciones moviles: las mejores herramientas. (n.d.). Retrieved from http://blog.aplicacionesmovil.com/aplicaciones-celular/herramientas-de-prototipado-de-aplicaciones-moviles/

Rairán, J. D., Chiquiza, D. F., & Parra, M. Á. (2012). Implementación de neurocontroladores en línea. Tres configuraciones, tres plantas. (Spanish). *The Implementation of On-Line Neurocontrollers: Three Configurations, Three Plants. (English)*, *16*(1), 163–182.

Rosales, L. P. P. (2004). Instrumentos de la Inteligencia Artificial para la adquisición y gestión del conocimiento. ¿capaces de apoyar el aprendizaje organizacional? (Spanish). *Administracion Y Organizaciones*, *7*(13), 53–67.

Rosario. (2014). El ranking de los 50 mejores robots de la historia, segun “wired.” Retrieved from http://www.tecnologiaparatodos.com.ar/robotica5.php

Sebe, N., Cohen, I., & Garg, A. (2006). *Machine Learning in Computer Vision* (1st ed.). Dordrecht: Springer.

Segura, B., Obregón, J., & Neira, N. (2005). Un modelo de lógica difusa y conjuntos difusos para el pronóstico de los niveles medios diarios del río Magdalena, en la estación limnigráfica de Puerto Salgar, Colombia. Retrieved from http://web.b.ebscohost.com/ehost/detail?sid=998188f4-9869-410b-85ad-d58145b24bda%40sessionmgr110&vid=6&hid=127&bdata=Jmxhbmc9ZXMmc2l0ZT1laG9zdC1saXZl#db=a9h&AN=20093939

Simonite, T. (2013, October 25). Google: Our Robot Cars Are Better Drivers Than Puny Humans. Retrieved April 8, 2014, from http://www.technologyreview.com/news/520746/data-shows-googles-robot-cars-are-smoother-safer-drivers-than-you-or-i/

Stanislawski, J., Kotulska, M., & Unold, O. (2013). Machine learning methods can replace 3D profile method in classification of amyloidogenic hexapeptides. *BMC Bioinformatics*, *14*(1), 1–9. http://doi.org/10.1186/1471-2105-14-21

The Univesity of Waikato. (2014, April 2). Weka 3- Data Mining with Open Source Machine Learning Software in Java. Retrieved from http://www.cs.waikato.ac.nz/ml/weka/

Thirumurugan, J., Vinoth, M., Kartheeswaran, G., & Vishwanathan, M. (2010). Line following robot for library inventory management system (pp. 1–3). IEEE. http://doi.org/10.1109/INTERACT.2010.5706151

Univision. (2013, November 13). Android es el sistema operativo móvil más usado en el mundo. Retrieved April 16, 2014, from http://noticias.univision.com/article/1738822/2013-11-13/tecnologia/noticias/android-es-el-sistema-operativo-movil-mas-usado-en-el-mundo?cmpid=FBshare:article

Vega, M. (2010). *Casos de uso UML*. Granada.

Vera, I. F. (2013). Clasificación de imágenes usando Weka. Retrieved from http://italoffvv.wordpress.com/2013/04/02/clasificacion-de-imagenes-usando-weka/

Wallisch, P., Lusignan, M. E., Benayoun, M. D., Baker, T. I., Dickey, A. S., & Hatsopoulos, N. G. (2014). *MATLAB for Neuroscientists : An Introduction to Scientific Computing in MATLAB* (2nd ed.). Burlington: Elsevier Science.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining : Practical Machine Learning Tools and Techniques* (3rd ed.). Burlington: Elsevier Science.

ANEXO 1

Manual de Instalación

MANUAL DE INSTALACIÓN

En el manual de instalación se incluyen los pasos para instalar la aplicación en un dispositivo móvil Samsung S4 mini, los pasos pueden variar dependiendo del modelo y la versión del dispositivo móvil.

El primer paso es pasar la aplicación al dispositivo móvil, puede ser mediante USB. (Ver Figura 79)

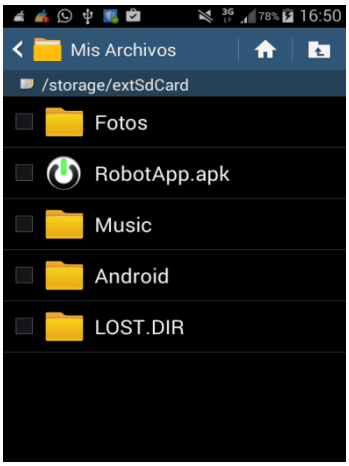

Figura 79: Pantalla del instalador de la aplicación

Pulse sobre el ícono RobotApp hasta obtener la pantalla que se indica en la Figura 80.

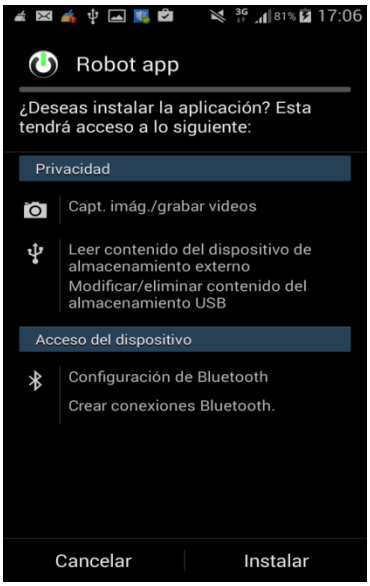

Figura 80: Pantalla de permisos de la aplicación

Pulse sobre Instalar y empezará con el proceso de instalación. Al finalizar se obtiene una pantalla como se indica en la Figura 81.

Figura 81: Pantalla de finalización

Pulse en Abrir, va a poder ver la pantalla principal de la aplicación. (Ver Figura 82)

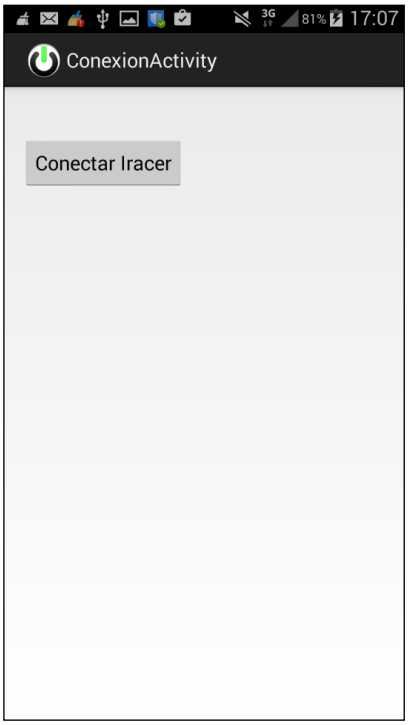

Figura 82: Pantalla inicio de aplicación

Presione en ConectarIracer, se le presentará un mensaje donde le pide instalar OpenCV-Manager como indica la Figura 83.

Figura 83: Pantalla de permisos OpenCV – Manager

Escoger **Yes**, y seguir el proceso hasta obtener la pantalla de instalación de OpenCV-Manager donde debe presionar en Instalar. (Ver Figura 84)

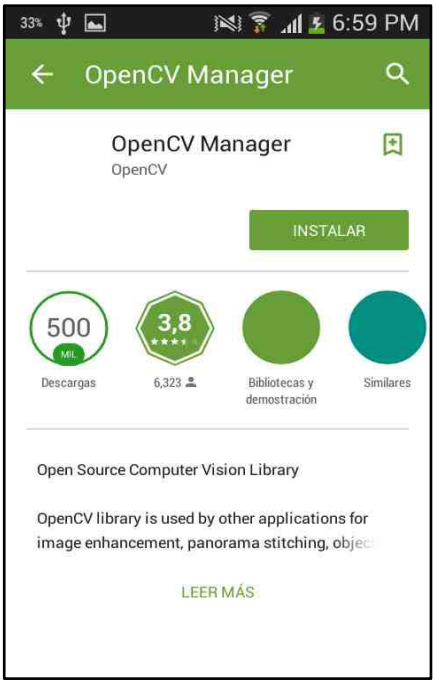

Figura 84: Pantalla de instalación OpenCV- Manager

En pantalla que se indica en la Figura 85, pulse en aceptar hasta que la descarga finalice.

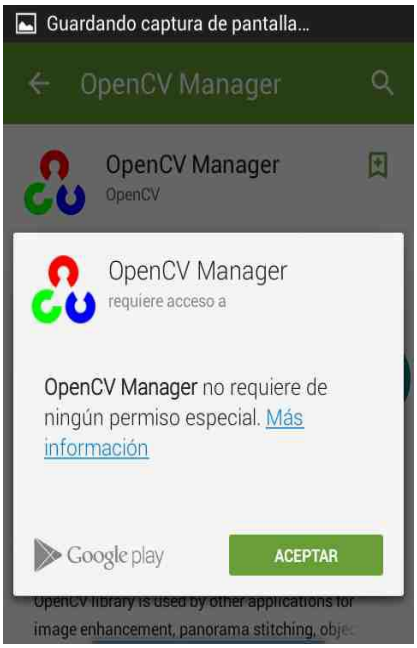

Figura 85: Pantalla aceptar condiciones OpenCV- Manager

Una vez finalizada la instalación se obtiene la pantalla como indica la Figura 86, presione en abrir.

Figura 86: Pantalla abrir OpenCV – Manager

Por último aparece la pantalla que se muestra en la Figura 87 acerca de la información de OpenCV- Manager, cierre y vuelva a ejecutar la aplicación.

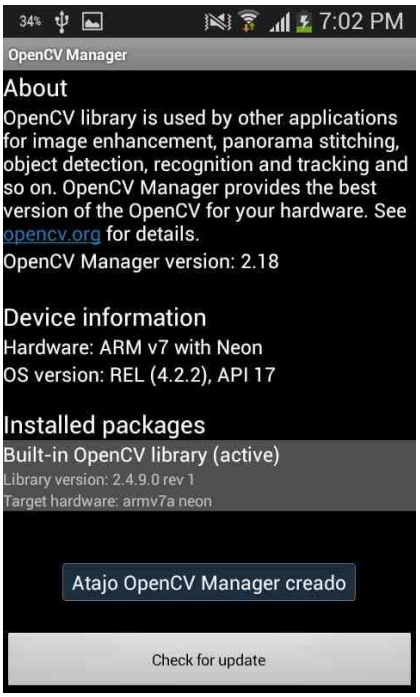

Figura 87: Información OpenCV - Manager