ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TITULO DE INGENIERO EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: "RECONOCIMIENTO DE GESTOS CORPORALES BASADO EN SVM Y SENSOR RGB-D PARA EL CONTROL DE MICRO VEHÍCULOS AÉREOS MULTIROTOR"

AUTOR:

COBEÑA ZAMBRANO, GEORGE BRYAN

DIRECTOR: DR. AGUILAR CASTILLO, WILBERT G.

SANGOLQUÍ

2018

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICADO

Certifico que el trabajo de titulación "**RECONOCIMIENTO DE GESTOS CORPORALES BASADO EN SVM Y SENSOR RGB-D PARA EL CONTROL DE MICRO VEHÍCULOS AÉREOS MULTIROTOR**" ha sido revisado en su totalidad y analizado por el software antiplagio el mismo cumple con los requisitos teóricos y científicos técnicos metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas - ESPE por lo tanto me permito acreditarlo y autorizarlo al señor **GEORGE BRYAN COBEÑA ZAMBRANO** para que lo sustente públicamente.

Sangolquí, 28 de Febrero de 2018

Ing. Wilbert G. Aguilar Ph.D.

DIRECTOR

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Yo, **GEORGE BRYAN COBEÑA ZAMBRANO,** con cédula de identidad 1718353863 declaro que este trabajo de titulación "**RECONOCIMIENTO DE GESTOS CORPORALES BASADO EN SVM Y SENSOR RGB-D PARA EL CONTROL DE MICRO VEHÍCULOS AÉREOS MULTIROTOR**", ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría en virtud de ello me declaro responsable del contenido veracidad y alcance de la investigación mencionada.

Sangolquí, 28 de Febrero de 2018

George Bryan Cobeña Zambrano

C.C. 1718353863

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Yo George Bryan Cobeña Zambrano autorizo a la Universidad de las Fuerzas Armadas Espe publicar en la biblioteca virtual de la institución el presente trabajo de titulación **"RECONOCIMIENTO DE GESTOS CORPORALES BASADO EN SVM Y SENSOR RGB-D PARA EL CONTROL DE MICRO VEHÍCULOS AÉREOS MULTIROTOR"**, cuyo contenido ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 28 de Febrero de 2018

George Bryan Cobeña Zambrano

C.C. 1718353863

DEDICATORIA

Este trabajo se lo dedico con todo cariño a mi padre Nery Cobeña y mi madre Oty Zambrano por haberme brindado su apoyo en todo momento, por su guía, su comprensión y los valores inculcados.

A mi hermana Carol Cobeña por ser mi confidente y saberme aconsejar en los momentos adecuados.

A mis compañeros de carrera con quienes hemos compartido buenas experiencias.

A mis amigos con quienes he compartido maravillosos momentos.

George Bryan Cobeña Zambrano.

AGRADECIMIENTO

Agradezco a Dios por guiarme, cuidarme y por su infinito amor.

A mis padres Nery y Oty que han sido un pilar fundamental en mi vida, que siempre han creído en mí, que me han inculcado valores para crecer como persona y de quienes me siento orgulloso de ser su hijo.

A mis amigos y compañeros por los momentos compartidos.

A los profesores que he tenido a lo largo de la carrera universitaria, quienes han sabido compartir sus conocimientos y formarme en vida profesional

Finalmente agradezco al Dr. Wilbert G. Aguilar por acompañarme como mi director de proyecto, por compartir sus conocimientos y recomendaciones para que este proyecto sea desarrollado de la mejor manera.

George Bryan Cobeña Zambrano.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto “Reconocimiento de gestos corporales basado en SVM y sensor RGB-D para el control de micro vehículos aéreos multirotor”, tiene como finalidad la integración de conocimientos adquiridos en la carrera de Ingeniería en Electrónica, Automatización y Control. Este proyecto tiene como finalidad permitir a cualquier persona, aun sin experiencia en el manejo de micro vehículos aéreos, controlar drones de manera intuitiva y natural. En la primera etapa del proyecto se realiza la extracción de características del cuerpo humano utilizando un sensor Kinect, con un algoritmo para la obtención de posiciones de varias articulaciones del cuerpo. La siguiente etapa consiste en el aprendizaje de maquina supervisado con SVM (Support Vector Machine) para la identificación de gestos en los que se basará el control del sistema, sistema que requiere de recolección de datos utilizando diferentes usuarios para generar los modelos de predicción. Una tercera etapa se enfoca en el desarrollo de una interfaz de usuario que conviertan las salidas del SVM en movimientos pre-programados además de contemplar la información necesaria y relevante para el control y manejo adecuado. En la etapa final del proyecto se realizan diferentes pruebas de funcionamiento y se comprar el sistema de reconocimiento de gestos frente al método tradicional de control de vuelo del drone.

Palabras clave:

- MÁQUINAS DE VECTORES SOPORTE (SVM)
- APRENDIZAJE DE MAQUINA SUPERVISADO
- SISTEMA OPERATIVO DE ROBÓTICA (ROS)
- MICRO VEHÍCULOS AÉROS MULTIROTOR
- SENSOR RGB-D

ABSTRACT

The following project "Body gesture recognition based on SVM and RGB-D sensor for the control of multirotor micro aerial vehicles" has as purpose, the integration of the knowledge acquired in the career of electronic, automation and control engineering. This project was develop in the robotics operating system ROS and has the finality of allowing any person, even without experience handling micro aerial vehicles, to control drones in a natural and intuitive way. In the first stage of the project, the feature extraction of the human body is made using a Kinect sensor; here an algorithm is implemented to attain the positions of several human body articulations. The second stage consists in supervised machine learning with SVM, in this stage gestures that will be used to control the system are identified, and data acquisition is made with different users to generate prediction models. The next stage of the project is focused on the development of a user interface, that converts signals from the SVM into pre-programmed movements, also it contemplates the necessary and relevant information for adequate handling and control. In the final stage of the project, different test runs are made, and the gesture recognition system is compared to the traditional flight control of the drone.

Keywords:

- **SUPPORT VECTOR MACHINE (SVM)**
- **SUPERVISED MACHINE LEARNING**
- **ROBOT OPERATING SYSTEM (ROS)**
- **MULTIROTOR MICRO AERIAL VEHICLES**
- **RGB-D SENSOR**

CAPÍTULO I

1 INTRODUCCIÓN

1.1 Antecedentes

La interacción existente entre el hombre y un computador (HCI) se ha convertido en una herramienta significativa para el desarrollo de diferentes aplicaciones como videojuegos (Kang, Woo, & Jung, 2004), reconocimiento de lenguaje de señas (Ren, Meng, & Yuan, 2011), tratamientos médicos (Gallo, Placitelli, & Ciampi, 2011) y otras aplicaciones que buscan bienestar y comodidad de las personas. Uno de los enfoques más utilizados de HCI es el reconocimiento de gestos corporales que consta de dos partes: extracción de características (front-end) y clasificación (back-end) (Kurakin, Zhang, & Liu, 2012).

Para la extracción de características, un tipo de sensor RGB-D que presenta un buen desempeño en el reconocimiento de gestos es el sensor Kinect (Shotton, Fitzgibbon, Cook, & Blake, 2011) (Ren, Yuan, Meng, & Zhang, 2013) (Aguilar W. G., Rodríguez, Álvarez, Sandoval, & Quisaguano, 2017) (2017). El Kinect es desarrollado por Microsoft y permite a usuarios controlar e interactuar con un sistema por medio de la captura de movimiento 3D de todo el cuerpo (Calle, Balseca, & Medina, 2015). Este sensor representa tecnología no invasiva, económica y usa un único sensor para la extracción de características del cuerpo humano (Ibañez, 2014).

Entre las aplicaciones logradas con el sensor Kinect se encuentran: El reconocimiento articulado del rostro (Cai, Gallup, Zhang, & Zhang, 2010), procesamiento de video 3D (Aguilar W. G., Rodríguez, Álvarez, Sandoval, & Quisaguano, 2017) (Aguilar & Morales, 2017) (Aguilar W. G., Morales, Ruiz, & Abad, 2017) (Ho & Joc, 2013), la identificación de articulaciones del cuerpo humano (Le, Nguyen, & Nguyen, 2013), detección de bordes (Lejune, Piérard, Droogenbroeck, & Verly, 2011) y la operación de robots (Afthoni, Rizal, & Susanto, 2013) (Jiménez, 2015).

Para la clasificación se han propuesto distintos algoritmos, entre los cuales se pueden mencionar:

- K-nn (K nearest neighbors), el cual es un método de clasificación supervisada basado en la cercanía que se obtiene a partir de las consultas realizadas sobre la métrica de distancia de los objetos. La desventaja que presenta este método es el lento aprendizaje (Weinberfer & Lawrence, 2009).
- El modelo de Hidden Markov que es un proceso estocástico con un número finito de estados y funciones aleatorias donde el proceso permanece en un estado y la función aleatoria determina el siguiente estado (Ding & Chang, 2015). La desventaja que presenta este método es que requiere gran cantidad de datos de entrenamiento y parámetros de configuración (Kurakin, Zhang, & Liu, 2012).
- Otro método utilizado para la clasificación es el uso de SVM (Maquinas de vectores de soporte) que basa su principio en la creación de hiperplanos para separar conjuntos que no son linealmente separables (Carmona, 2014).

El método basado en el uso de SVM presenta algunas ventajas ante los otros métodos mencionados anteriormente como son: una mayor precisión en la predicción, eficiencia en el uso de memoria, robustez ante los errores que se presente en el entrenamiento y eficacia en espacios de alta dimensión (Kurakin, Zhang, & Liu, 2012) (Kurtulmus & Kavdir, 2014).

ROS (Sistema operativo de robótica) representa una excelente alternativa que permite el desarrollo de código abierto, además en ROS, se puede acceder a librerías y herramientas específicas de manera gratuita para aplicaciones de robótica (Quigley, Gerkey, Conley, & Faust, 2010).

Tomando en cuenta los antecedentes mencionados y con el objetivo de superar las dificultades que representa el reconocimiento de gestos corporales, se propone en este proyecto de investigación desarrollar un algoritmo basado en máquinas de vectores de soporte, además del uso de un sensor RBG-D con el fin de realizar la identificación de diferentes gestos corporales para el control de desplazamiento de un micro vehículo aéreo multirotor, montado sobre ROS.

1.2 Justificación e Importancia

La interacción entre el hombre y la computadora (HCI) por más de 40 años se vio limitada al uso exclusivo del teclado y el mouse, lo que lo hace un modelo muy restringido como modelo de interacción (Kang, Woo, & Jung, 2004). Con la tecnológica en constante evolución, aparece el reconocimiento de gestos corporales como método de HCI natural e intuitivo para el control de dispositivos (Aguilar & Angulo, 2014) (2014) (2012) (2016). Los gestos son una habilidad que todos los humanos podemos realizar sin que sea necesario una gran capacidad de análisis y pensamiento.

El proyecto de investigación se desarrolla para el control de micro vehículos aéreos no tripulados (UAV), también conocidos como drones, debido a que han adquirido gran popularidad en los últimos años por la autonomía, y alto grado de aplicación en diversas actividades como: Detección de objetos, vigilancia, recolección de datos, detección de fuego, seguimiento y planificación de trayectorias, entre otras (Chmaj & Selvaraj, 2015) (Aguilar W. G., y otros, 2017) (2017) (2017) (Aguilar & Angulo, 2014) (Aguilar W. G., Morales, Ruiz, & Abad, RRT* GL based optimal path planning for real-time navigation of UAVs, 2017).

Pese a la gran cantidad de aplicaciones que tienen los drones, muchas personas por falta de experiencia no pueden aprovecharlas al no estar familiarizadas con las interfaces de control que muchas veces son complejas y de difícil manipulación.

Este trabajo sobre reconocimiento de gestos corporales se desarrolla con el fin de permitir a cualquier persona, aún sin experiencia en el manejo de micro vehículos aéreos multirotor, controlar drones de manera intuitiva y natural.

1.3 Alcance del Proyecto

El presente proyecto de investigación tiene como objetivo el reconocimiento de gestos corporales, para controlar el desplazamiento de un micro vehículo aéreo

multirotor en cuatro grados de libertad (arriba, abajo, adelante, atrás, derecha e izquierda). El sistema estará montado sobre el Sistema Operativo de Robótica (ROS) y para la adquisición de las señales se utilizara Visión por Computador y el aprendizaje de maquina con un sensor RGB-D.

El trabajo se divide en cuatro etapas:

En la primera etapa del proyecto, se realizará un estudio de las técnicas utilizadas para la detección de puntos de referencia del cuerpo humano. Posteriormente se desarrollará una interfaz para captura de datos mediante el sensor RGB-D que permita reconocer puntos de articulaciones y su respectiva posición en el espacio.

La segunda etapa del proyecto consiste en la clasificación de patrones e identificación de gestos corporales. Estos gestos identificados se usarán para el control del micro vehículo aéreo. Se desarrollará un algoritmo utilizando Maquinas de vectores de soporte (SVM) y las pruebas se realizaran con cinco personas diferentes.

La siguiente etapa del proyecto se enfoca en una interfaz intuitiva y fácil de utilizar para el usuario donde se conviertan las señales del SVM en movimientos pre-programados, en la cual se contemple toda la información necesaria y relevante para el control y manejo adecuado. Adicionalmente, en esta etapa se realizará la comunicación mediante wifi de la interfaz con el micro vehículo aéreo.

La etapa final del proyecto se enfoca en la realización de pruebas de funcionamiento que busca analizar la precisión y velocidad de respuesta del sistema realizando pruebas en cinco personas diferentes. Se realizaran pruebas en las cuales se compare el sistema de reconocimiento de gestos corporales frente a los sistemas tradicionales para controlar un drone.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un sistema para el reconocimiento de gestos corporales mediante el uso de SVM y un sensor RGB-D para el control de un micro vehículo aéreo multirotor.

1.4.2 Objetivos Específicos

- Determinar la posición de puntos de referencia del cuerpo humano basados en algoritmos de extracción de características.
- Clasificar gestos corporales mediante el entrenamiento y aprendizaje de maquina supervisado.
- Diseñar una interfaz de usuario intuitiva y de fácil manejo para poder controlar el sistema.
- Realizar la comunicación del sistema de reconocimiento de gestos corporales con el micro vehículo aéreo multirotor.
- Contrastar los resultados del sistema de reconocimiento de gestos corporales frente a un sistema tradicional de vuelo, mediante pruebas experimentales.

CAPÍTULO II

2 FUNDAMENTACIÓN TEÓRICA

Con el fin de sustentar teóricamente las herramientas y conceptos a utilizar en la investigación para el desarrollo del Sistema de Reconocimiento de Gestos Corporales mediante SVM y Sensor RGB-D en el presente capítulo se realiza un análisis y descripción de los sistemas de aprendizaje de maquina supervisados, máquinas de soporte vectorial y los gestos corporales. Finalmente, se realiza una descripción del software y herramientas a utilizar tales como: Kinect, Sistema operativo de robótica (ROS) y Parrot bebop drone.

2.1 Aprendizaje de Máquina

Los algoritmos de aprendizaje de maquina son técnicas para el análisis de datos, de modo que, un computador pueda hacer lo que resulta natural para un ser humano. Estos algoritmos emplean complejos métodos de cálculo para generar modelos que caractericen el comportamiento de un conjunto de datos y mejoran su rendimiento a medida que aumenta el número de muestras para el aprendizaje. Al manejar grandes cantidades de datos son ideales para resolver problemas como:

- Finanzas computacionales
- Procesamiento de imágenes y visión artificial
- Biología computacional
- Producción de energía
- Procesamiento del lenguaje natural (Aprendizaje automático, 2017).

El aprendizaje de máquina en su clasificación presenta dos grupos principales: El aprendizaje de maquina supervisado el cual se genera con un modelo con datos de entrada y salidas conocidos y que ayudan a predecir futuras salidas, y el aprendizaje de maquina no supervisado, el cual, a partir de datos de entrada encuentra patrones para un agrupamiento.

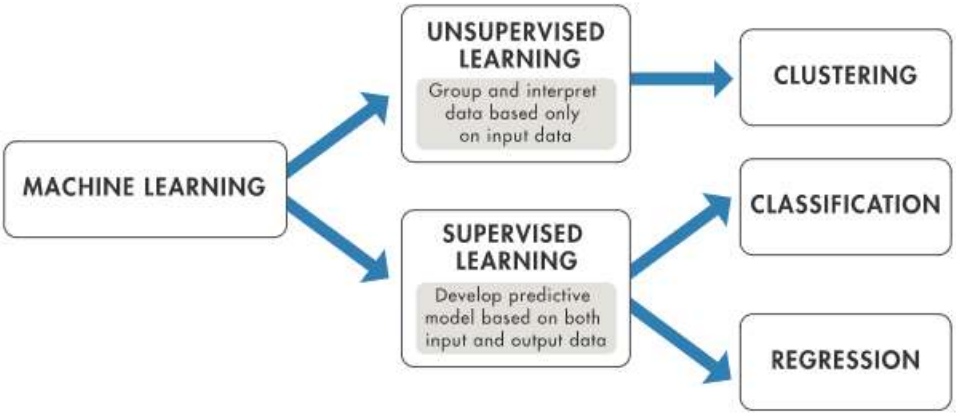


Figura 1. Clasificación del aprendizaje de máquina
Fuente: (Aprendizaje automático, 2017)

2.1.1 Aprendizaje de máquina supervisado

El aprendizaje de maquina supervisado se utiliza en problemas de clasificación o regresión, para lo cual se genera un modelo que realiza predicciones de un conjunto nuevo de datos. Para la creación del modelo se utiliza un conjunto de datos conocidos de entrada y salida, además de un algoritmo de entrenamiento. Los algoritmos que se pueden utilizar son varios como se muestra en la Figura 2.

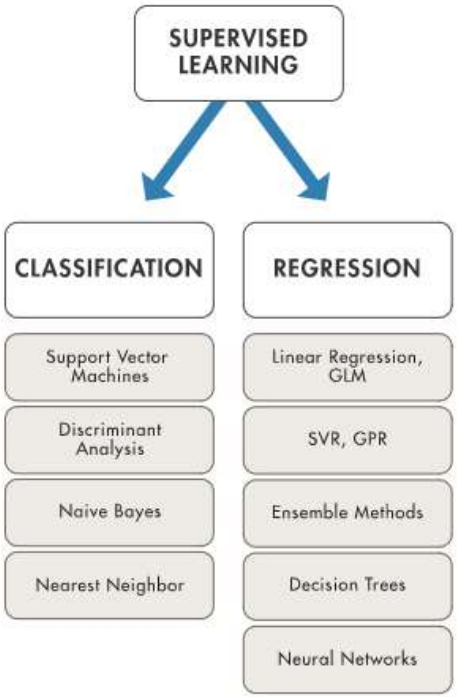


Figura 2. Algoritmos de aprendizaje de máquina supervisado
Fuente: (Aprendizaje automático, 2017)

2.2 Máquinas de Vectores Soporte (SVM)

Las máquinas de vectores soporte se originan en la teoría de aprendizaje estadístico en los años 90 por Cortes y Vapnik (Cortes & Vapnik, 1995). Este método en sus principios fue pensado y desarrollado para realizar clasificaciones binarias, pero debido a su alto potencial, en la actualidad se utiliza para la resolución de otro tipo de problemas como son: regresiones, agrupamiento y multiclasificación. Las SVM se utilizan en aplicaciones relacionadas con la interacción humano-máquina, además en campos como la visión artificial y reconocimiento de caracteres (Carmona, 2014).

El algoritmo de las SVM se utilizan para problemas de clasificación y regresión en el aprendizaje de maquina supervisado, donde se construye un hiperplano de alta dimensión para realizar la separación de conjuntos de datos. La selección del hiperplano se realiza aplicando el concepto de margen máximo, es decir, se busca un hiperplano ideal que equidiste de todos los conjuntos a separar.

2.2.1 Funcionamiento de los SVM

La mayoría de los métodos de aprendizaje de máquina supervisados se enfocan en minimizar los errores de modelo generado, a partir de datos de entrenamiento o lo que se conoce como error empírico, es por ello que a la hora de realizar la separación de un conjunto de datos se puede tener un número infinitos de hiperplanos de solución (ver Figura 3).

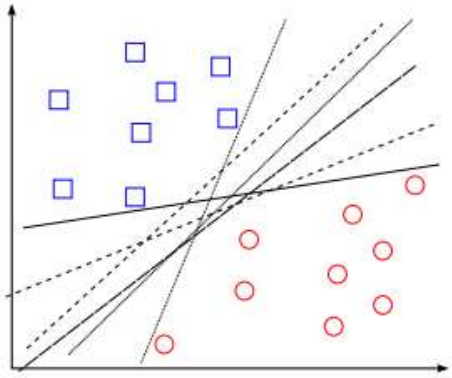


Figura 3. Hiperplanos de solución para separación de un espacio bidimensional
Fuente: (Carmona, 2014)

Las SVM se centran en la minimización del riesgo estructural, donde se busca el hiperplano que separe de forma óptima los puntos de las distintas clases. Esta selección se la hace al seleccionar el hiperplano que equidista de los ejemplos más cercanos de cada clase para de esta forma conseguir el hiperplano de margen máximo.

Es decir, dado un conjunto de datos separables $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$, donde $x_i \in \mathbb{R}^d$ y $y_i \in \{+1, -1\}$, se puede definir como una función lineal un hiperplano de separación sin error de la forma:

$$D(x) = (w_1 x_1 + \cdots + w_d x_d) + b = < w, x > + b \tag{1}$$

Donde w y b son coeficientes reales. El hiperpalno de separación cumplirá las siguientes restricciones para todo x_i:

$$y_i D(x_i) \geq 0, \qquad i = 1, \dots, n \tag{2}$$

La distancia de un hiperplano $D(x)$ y un conjunto de datos x' viene dada por

$$\frac{|D(x')|}{||w||} \tag{3}$$

Utilizando las ecuaciones (2) y (3), todos los ejemplos de entrenamiento deben cumplir con

$$\frac{y_i D(x')}{||w||} \geq \tau, \qquad i = 1, \dots, n \tag{4}$$

Para limitar el número de soluciones a una sola, la escala del producto τ y $||w||$ será la unidad, es decir

$$\tau \left|\left|\mathrm{w}\right|\right| = 1 \tag{5}$$

Donde es claro que al aumentar el margen se disminuirá la norma w por la relación de proporcionalidad inversa

$$\tau = \frac{1}{||w||} \tag{6}$$

Por lo tanto, el hiperplano de separación optimo (ver Figura 4) es aquel que posee el margen máximo, lo que a su vez significa un valor mínimo de $||w||$, además de, cumplir con la siguiente restricción:

$$y_i(< w, x_i > +b) \geq 1, \qquad i = 1, \dots, n \tag{7}$$

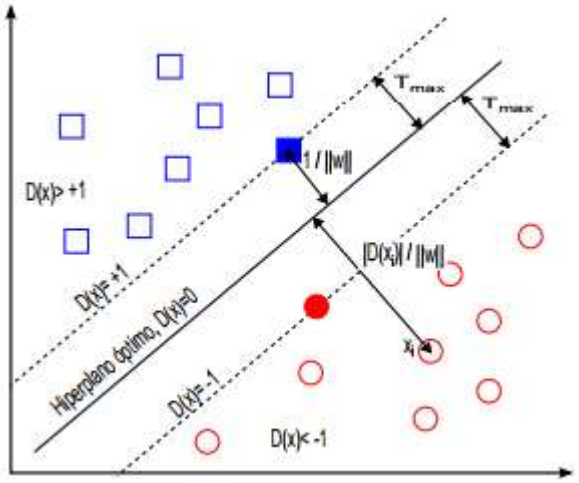


Figura 4. Hiperplano óptimo de separación
Fuente: (Carmona, 2014)

2.2.2 SVM con margen blando

Debido a que los problemas en la realidad tienden a no ser perfectamente separables linealmente como se muestra en la Figura 5, y con el fin de dar flexibilidad, los SVM agregan un parámetro de costo C el cual permite algunos errores en la clasificación por medio de las variables de holgura ξ.

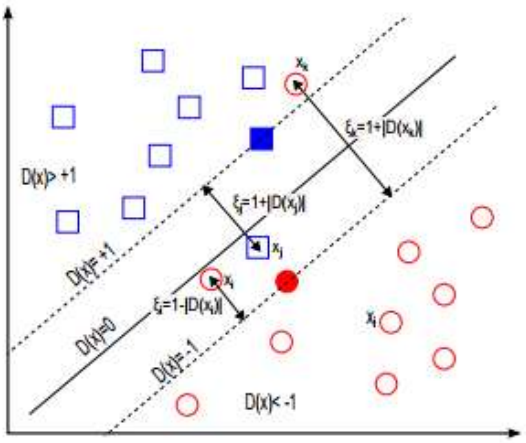


Figura 5. Variables de holgura para obtención de hiperplano óptimo
Fuente: (Carmona, 2014)

Por tanto, la función a optimizar será:

$$f(w,\xi) = \frac{1}{2}\left||w|\right|^2 + C\sum_{i=1}^{n}\xi_i \quad (8)$$

Donde C es una constante elegida por el usuario para controlar el grado de coste en la minimización de la norma. Al utilizar un coste C de un valor alto cuando los datos a clasificar son casi perfectamente separables, y cuando exista un numero alto de datos mal clasificados es necesaria una mayor holgura ξ por lo cual se utiliza un valor pequeño en el coste C.

2.2.3 SVM para datos no separables linealmente

Cuando los datos no se pueden separar linealmente es necesario utilizar una función kernel, la cual se encarga de realizar un cambio de espacio aumentando la dimensionalidad de los vectores de entrada a un espacio en el cual puedan ser linealmente separables como se ve en la Figura 6 (Estrada & Mera, 2017).

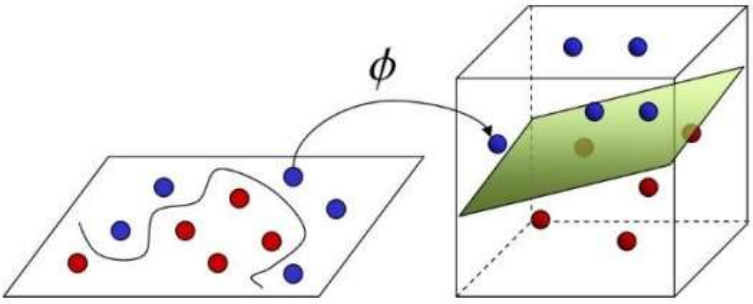


Figura 6. Aumento de la dimensional dad a los vectores entrada
Fuente: (Estrada & Mera, 2017)

Existen diferentes tipos de kernel, entre ellos se tiene:

- Kernel Lineal:

$$K(x,x') = < x,x' > \quad (9)$$

- Kernel polinómico de grado $-p$:

$$K_p(x,x') = [\gamma < x,x' > +\tau\ ]^p \quad (10)$$

- Kernel gaussiano o en función de base radial (RBF):

$$K(x, x') = \exp\left(-\gamma \left||x - x'|\right|^2\right), \quad \gamma > 0 \tag{11}$$

- Kernel sigmoidal:

$$K_p(x, x') = \tanh\left(\gamma < x, x' > + \tau\right) \tag{12}$$

2.3 Gestos corporales

Los gestos se realizan con diferentes partes del cuerpo humano con el fin de expresar, emociones, sentimientos o realizar acciones. Los gestos resultan ser intuitivos y naturales para toda persona con la capacidad de análisis para realizar movimientos. Existe una infinidad de gestos y cada uno tiene ciertas particularidades que lo caracterizan, es por ello que, su estudio involucra varias disciplinas como son la psicología, comunicación, expresión corporal, actuación, danza y la computación.

2.3.1 Clasificación de los gestos

Según (Mcneill, 1992) analizando desde un punto de vista comunicacional, los gestos pueden clasificarse en cuatro tipos principales que son: icónicos, metafóricos, ilustrativos y deícticos (ver Figura 7). Esta distinción se enfoca en el estudio de los gestos como parte del habla, sin embargo, su distinción es relevante al uso de gestos en la interacción humano máquina (Quiroga, 2014).

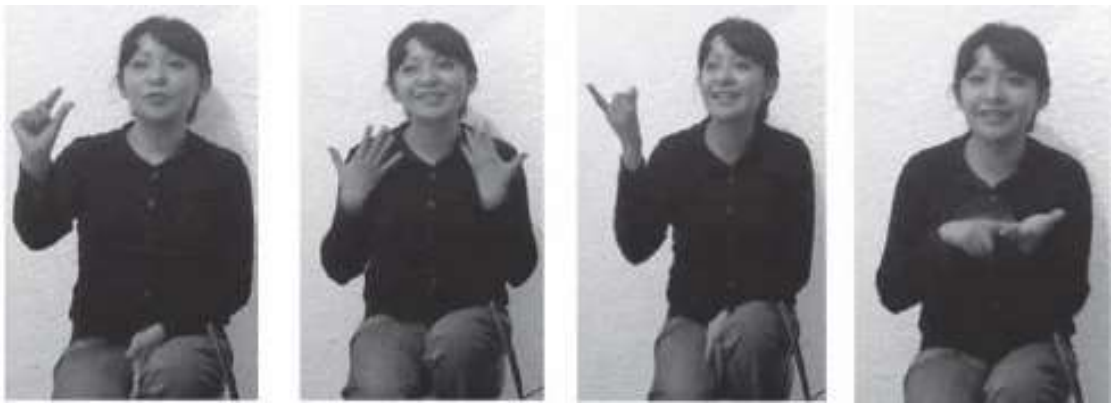


Figura 7. De izquierda a derecha, gestos icónicos, metafóricos, deícticos e ilustrativos

Fuente: (Aguilar & Galicia, 2016)

2.3.1.1 Gestos icónicos

Este tipo de gestos también conocidos como gestos de lo concreto, representan algún objeto y sus características como su tamaño, forma, orientación, entre otras. Por lo que, resultan ser fáciles de identificar y entender.

2.3.1.2 Gestos metafóricos

Este tipo de gestos representan ideas abstractas o conceptos, por lo que se utiliza comúnmente en el lenguaje de señas.

2.3.1.3 Gestos ilustrativos

Los gestos ilustrativos acompañan a la forma de hablar con el fin de recalcar o dar un matiz a lo que se está expresando, es por ello que suelen ser movimientos simples donde las manos se mueven de forma rítmica con lo que se está hablando.

2.3.1.4 Gestos deícticos

Son gestos para señalar o seleccionar objetos. Este tipo de gestos cumplen con una importante función comunicativa cuando se inicia el aprendizaje del lenguaje por lo por lo que son los más intuitivos. Este tipo de gestos se puede utilizar en la interacción humano computador, ya que pueden representar acciones como el moverse, salir, entrar, adelante, atrás, entre otras (Quiroga, 2014).

2.4 Kinect

El sensor Kinect es un dispositivo desarrollado por Microsoft inicialmente para la consola de juegos Xbox 360. A partir de junio del 2011 está disponible para PC en Windows 7 y versiones posteriores con el objetivo de lograr que un mayor número de desarrolladores de software tengan alcance a esta tecnología. Este sensor se puede utilizar para desarrollar aplicaciones de interacción natural como la extracción de características del esqueleto humano y el reconocimiento de voz (Ñecato, 2014).

2.4.1 Características técnicas

Entre los componentes principales del sensor Kinect se tiene un sensor de profundidad, una cámara RGB, arreglo de micrófonos y un sensor infrarrojo (Ver Figura 8) los cuales al trabajar en conjunto permiten realizar la captura de movimientos de todo el cuerpo humano, así como su posicionamiento en el espacio.

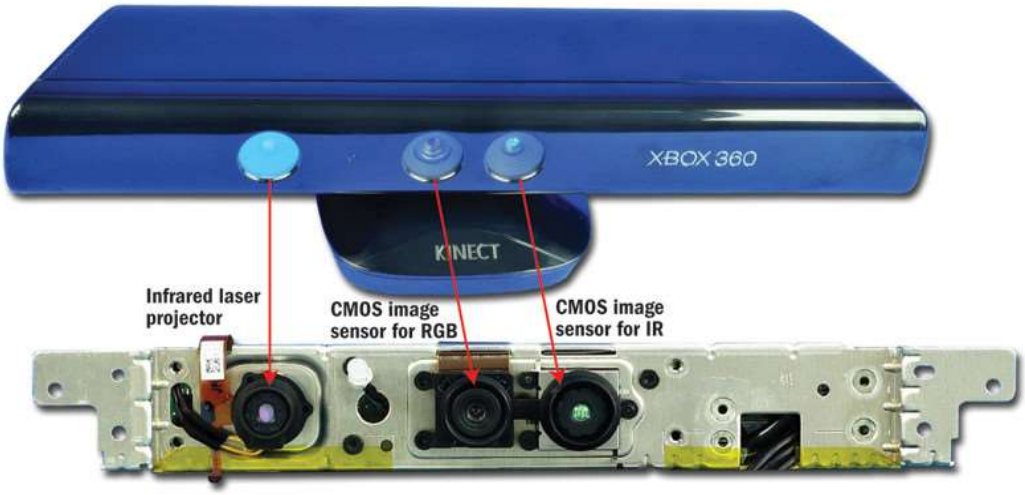


Figura 8. Componentes del sensor Kinect
Fuente: (Widenhofer, 2010)

2.4.1.1 Sensor de Profundidad

Dicho sensor está compuesto por un emisor infrarrojo y un sensor CMOS monocromático y permite la captura de datos independientemente del grado de iluminación; este sensor opera con una resolución de 320x240 pixeles con 16 bits de profundidad a 30 fps (imágenes por segundo) (Sandoval & Trujillo, 2016).

2.4.1.2 Cámara RGB

La cámara RGB en sus siglas en inglés, *Red, Green & Blue* (rojo, verde y azul), posee una resolución de 640x480 pixeles con 32 bits de color y puede transmitir imágenes o videos a una tasa de 30fps (Ñecato, 2014).

2.4.1.3 Arreglo de micrófonos

El arreglo de micrófonos integrado en el sensor Kinect se utiliza en aplicaciones de reconocimiento de voz y está formado por cuatro micrófonos que trabajan de forma independiente con la capacidad de muestrear a una frecuencia de 16KHz.

2.4.1.4 Campo de visión

El campo de visión del sensor Kinect abarca un área de trabajo con las siguientes dimensiones:

- Ángulo de visión horizontal: 57 grados (ver Figura 9).
- Ángulo de visión vertical: 43 grados (ver Figura 10).
- Rango de inclinación: ± 27 grados.
- Rango del sensor de profundidad: 1,2 – 3,5 metros.

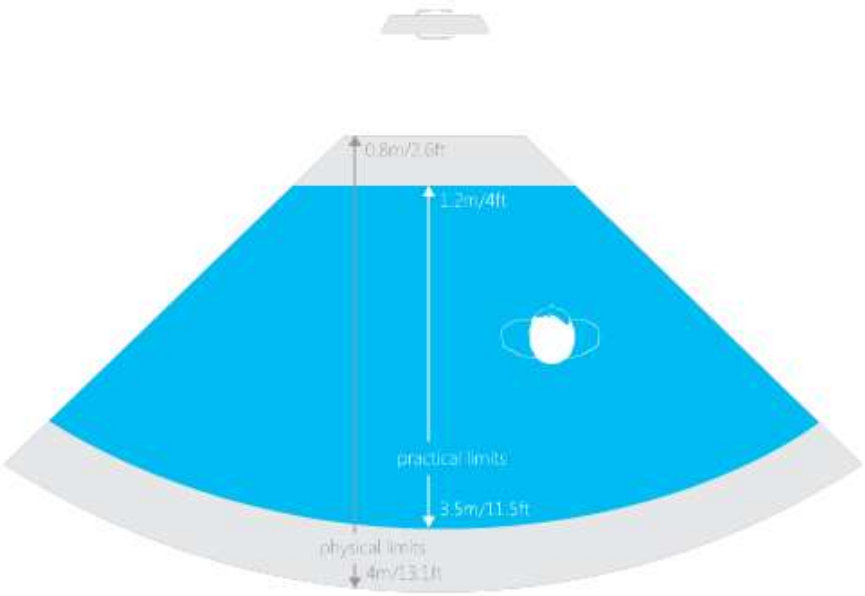


Figura 9. Campo de visión horizontal del sensor Kinect
Fuente: (Microsoft, 2017)

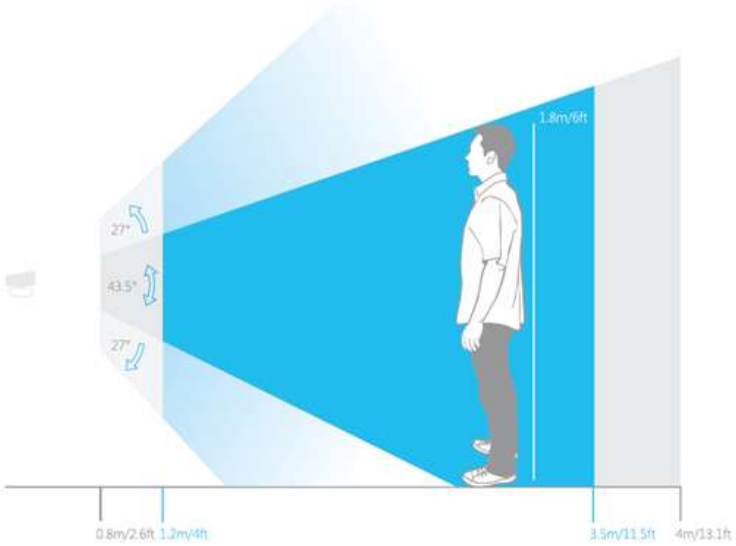


Figura 10. Campo de visión vertical del sensor Kinect
Fuente: (Microsoft, 2017)

2.4.2 Posición del esqueleto humano

Mediante el sensor de profundidad y la cámara RGB integrados en el sensor Kinect se puede realizar el posicionamiento del esqueleto humano en tiempo real, es decir, se obtiene información que incluye la ubicación en el espacio y la rotación en la que se encuentran diferentes articulaciones del cuerpo humano (Ver Figura 11).

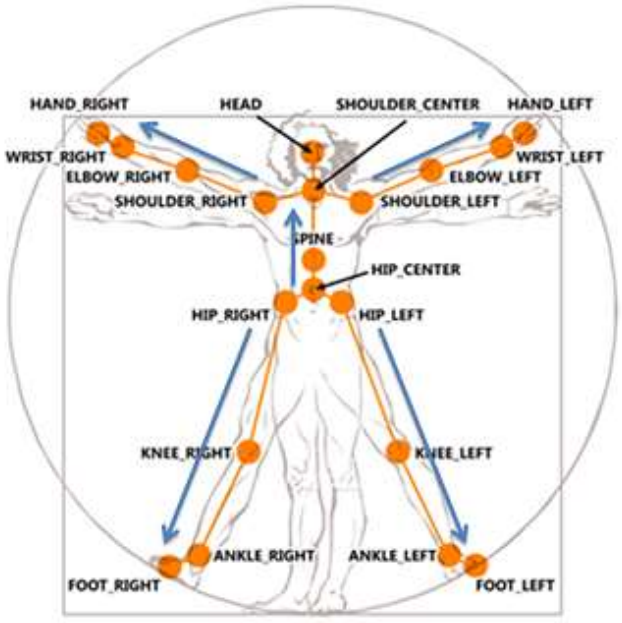


Figura 11. Posición de articulaciones del cuerpo humano
Fuente: (Microsoft, 2017)

Para la obtención de la pose el sensor Kinect tiene un sistema de referencia de coordenadas cartesianas que abarcan un espacio tridimensional, es decir, cada articulación proporciona datos en las coordenadas en los ejes x, y, z. Las coordenadas tienen como punto de referencia el sensor Kinect que presenta una distribución de sus ejes cono se muestra en la Figura 5.

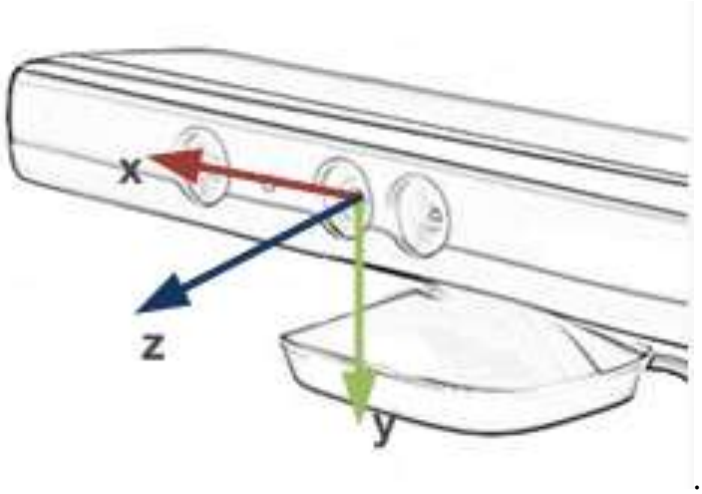


Figura 12. Sistema de coordenadas del sensor Kinect
Fuente: (Integración de dispositivos Leap y Kinect, 2013)

2.5 Sistema Operativo de Robótica (ROS)

ROS es un entorno de trabajo que nace en 2007 como un proyecto del Laboratorio del Inteligencia Artificial de Stanford y el cual estructuralmente está desarrollado para proporcionar un entorno adecuado para el desarrollo de aplicaciones de robótica bajo licencia Open Source, de modo que, se pueda simplificar la tarea de controlar el complejo comportamiento de una gran variedad de plataformas robóticas (Erle Robotics, 2017).

Debido a la popularidad que ha ido adquiriendo este sistema operativo por su arquitectura flexible y adaptable, la cual, permite la comunicación de múltiples nodos sin que sea necesario estar en un mismo sistema, actualmente consta con una variedad creciente de librerías, paquetes, herramientas de visualización, comunicación de mensajes y administración de paquetes para ayudar en el desarrollo de aplicaciones para robots (ROS Documentation, 2017).

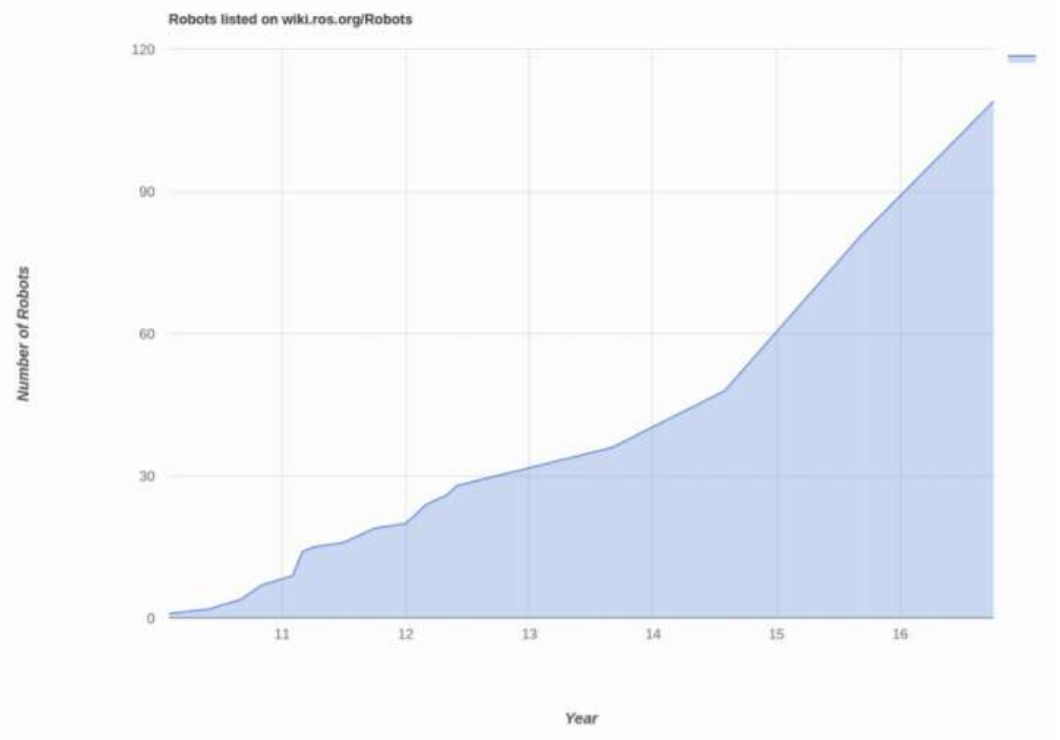


Figura 13. Robots disponibles para entorno de trabajo ROS
Fuente: (Conley & Foote, 2017)

2.5.1 Conceptos Básicos

Para comprender como funciona este sistema operativo es necesario conocer los conceptos básicos de los principales elementos que conforman un sistema en ROS. A continuación se realiza una breve descripción de cada uno de estos.

2.5.1.1 Maestro ROS

También conocido como "Roscore" funciona como núcleo del sistema operativo de ROS, es decir, al realizar su ejecución habilita los canales de comunicación de modo que los nodos puedan intercambiar información entre ellos, por lo cual, siempre es necesario realizar su ejecución de manera inicial (ver Figura 14).

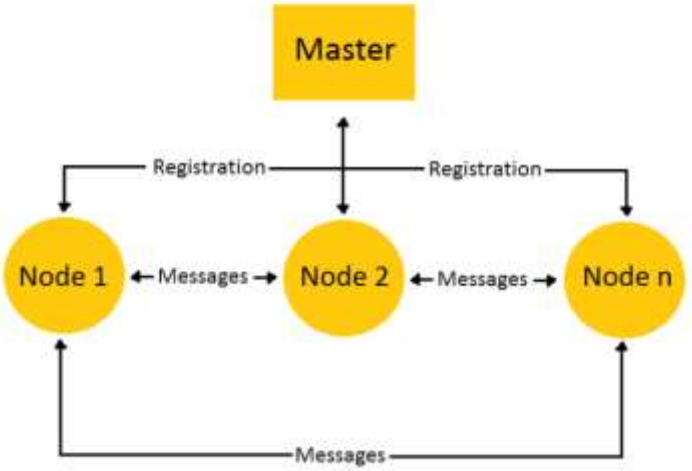


Figura 14. Funcionamiento del Maestro ROS
Fuente: (Erle Robotics, 2017)

2.5.1.2 Nodos

Un nodo es un proceso ejecutable el cual se desarrolla dentro de un paquete ROS, es por ello, que un nodo va a contener servicios, tópicos y para su construcción se utilizan librerías denominadas clientes, las cuales pueden ser desarrolladas en diferentes lenguajes de programación como C++ (roscpp) y python (rospy).

2.5.1.3 Mensajes

Los mensajes son los datos que son publicados en los tópicos para que los nodos se puedan comunicar entre sí y estos pueden ser de diferente tipo como: enteros, flotantes, boléanos y cadenas de texto, además, existe la opción para crear tipo de datos personalizados dependiendo la aplicación a desarrollar.

2.5.1.4 Tópicos

Un tópico es el canal de comunicación que va a existir para que dos o más nodos puedan comunicarse, es decir, en un tópico se publican los mensajes y para recibirlos es necesario suscribirse a este tópico, por lo cual, es necesario asignar una etiqueta de identificación que describa el tipo de mensaje que se está transmitiendo.

2.5.1.5 Paquete

El paquete representa la principal unidad de organización dentro del sistema operativo ROS ya que este contiene nodos y tópicos para poder desarrollar una aplicación (ver Figura 15).

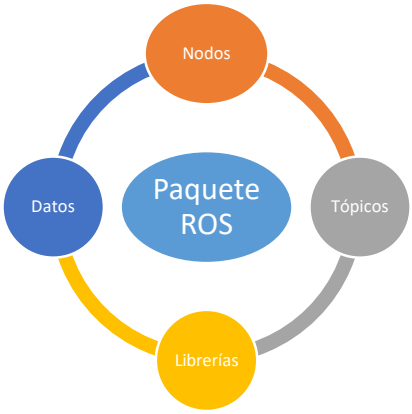


Figura 15. Estructura de un paquete ROS

2.5.1.6 Pila

La pila o también conocida como “stack” es un conjunto de paquetes los cuales al trabajar en conjunto brindan una funcionalidad (ver Figura 16).

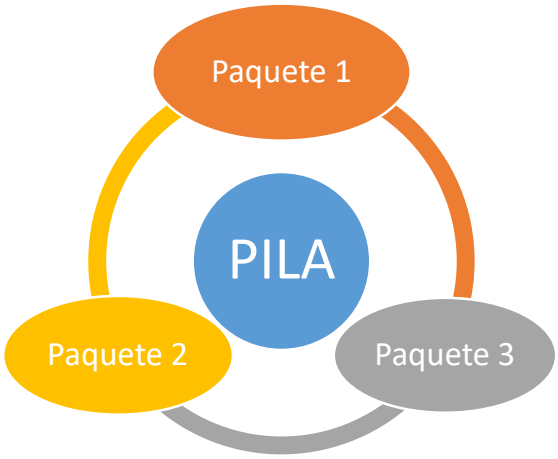


Figura 16. Estructura de la pila en ROS

2.6 Vehículo aéreo no tripulado

Un vehículo aéreo no tripulado también conocido como UAV por sus siglas en inglés *Unmanned Aerial Vehicle*, es un sistema capaz de volar sin la necesidad de tener a bordo un piloto, por lo que, en la plataforma de vuelo tiene incorporados procesadores, sensores, y un sistema de comunicación para ser controlado remotamente desde una estación de mando en tierra (Gonzáles, 2017).

2.6.1 Clasificación de los UAV

Según la Organización del Tratado del Atlántico Norte u OTAN los UAV se clasifican en tres clases dependiendo su tamaño, capacidad de vuelo, altura y alcance (ver Figura 17).

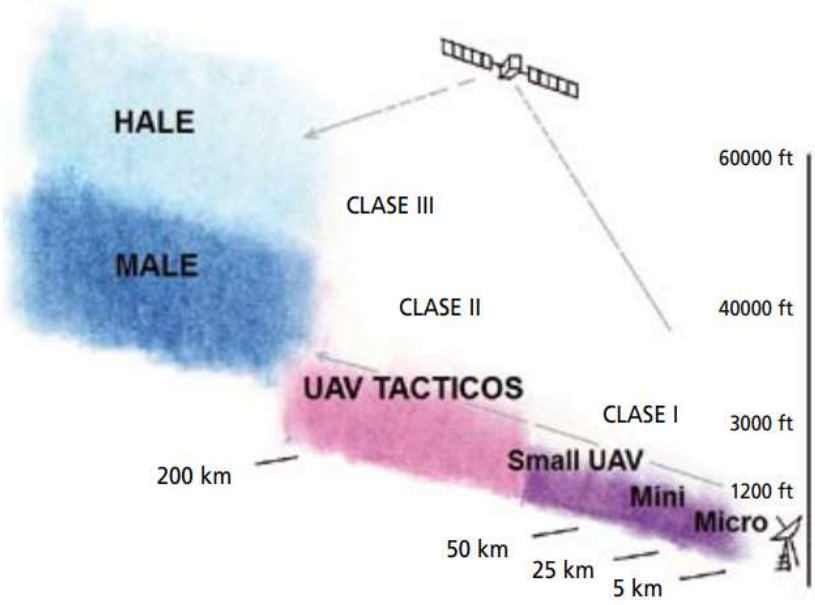


Figura 17. Clases de UAV según la OTAN
Fuente: (Real, 2013)

Otra forma de clasificar los UAV se basa en el mecanismo de despegue; en este tipo de clasificación se tiene los UAV de despegue vertical y no vertical como se muestra en la Figura 18.

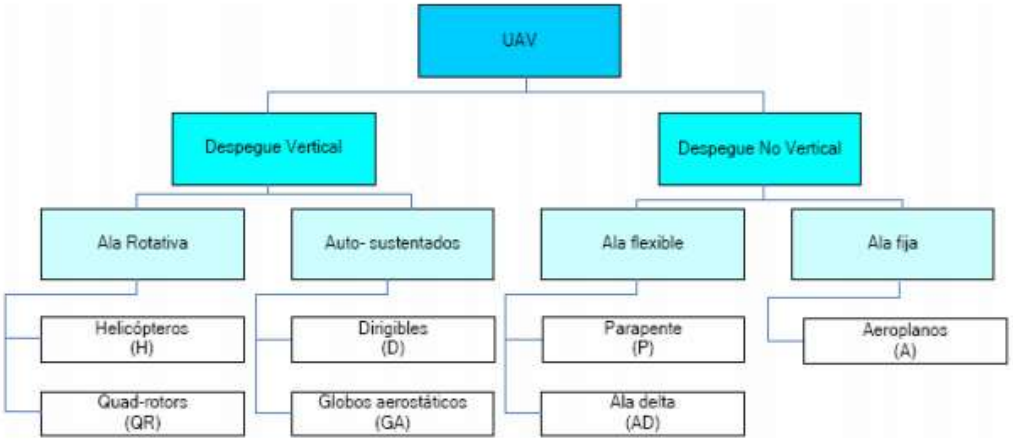


Figura 18. Clases de UAV según la OTAN
Fuente: (Mayorga, 2009)

2.6.2 Ventajas y desventajas

Según los tipos de UAV de despegue vertical y no vertical, en la Tabla 1 se presentan las principales ventajas y desventajas.

Tabla 1.
Tabla de Ventajas y Desventajas de UAV según su tipo de despegue

Configuración	Ventajas	Desventajas
Ala Rotativa	• Fácil Manejo • Mecanismo de baja complejidad • Gran autonomía • Capaces de permanecer inmóviles en el aire	• Reducción en su velocidad • Estructura de mayor complejidad
Ala Fija	• Gran capacidad de vuelo • Alcanzan grandes velocidades	• Necesita siempre una pista para el despegue y aterrizaje
Auto Sustentados	• Bajo consumo de energía	• Velocidades muy pequeñas

2.6.3 Aplicaciones de los UAV

De acuerdo con (Chmaj & Selvaraj, 2015) , existe un gran número de aplicaciones en las que se utilizan los UAV debido a clasificación y versatilidad. Entre las aplicaciones más importantes se puede nombrar:

- Detección de Objetos
- Seguimiento y evasión de obstáculos (Aguilar, Casaliglla, & Polit, 2017) (2017) (2017)
- Vigilancia (Aguilar, Salcedo, Sandoval, & Cobeña, 2017)
- Recolección de datos
- Planificación de trayectorias
- Navegación
- Prevención de colisiones
- Monitoreo Ambiental.

2.7 Cuadricópteros UAV

El cuadricóptero es un tipo de UAV de despegue vertical y de ala rotativa con cuatro rotores instalados de manera equidistante en forma de cruz en la plataforma de vuelo y utiliza energía eléctrica que es suministrada por una batería, además de poseer algoritmos de control y sensores para la estabilización de vuelo del UAV. Las principales ventajas y desventajas de este tipo de UAV se pueden ver en la Tabla 2.

Tabla 2.
Ventajas y Desventajas de cuadricópteros

Ventajas	Desventajas
• Rápida aceleración • Estructura Robusta • Gran variedad de modelos en el mercado	• Velocidad limitada • Capacidad de carga limitada

2.7.1 Metodología de vuelo

Para el desplazamiento y la maniobrabilidad el cuadricóptero se varía la velocidad angular de dos de sus rotores con la combinación de los tres grados de libertad que posee que son yaw, pitch y roll. Donde el movimiento en yaw está definido por el ángulo de giro Psi (ψ*)*, pitch por el ángulo de giro Zeta (θ*)* y roll por el ángulo de giro Fi (ϕ*)* (ver Figura 19).

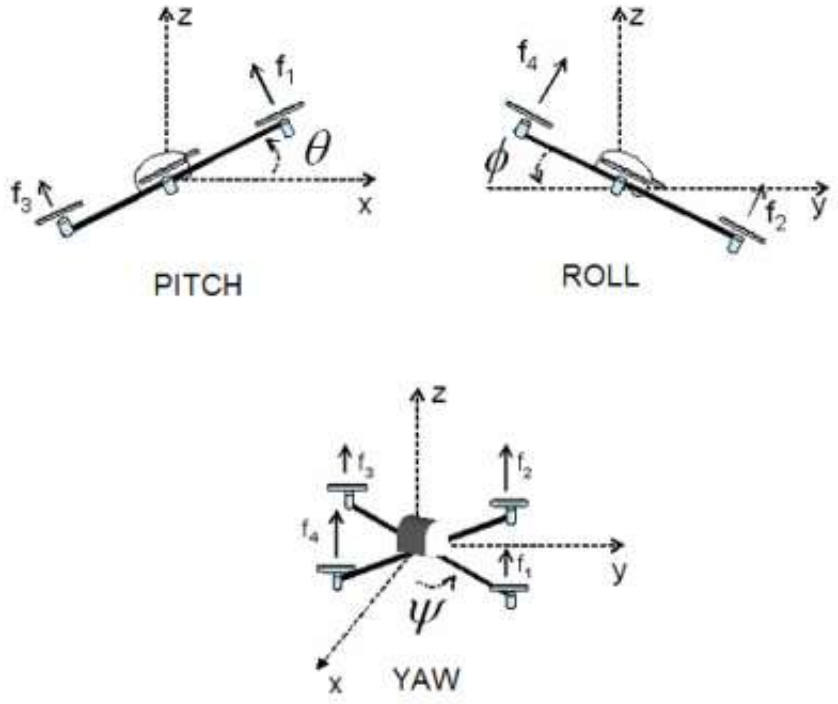


Figura 19. Grados de libertad de un cuadricóptero
Fuente: (Mayorga, 2009)

2.7.2 Parrot Bebop

El Parrot Bebop es un cuadricóptero originario de la empresa francesa Parrot, desarrollado para ser robusto y que pueda ser controlado de forma segura (ver Figura 20). Tiene integrado en su controlador algoritmos como el retorno a casa por medio del GPS y la estabilización de video. Posee una cámara "ojo de pez" de 14 megapíxeles, lo que le permite grabar videos y tomar imágenes en un campo de 180 grados con una alta calidad. Para su manejo se utiliza la aplicación "Freeflight 3" que se encuentra disponible para iOS y android.

Figura 20. Parrot Bebop drone
Fuente: (Parrot Bebop Drone y Skycontroller, 2017)

2.7.2.1 Especificaciones técnicas

Tabla 3.
Especificaciones Técnicas de UAV Parrot Bebop

Especificaciones Técnicas de Parrot Bebop	
Conectividad	Wi-Fi 802.11 a/b/g/n/ac Antenas Wi-Fi: MIMO de doble banda para 2.4 y 5 GHz Potencia de envio: hasta 21 dbm Rango de señal: hasta 250 metros
Estructura	Alta resistencia 4 Motores sin escobillas Hélices de bloqueo automático de tres palas Anti vibración
Velocidad	13 m/s
Cámara	CMOS 14 Megapíxeles Lente ojo de pez Estabilización de video: digital en 3 ejes Definición de video: 1920x1080p (30 fps) Definición de la foto: 4096x3072 px Codificación de video: H264 Formato de archivo de foto: JPEG, SIN PROCESAR, DNG Memoria interna: Flash 8 GB
Batería	Polímero de litio 1200 mAh Tiempo de vuelo: 22 minutos con 2 baterías incluidas

CONTINÚA

Especificaciones Técnicas de Parrot Bebop	
Procesador	CPU: Dual-Core CPU Parrot P7 de doble núcleo Cortex 9 GPU de cuatro núcleos Memoria flash de 8Gb Sistema operativo: Linux Desarrollo: SDK de código abierto
Sensores	Magnetómetro de 3 ejes Giroscopio de 3 ejes Acelerómetro de 3 ejes Sensor de flujo óptico Sensor de ultrasonido (analiza la altitud de vuelo hasta 8 metros) Sensor de presión
Geo-ubicación	GNSS (GPS + GLONASS)
Dimensiones	28x32x3.6cm sin el casco 33x38x3.6cm con el casco
Peso	400 gramos
Compatibilidad	iOS, Android, Windows Phone Smartphones

Fuente: (Parrot Bebop Drone, 2017)

CAPÍTULO III

3 EXTRACCIÓN DE CARACTERÍSTICAS DEL CUERPO HUMANO

3.1 Configuración de entorno de trabajo

Para el desarrollo del sistema, en la etapa de extracción de características se utiliza un sensor Kinect, el cual, es el encargado de proveer las coordenadas en el espacio de las articulaciones del cuerpo humano gracias a la combinación de la cámara RGB y el sensor de profundidad. El sistema está desarrollado sobre ROS y Ubuntu 14.04 por lo que en esta etapa de desarrollo del proyecto se realiza una configuración del entorno de trabajo, luego se realizan pruebas para comprobar la correcta instalación de las herramientas necesarias y se implementa un algoritmo para la obtención de puntos de interés del cuerpo humano los cuales se presentan en una interfaz.

3.1.1 Instalación de ROS

Primero es necesario realizar la instalación del sistema operativo de robótica ROS. La versión elegida es ROS Indigo y estará montada sobre Ubuntu Trusty 14.04. Para poder realizar la instalación de ROS en Ubuntu sin restricciones es necesario dar permisos para orígenes de software de tipo libre y abierto de Ubuntu, así como, software de controladores privativos (ver Figura 21).

Figura 21. Origines de software necesario para instalar ROS

Cuando ha sido verificado los orígenes del software en Ubuntu se puede continuar con la instalación de ROS. Para realizar la instalación, la actualización de los paquetes, la inicialización del sistema y finalmente la configuración necesaria para inicializar ROS siempre que se ejecute un nuevo terminal en Ubuntu, se utilizan los siguientes comandos:

1. “sudo apt-get install ros-indigo-desktop-full”
2. “sudo rosdep init”
3. “rosdep update”
4. “echo “source /opt/ros/indigo/setup.bash” " >> ~/.bashrc”
5. “source ~/.bashrc”

Luego de ejecutar los comandos antes mencionados, se debe ejecutar un comando “roscore” para verificar que ROS este correctamente instalado como se muestra en la Figura 22.

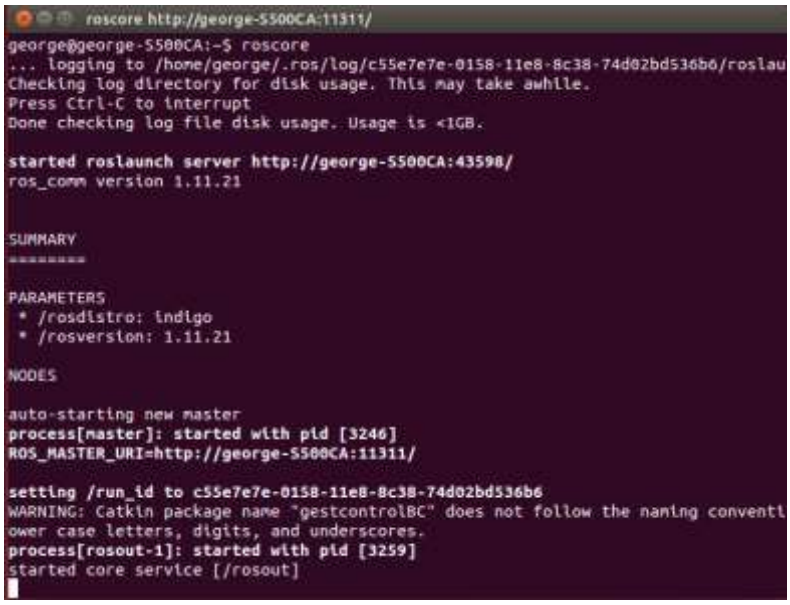


Figura 22. Comprobación de la instalación de ROS

Finalmente en esta sección, es necesario crear y configurar un espacio de trabajo o también conocido en ROS como “catkin_ws” como se muestra en la Figura 23 que servirá como base para instalación de librerías y creación de paquetes para llevar a cabo el sistema de reconocimiento de gestos corporales. Se utilizan los siguientes comandos para realizar su creación:

1. “mkdir -p ~ / catkin_ws / src”
2. “cd ~ / catkin_ws/”
3. “catkin_make”

Figura 23. Directorio de espacio de trabajo catkin_ws

3.2 Instalación del sensor Kinect

En esta etapa del proyecto de investigación es necesario instalar varias librerías y frameworks para que el sensor Kinect se pueda comunicar en ROS. Entre estos se tiene: OpenNI, Sensor Kinect, Openni_launch y Openni_tracker.

3.2.1 OpenNI

Es una framework de acceso libre que sirve para el desarrollo de aplicaciones de interacción natural. Por lo que facilita la comunicación con sensores de video y profundidad como los que posee el sensor Kinect. Los comandos necesarios para realizar la instalación de OpenNI son los siguientes:

1. “git clone https://github.com/OpenNI/OpenNI.gi”
2. “cd OpenNI”
3. “git checkout Unstable-1.5.4.0”
4. “cd Platform/Linux/CreateRedist”
5. “sudo chmod +x RedistMaker”
6. “./RedistMaker”
7. “cd ../Redist/OpenNI-Bin-Dev-Linux-[xxx]”
8. “sudo ./install.sh”

Finalizada la instalación se deberá tener un directorio como el que se muestra en la Figura 24.

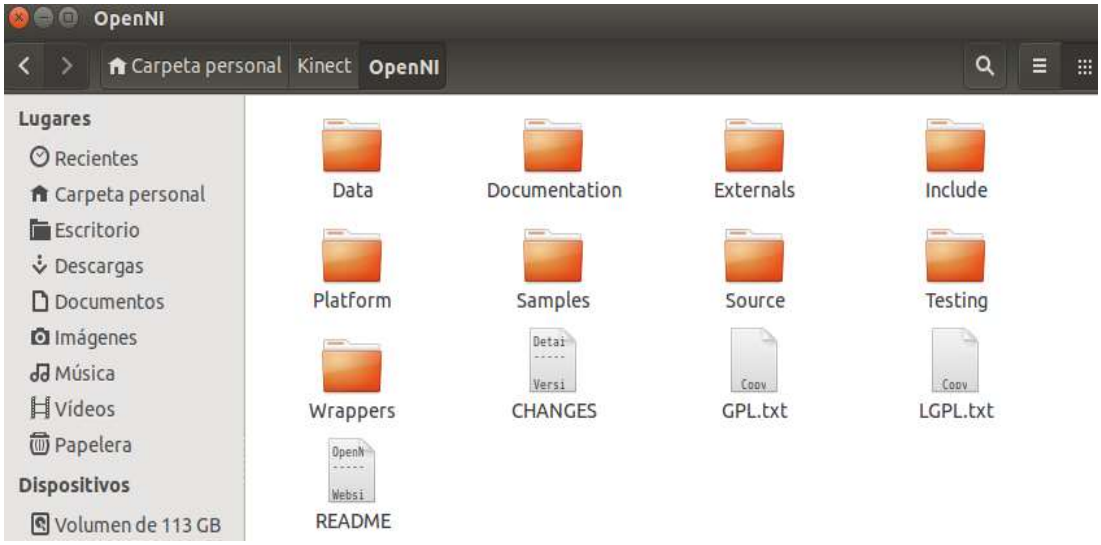


Figura 24. Directorio de OpenNI

3.2.2 Sensor Kinect

El módulo del sensor Kinect se procede a instalar luego de instalar el framework para interacción natural y cuando se termine su instalación se tendrá un directorio como el que se observa en la Figura 25. Este directorio contiene los controladores necesarios para poder acceder al sensor Kinect por medio de ROS. Los comandos para realizar la instalación son los siguientes:

1. "git clone git://github.com/ph4m/SensorKinect.git"
2. "cd SensorKinect/Platform/Linux/CreateRedist"
3. "sudo chmod +x RedistMaker"
4. "./RedistMaker"

Figura 25. Directorio de SensorKinect

Ahora es necesario comprobar que el sensor Kinect este correctamente instalado para lo cual es necesario la instalación de “Openni_launch”. El cual es ejecutable que permite abrir un dispositivo de interacción natural como el sensor Kinect y además carga los nodos de ROS necesarios para poder visualizar la cámara RGB y obtención de datos del sensor de profundidad.

El comando para su instalación es:

- “sudo apt-get install ros-indigo-openni-camera ros-indigo-openni-launch”

Si la ejecución se realiza de manera correcta debe mostrar un mensaje como el que se muestra en la figura 26. El comando para realizar la ejecución es el siguiente:

- “roslaunch openni_launch openni.launch

Figura 26. Ejecución de openni launch

Cuando se ejecuta un archivo launch en ROS se inician varios nodos correspondientes al proceso. En la Figura 27 se muestra los nodos que se inicial al ejecutar openni launch. El comando para ver los nodos que se inician es el siguiente:

- "rosrun rqt_graph rqt_grap"

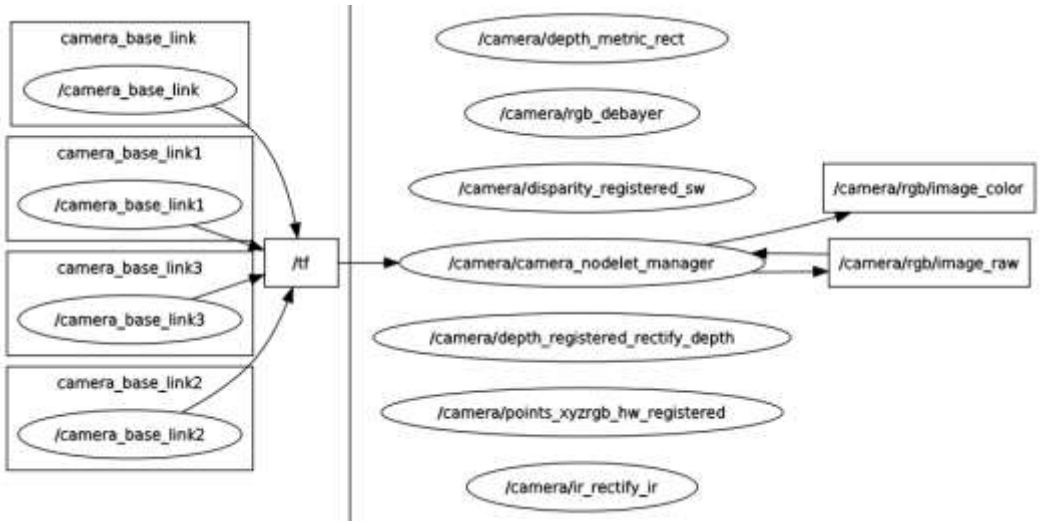


Figura 27. Nodos activos con openni launch

Finalmente se comprueba el sensor Kinect por medio de la cámara RGB como se ve en la Figura 28. El comando a ejecutar es:

- "rosrun image_view image_view image:=/camera/rgb/image_color"

Figura 28. Cámara RGB del sensor Kinect

3.3 Openni Tracker

La librería Openni_tracker desarrollada por Tim Field y con licencia BSD, permite detectar e identificar la pose de un usuario que frente al sensor Kinect adopte una posición PSI (ver Figura 29).

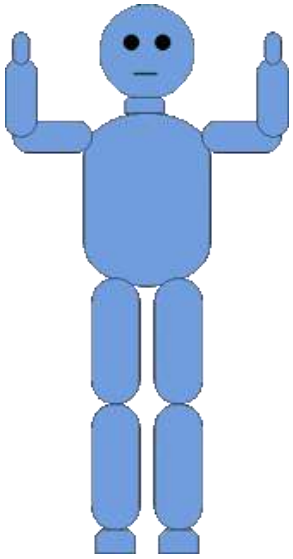


Figura 29. Pose PSI
Fuente: (Openni Tracker, 2017)

La pose se publica como un conjunto de transformaciones y a partir de estas se obtiene las coordenadas en el espacio de 15 partes del cuerpo humano. En la Tabla 4 se presentan las partes del cuerpo entregadas por Openni Tracker y sus transformaciones para ROS.

Tabla 4.
Partes reconocidas por Openni Tracker y transformaciones para ROS

Parte del cuerpo	**Transformación en ROS**
Cabeza	/head
Cuello	/neck
Torso	/torso
Hombro Izquierdo	/left_shoulder
Codo Izquierdo	/left_elbow
Mano Izquierda	/left_hand
Hombro Derecho	/right_shoulder

CONTINÚA

Parte del cuerpo	Transformación en ROS
Codo Derecho	/right_elbow
Mano Derecha	/right_hand
Lado izquierdo de la cadera	/left_hip
Rodilla izquierda	/left_knee
Pie izquierdo	/left_foot
Lado derecho de la cadera	/right_hip
Rodilla derecha	/right_knee
Pie derecho	/right_foot

Para realizar la instalación primero es necesario descargar e instalar NITE en su versión 1.5.2.23 para Linux ya que es un complemento para que Openni Tracker pueda realizar el reconocimiento de las partes del cuerpo humano mencionados en la Tabla 4. Luego se realiza la instalación de Openni Tracker en el espacio de trabajo creado "catkin_ws" con los siguientes comandos:

1. "cd ~catkin_ws/src"
2. "git clone https://github.com/ros-drivers/openni_tracker.git"
3. "cd .."
4. "catkin_make"

3.4 Desarrollo de algoritmo de extracción de características

El script de extracción de características está desarrollado en el lenguaje de programación Python y para su ejecución es necesario realizar una secuencia de inicio como la que se muestra en el diagrama de flujo de la Figura 30.

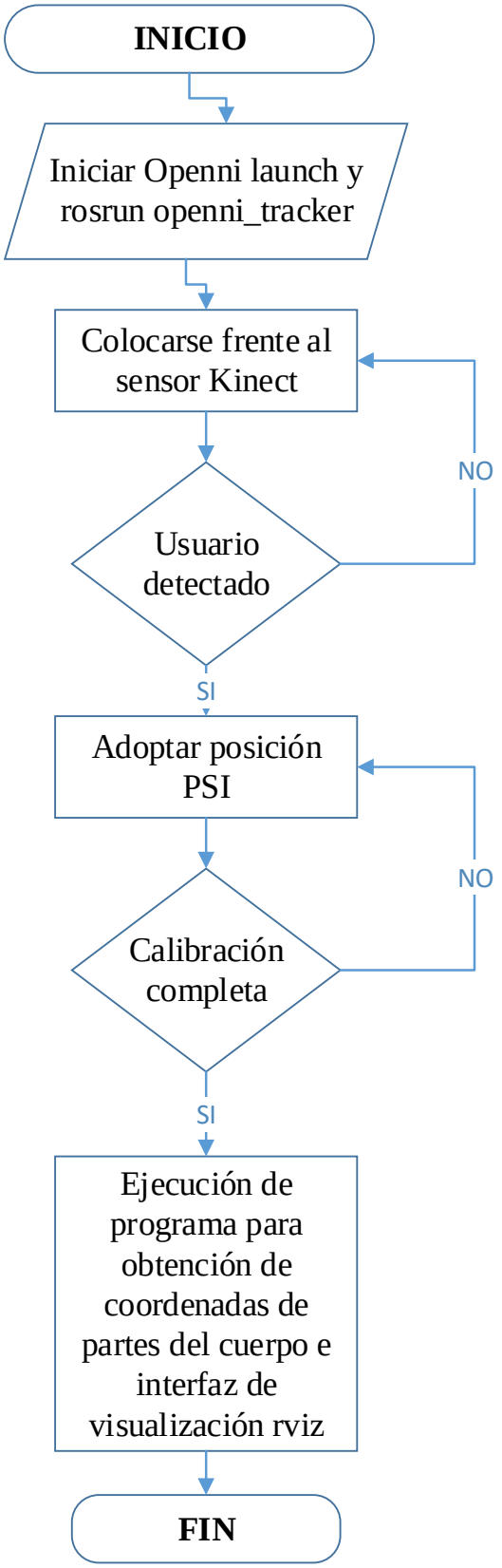


Figura 30. Diagrama de flujo de reconocimiento del usuario con Kinect

Al iniciar el nodo openni tracker es necesario adoptar la posición PSI para detectar al nuevo usuario como se muestra en la Figura 31.

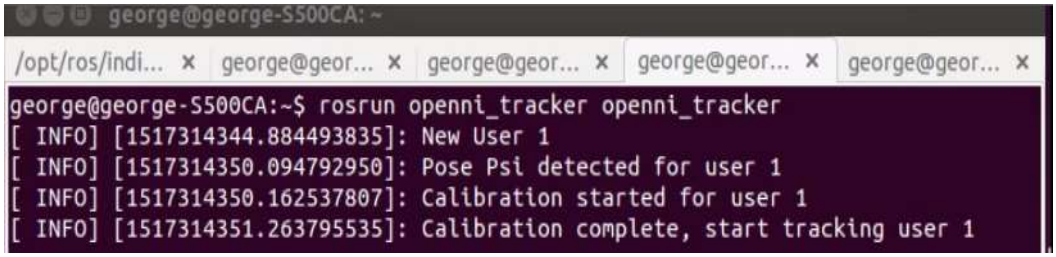


Figura 31. Detección de usuario con Openni Tracker

Cuando el usuario es detectado se puede iniciar el script desarrollado en Python. El programa se inicia como un nuevo nodo en ROS y lo que realiza es suscribirse al tópico de mensajes de transformaciones (/tf) de manera que recibe la pose entregada por Openni Tracker.

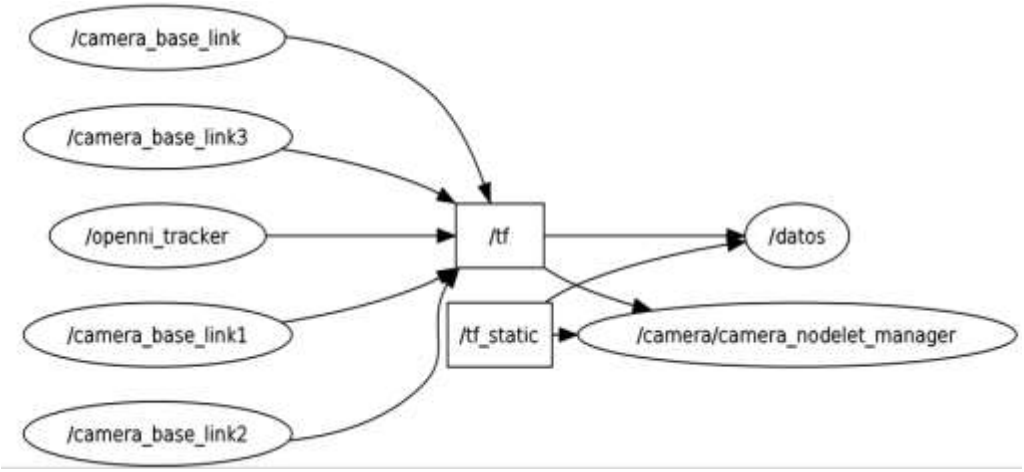


Figura 32. Nodo de script de extracción de características recibiendo datos /tf

Finalmente en el script al recibir los datos /tf que contienen la pose de las articulaciones mostradas en la Tabla 4. Se realiza una segmentación de manera que se obtenga únicamente las posiciones en el eje de coordenadas x, y, z como se muestra en la Figura 33.

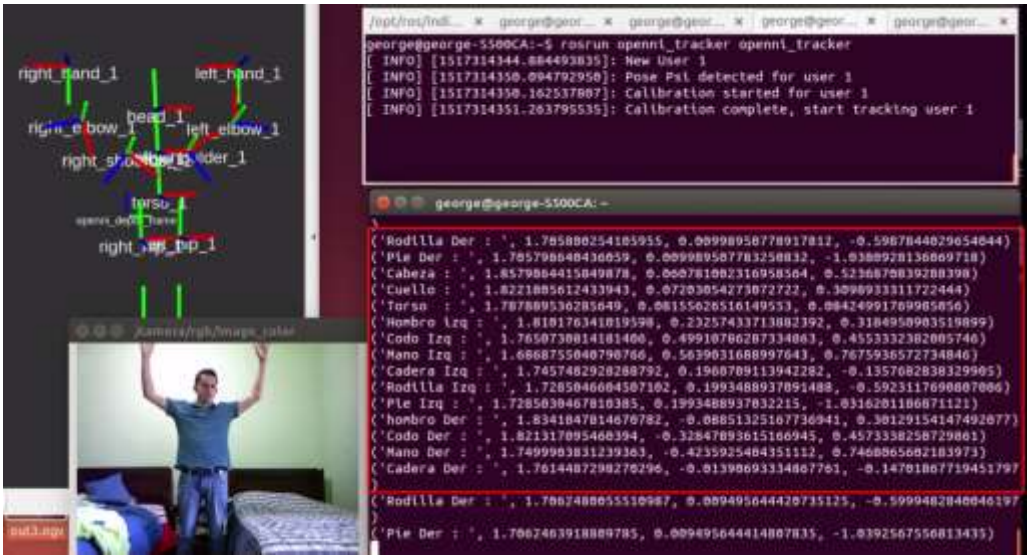


Figura 33. Ejecución de script de extracción de características.

CAPÍTULO IV

4 CLASIFICACIÓN DE GESTOS CORPORALES

4.1 Gestos corporales

Para el sistema de reconocimiento de gestos corporales es necesario definir los gestos a utilizar para el control del micro vehículo aéreo. Los gestos ideales son los gestos deícticos ya que permiten señalar o seleccionar objetos por lo que son gestos intuitivos y se pueden utilizar para representar acciones como moverse adelante, atrás, entre otros.

En el sistema de reconocimiento de gestos corporales se tiene cuatro grados de libertad por lo que es necesario definir los gestos para las acciones de moverse hacia, adelante, atrás, derecha, izquierda, arriba y abajo. Adicional a esto, es necesario definir gestos para instrucciones como despegar, aterrizar y emergencia.

4.1.1 Gestos con el brazo izquierdo

Con el brazo izquierdo se controla las acciones de mover hacia arriba y abajo el micro vehículo aéreo multirotor. Los gestos a utilizar se muestran en la Figura 34.

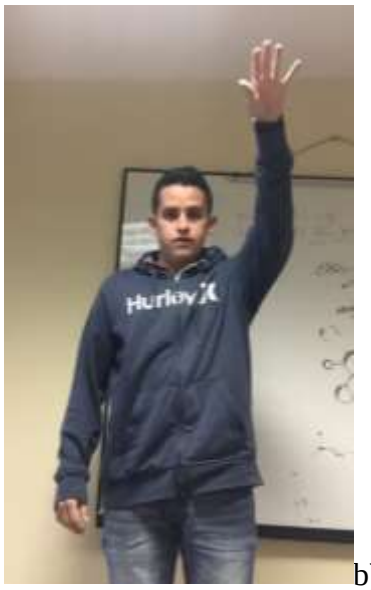
a)  b) 

Figura 34.Gestos con el brazo izquierdo a) Arriba b) Abajo

4.1.2 Gestos con el brazo derecho

Con el brazo derecho se controla las acciones de mover hacia adelante, atrás, derecha e izquierda el micro vehículo aéreo. Los gestos utilizados se muestran en la Figura 35.

a)  b) 

c)  d) 

Figura 35.Gestos con el brazo derecho a) Adelante b) Atrás c) Derecha d) Izquierda

4.1.3 Gestos con los dos brazos

Este tipo de gestos se utilizan en el sistema para acciones de despegar, aterrizar y emergencia. Los gestos se muestran en la Figura 36.

a)  b) 

c) 

Figura 36.Gestos con dos brazos a) Despegar b) Aterrizar c) Emergencia

4.2 Entrenamiento del sistema

Al tener identificado los gestos corporales para el control del micro vehículo aéreo multirotor, es necesario realizar el aprendizaje de maquina supervisado. Para lo cual se debe realizar una recolección y almacenamiento de datos a partir del sensor Kinect de puntos específicos del cuerpo humano, la clasificación mediante SVM para finalmente generar el modelo de predicción (ver Figura 37).

Figura 37. Modelo general de entrenamiento del sistema

Se deben generar tres modelos de predicción ya que los gestos corporales a utilizar en el sistema se encuentran divididos en gestos con el brazo izquierdo, derecho y los dos brazos (ver Figura 38). Este seccionamiento de datos evita el utilizar datos

innecesarios en gestos que se realizan únicamente con la mano izquierda o derecha, además, reduce el tiempo de entrenamiento y la complejidad del modelo de predicción.

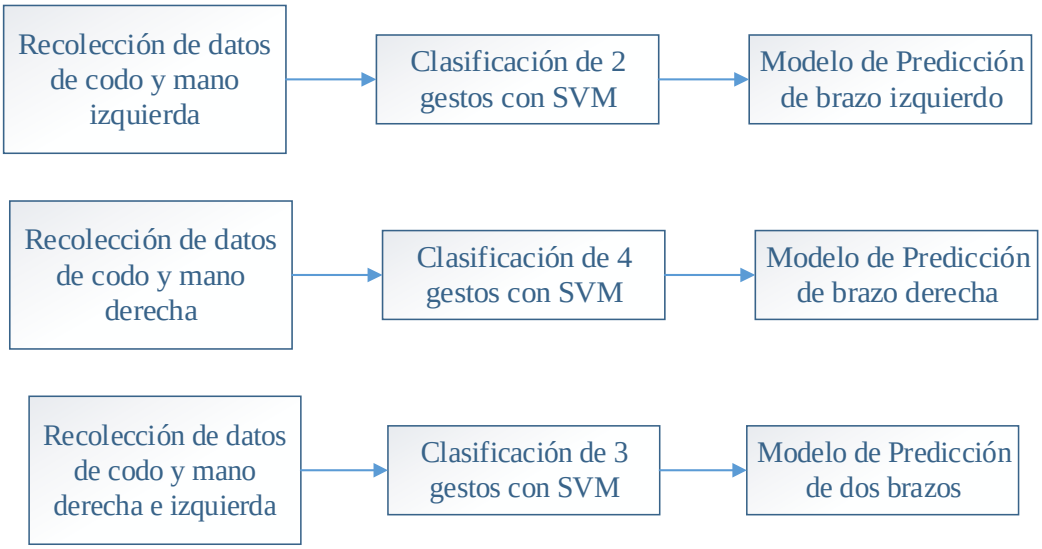


Figura 38. Entrenamiento de sistema

4.2.1 Recolección y almacenamiento de datos

Los gestos corporales del sistema utilizan el brazo izquierdo y derecho para cada una de las acciones, por lo que la recolección de datos se realiza únicamente para el codo y la mano de cada brazo debido a que son las partes del cuerpo que varían al realizar cada acción.

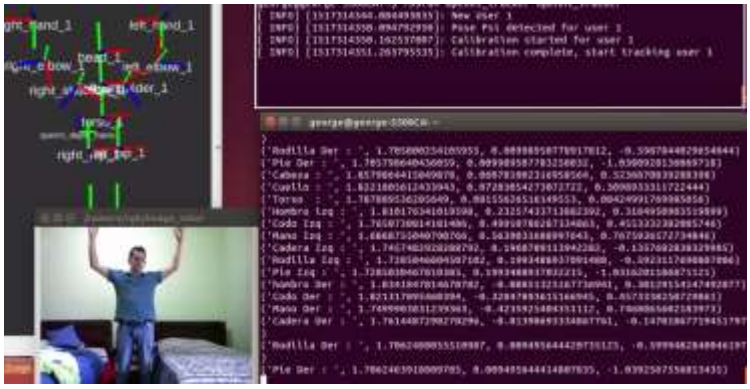
La recolección de los puntos de interés se la realiza mediante el sensor Kinect y son guardados en un archivo de texto con el formato como se muestra en la Figura 39. En el formato utilizado la primera columna representa un conjunto de clasificación, los datos siguientes del 1 al 6 representan las coordenadas en x, y, z del codo y la mano respectivamente. Este formato es el necesario para realizar el entrenamiento mediante SVM.

```
1 1:2.1562830838 2:-0.0776568439348 3:0.421570906438 4:1.91260758274 5:-0.0598617642733 6:0.5821122373
1 1:2.1567137217 2:-0.0798521472236 3:0.412254803782 4:1.91150073988 5:-0.0623904014567 6:0.571052976229
1 1:2.15859479912 2:-0.0871015299625 3:0.397901555906 4:1.90188149323 5:-0.0663171564557 6:0.538263155605
1 1:2.1486731081 2:-0.0851178246466 3:0.383506816959 4:1.88928409506 5:-0.0670511275943 6:0.50809105222
1 1:2.14381662808 2:-0.0872523029347 3:0.371922667318 4:1.87598783541 5:-0.0668509658628 6:0.476419988999
1 1:2.1423720469 2:-0.0905937541638 3:0.368282230729 4:1.86909547664 5:-0.076989045074 6:0.459534211093
1 1:2.15143799919 2:-0.0994472517466 3:0.368770783854 4:1.87846756655 5:-0.0821516006753 6:0.459705258211
```

Figura 39. Formato de almacenamiento de datos

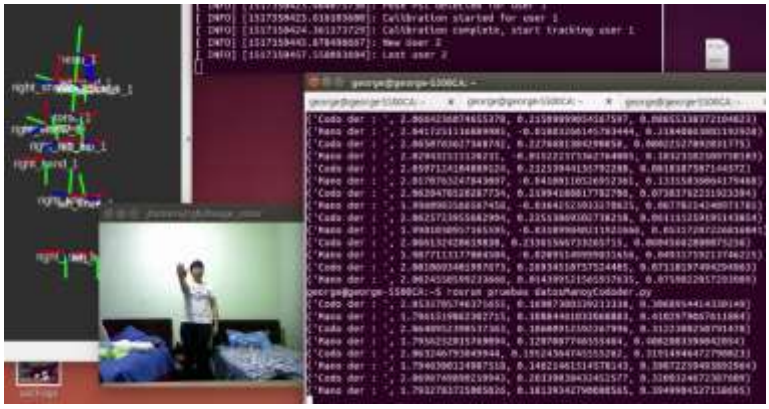
Los datos de cada uno de los gestos son recolectados con 5 usuarios como se muestra en la Figura 40. Para cada uno de los usuarios se recibe y almacena los datos de los gestos identificados para el sistema y se los clasifica en tres principales grupos de datos que son los realizados con el brazo derecho, izquierdo y los dos brazos.

a)



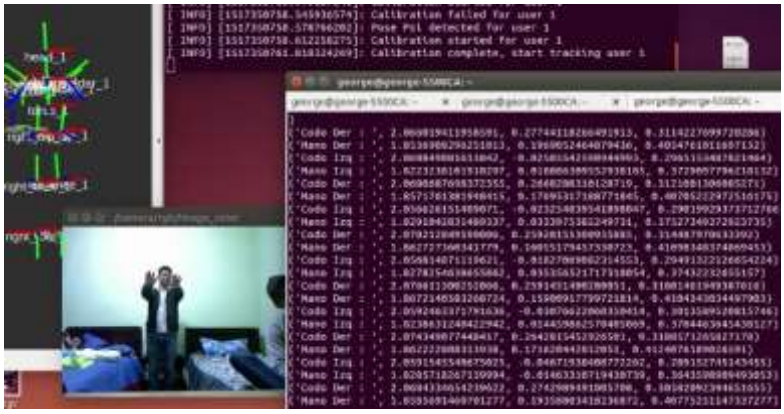
b)

c)



CONTINÚA

d)



e)

Figura 40. Usuarios para recolección de datos a) Usuario 1 b) Usuario 2 c) Usuario 3 d) Usuario 4 e) Usuario 5

Las características de los usuarios utilizados para el entrenamiento del sistema como el nombre, la edad y la estatura se muestran en la Tabla 5. La recolección de datos se lo realiza con cinco usuarios con el fin de tener mayor cantidad de información para realizar el entrenamiento con SVM y que los modelos generados para la detección de gestos sean más robustos y puedan realizar una mejor predicción; inclusive con usuarios que no participaron en la recolección de datos.

Tabla 5.
Usuarios para el entrenamiento del sistema de reconocimiento de gestos

Usuario	Nombre	Edad (años)	Estatura (m)
Usuario 1	George Bryan Cobeña Zambrano	23	1.72
Usuario 2	Jurgen Ronaldo Varela Bustamante	20	1.78
Usuario 3	José Luis Jiménez Álvarez	20	1.80
Usuario 4	Patricio Daniel Jiménez Álvarez	21	1.82
Usuario 5	Jefferson Guillermo Potosí Toledo	21	1.70

Finalmente al recolectar y almacenar los datos se tiene tres archivos correspondientes a gestos con el brazo izquierdo, derecho y dos brazos. Estos archivos utilizan identificadores de clase para diferenciar entre cada uno de los gestos y en la Tabla 6 se presenta un resumen de los archivos.

- Los gestos con el brazo izquierdo contiene identificadores de cada clase 1 y 2 para las acciones de moverse hacia arriba y abajo respectivamente. La recolección de datos se la realiza con 5 usuarios es por ello que total se tiene 1000 datos de gestos.
- Los gestos con el brazo derecho contiene identificadores de clase 1, 2, 3, 4 para las acciones de moverse hacia adelante, atrás, derecha e izquierda respectivamente. Por lo que se tiene un total de 2000
- Los gestos con los dos brazos contiene identificadores de clase 1, 2 y 3 para las acciones despegar, aterrizar y emergencia respectivamente. Por lo que se tiene un total de 1500 de datos de gestos.
- Adicionalmente en cada archivo se agrega una clase más de 300 datos de gestos no utilizados o que representan acciones nulas de manera que se pueda descartar como posibles gestos.

Tabla 6.
Resumen de los datos recolectados y almacenados

Tipo de gestos	Filas	Columnas	Identificador de Clase
Brazo izquierdo Gestos: • Arriba • Abajo	1300: 1–500 arriba 501–1000 abajo 1001-300 sin gestos	7 1: clase 2– 4: coordenadas x,y,z codo 5–7: coordenadas x,y,z mano	1: arriba 2: abajo 3: nada
Brazo derecho Gestos: • Adelante • Atrás • Derecha • Izquierda	2300: 1–500 adelante 501–1000 atrás 1001–1500 derecha 1501–2000 izquierda 2001–2300 sin gestos	7: 1: clase 2– 4: coordenadas x,y,z codo 5–7: coordenadas x,y,z mano	1: adelante 2: atrás 3: derecha 4: izquierda 5: nada

CONTINÚA

Tipo de gestos	Filas	Columnas	Identificador de Clase
Dos brazos Gestos: • Despegar • Aterrizar • Emergencia	1800: 1–500 despegar 501–1000 aterrizar 1001–1500 emergencia 1501–1800sin gestos	13: 1: clase 2– 4: coordenadas x,y,z codo derecho 5–7: coordenadas x,y,z mano derecha 8– 10: coordenadas x,y,z codo izquierdo 11–13: coordenadas x,y,z mano izquierda	1: despegar 2: aterrizar 3: emergencia 4: nada

4.2.2 Clasificación con SVM

En esta sección se realiza la clasificación mediante SVM para los gestos realizados con el brazo izquierdo, derecho y los dos brazos, para finalmente obtener sus respectivos modelos de predicción. Para realizar la clasificación de los gestos corporales se utiliza la librería LIBSVM desarrollada por Chang y Lin (Chang & Lin, 2017).

En el diagrama de flujo mostrado en la figura 41 se muestra la secuencia a seguir para realizar la clasificación y finalmente obtener el modelo de predicción. Cabe recalcar que esta secuencia se aplica para los tres modelos de gestos. Por lo que al finalizar esta etapa se cuenta con tres modelos de predicción.

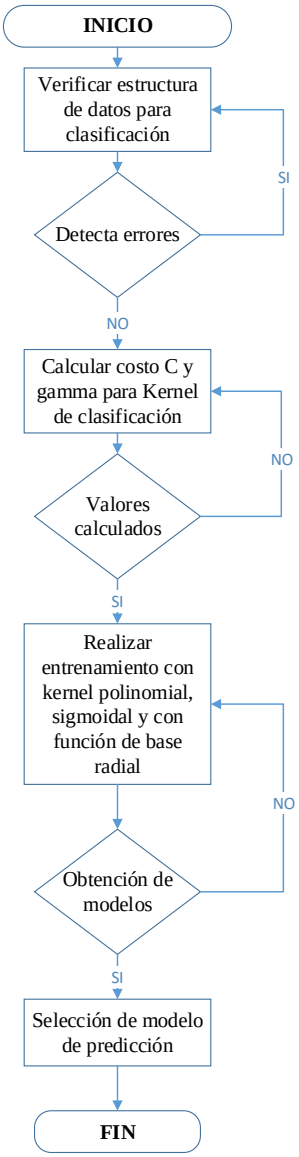


Figura 41. Diagrama de flujo para el Entrenamiento SVM

4.2.2.1 Verificación de datos de entrenamiento

Para verificar los datos de entrenamiento para la librería SVM se utiliza la herramienta llamada "checkdata.py" la cual se encarga de verificar que todas las líneas del archivo se encuentren con la estructura correcta. Para realizar esta verificación es necesario copiar el archivo en la carpeta "tolos" en el directorio de la librería SVM y luego se ejecuta el siguiente comando:

- "./checkdata.py nombre_del_archivo.txt"

Al realizar la ejecución se obtiene una respuesta sobre si el archivo está correctamente estructurado, como se muestra en la Figura 42.

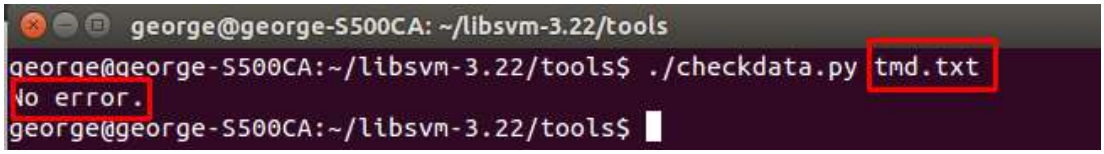


Figura 42. Verificación de estructura de datos para libSVM

4.2.2.2 Cálculo de parámetros para Kernel de clasificación

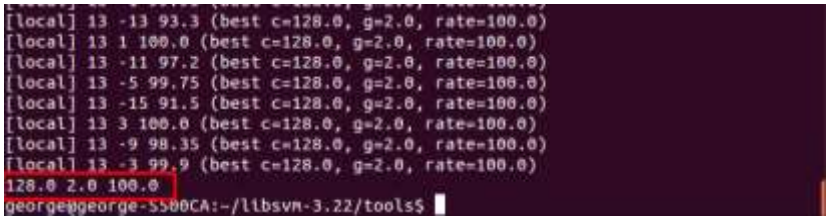
Son varios los gestos que se desean clasificar por lo que se el problema de reconocimiento de gestos se trata de una multiclasificación. En este tipo de problemas se pueden utilizar un kernel polinomial, kernel de base radial (RBF) o kernel sigmoidal.

Estos tipos de kernel necesitan parámetros como son el coste "C" y una constante gamma "g" que sirven para minimizar la norma de los vectores soporte. En la librería LIBSVM se cuenta con una herramienta que facilita el cálculo de estos valores llamada "grid.py". Para su ejecución se utiliza el siguiente comando:

- "./grid.py nombre_del_archivo.txt"

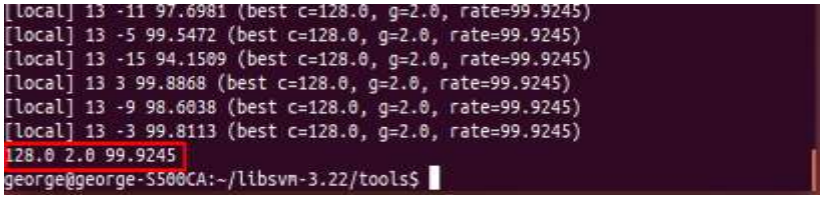
Al realizar la ejecución se obtiene los parámetros de kernel con la mayor tasa de clasificación como se muestra en la Figura 43. El coste "C" representa el margen blando que se debe añadir en el conjunto de entrenamiento por lo que un valor alto significa que los datos son casi perfectamente separables y un valor pequeño del coste representa un numero de muestras que presentan dificultad para su clasificación.

a)



CONTINÚA

b)



c)

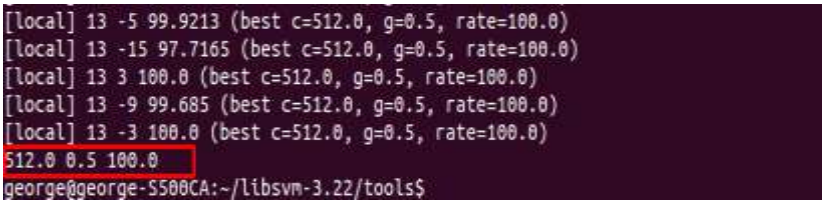


Figura 43. Parámetros para kernel a) brazo izquierdo b) brazo derecho c) dos brazos

En la Tabla 7 se presentan resúmenes se los valores calculados hasta alcanzar la combinación de parámetros en la que se alcance una mayor tasa de clasificación para los gestos.

Tabla 7.
Resumen de los parámetros calculados para entrenamiento

Gestos	Coste "C"	Gamma	Tasa (%)
Brazo izquierdo	32	0.0078125	91.6
	32	0.5	96.3
	2048	0.5	99.6
	2048	2	99.95
	512	2	100
	128	2	100
Brazo derecho	32	0.0078125	95.05
	32	0.5	99.54
	2048	0.5	99.88
	128	0.2	99.92
Dos brazos	32	0.0078125	97.87
	32	0.5	99.76
	2048	0.5	99.86
	512	0.5	100

4.2.2.3 Obtención de modelos de entrenamiento

Para realizar el entrenamiento se utiliza la herramienta “svm-train” de la librería LIBSVM. Para ejecutar esta instrucción es necesario seleccionar entre los parámetros que se muestran en la Figura 44.

```
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
```

Figura 44. Parámetros para herramienta svm-train
Fuente: (Chang & Lin, 2017)

Cuando se realiza el entrenamiento se obtiene un resultado como el que se muestra en la Figura 45. El comando para realizar el enteramiento es el siguiente:

- “./svm-train –c 128 –g 2 –t 2 nombre_del_archivo.txt nombre_archivo_almancenamiento.train”

```
optimization finished, #iter = 206
nu = 0.000860
obj = -46.769345, rho = 2.409783
nSV = 13, nBSV = 0
Total nSV = 67
```

Figura 45. Resultado de herramienta svm-train

Para realizar el entrenamiento se utilizan los parámetros de coste y gamma se utilizan los seleccionados en la Tabla 7 y se realiza el entrenamiento con el kernel polinomial, kernel en función de base radial (RBF) y kernel sigmoidal para obtener el número de iteraciones necesarias para realizar la clasificación, así como, el número de vectores soporte, los cuales sirven como indicadores para la selección del Kernel. La

Tabla 8 muestra un resumen de los entrenamientos realizados y los resultados obtenidos para los modelos del brazo izquierdo, derecho y dos brazos.

Tabla 8.
Resumen de entrenamiento con diferentes Kernel

Gestos	Kernel	Numero de Iteraciones	Numero de vectores soporte
Brazo izquierdo	Polinomial	211723	23
	RBF	261	29
	Sigmoidal	300	2000
Brazo derecho	Polinomial	5940	44
	RBF	206	67
	Sigmoidal	250	2650
Dos brazos	Polinomial	31	25
	RBF	48	31
	Sigmoidal	96	1270

Al finalizar el entrenamiento y realizando la comparativa se observa en la Tabla 8 que el kernel con un mejor funcionamiento para el sistema de reconocimiento de gestos corporales es el kernel en función de base radial ya que con menor número de interacciones logra realizar la multiclasificación en un menor número de iteraciones.

4.3 Implementación de algoritmo de detección de gestos

El script para la detección de gestos corporales está desarrollado en el lenguaje de programación Python y se estructura de tal forma que cargue los modelos de predicción generados en la sección anterior, lea los datos de la posición del codo y mano derecha e izquierda desde el sensor Kinect y realizar una comparación finalmente predecir el tipo de gesto. Para el desarrollo de este script y poder realizar la comparación con el modelo de predicción se sigue la estructura que se muestra en el diagrama de la Figura 46.

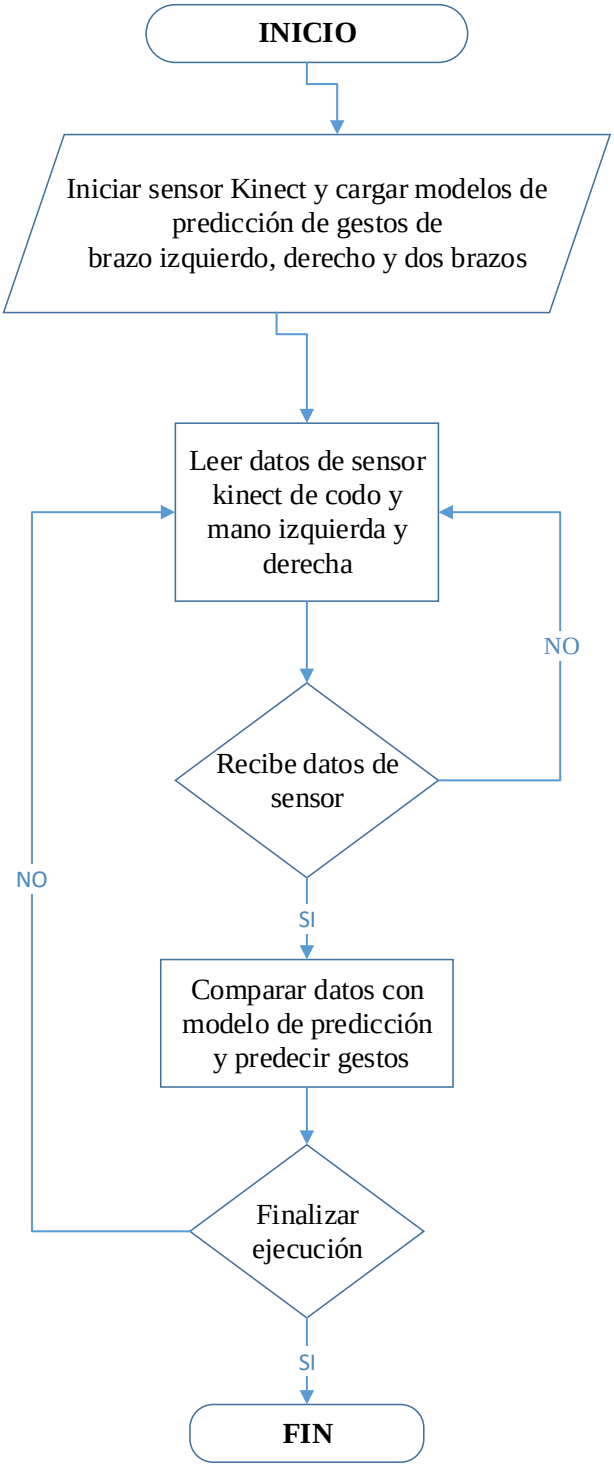


Figura 46. Diagrama de flujo de script de detección de gestos

El script para la detección de gestos corporales está desarrollado en el lenguaje de programación Python y para su funcionamiento primero se realiza la importación de

librerías “svm” para poder realizar las predicciones, “tf” para poder leer los datos del sensor Kinect y “rospy” necesaria para iniciar el nodo python en ROS (ver Figura 47).

```
#!/usr/bin/env python

import sys
sys.path.append('/home/george/libsvm-3.22/python')
import rospy
from svm import *
from svmutil import *
import tf
import numpy as np
import threading
```

Figura 47. Importación de librerías para script de reconocimiento de gestos

Ahora es necesario iniciar el nodo correspondiente al reconocimiento de gestos y cargar los modelos de clasificación obtenidos con el entrenamiento SVM. En esta etapa se cargan tres modelos correspondientes al brazo izquierdo, derecho y los dos brazos como se muestra en la Figura 48.

```
rospy.init_node('reconocimiento_de_gestos')
m = svm_load_model('t2m.train')
m1 = svm_load_model('tmd.train')
m2 = svm_load_model('tmi.train')
```

Figura 48. Importación de modelos

Lo siguiente es realizar la lectura de la posición de codo y mano derecha e izquierda con el sensor Kinect para que sean comparados mediante la función “svm_predict” de la librería importada svm con el modelo de clasificación. Finalmente el resultado de la función “svm-predict” servirá para identificar el tipo de gesto que se está realizando.

```
if __name__ == '__main__':
    listener = tf.TransformListener()
    rate = rospy.Rate(10.0)
    while not rospy.is_shutdown():
    try:
        (trans,rot) = listener.lookupTransform('/openni_depth_frame', '/right_elbow_1', rospy.Time(0))
        (trans1,rot1) = listener.lookupTransform('/openni_depth_frame', '/right_hand_1', rospy.Time(0))
        (trans2,rot2) = listener.lookupTransform('/openni_depth_frame', '/left_elbow_1', rospy.Time(0))
        (trans3,rot3) = listener.lookupTransform('/openni_depth_frame', '/left_hand_1', rospy.Time(0))
        except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
            continue
    y, x = [1], [[trans[0],trans[1],trans[2],trans1[0],trans1[1],trans1[2],trans2[0],trans2[1],trans2[2],
    y1, x1 = [1], [[trans2[0],trans2[1],trans2[2],trans3[0],trans3[1],trans3[2]]]
    y2, x2 = [1], [[trans[0],trans[1],trans[2],trans1[0],trans1[1],trans1[2]]]
    p_label, p_acc, p_val = svm_predict(y, x, m)
    p_label1, p_acc1, p_val1 = svm_predict(y1, x1, m1)
    p_label2, p_acc2, p_val2 = svm_predict(y2, x2, m2)
```

Figura 49. Lectura del sensor Kinect y comparación con modelo

4.3.1 Prueba del algoritmo de detección de gestos

Para probar el algoritmo de detección de gestos corporales es necesario ejecutar el sensor Kinect, el nodo “openni_tracker” y el nodo de reconocimiento de gestos. Los nodos en ejecución para esta parte del proyecto se observan en la Figura 50.

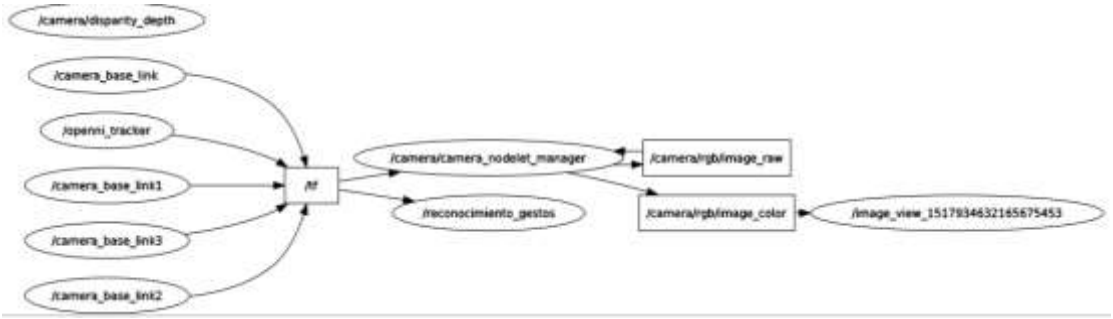


Figura 50. Nodos activos para el sistema de reconocimiento

Cuando los nodos están en ejecución se realiza todos los gestos corporales necesarios para observar el funcionamiento del sistema. En la Figura 51 se muestran las pruebas realizadas con gestos del brazo izquierdo, derecho y los dos brazos.

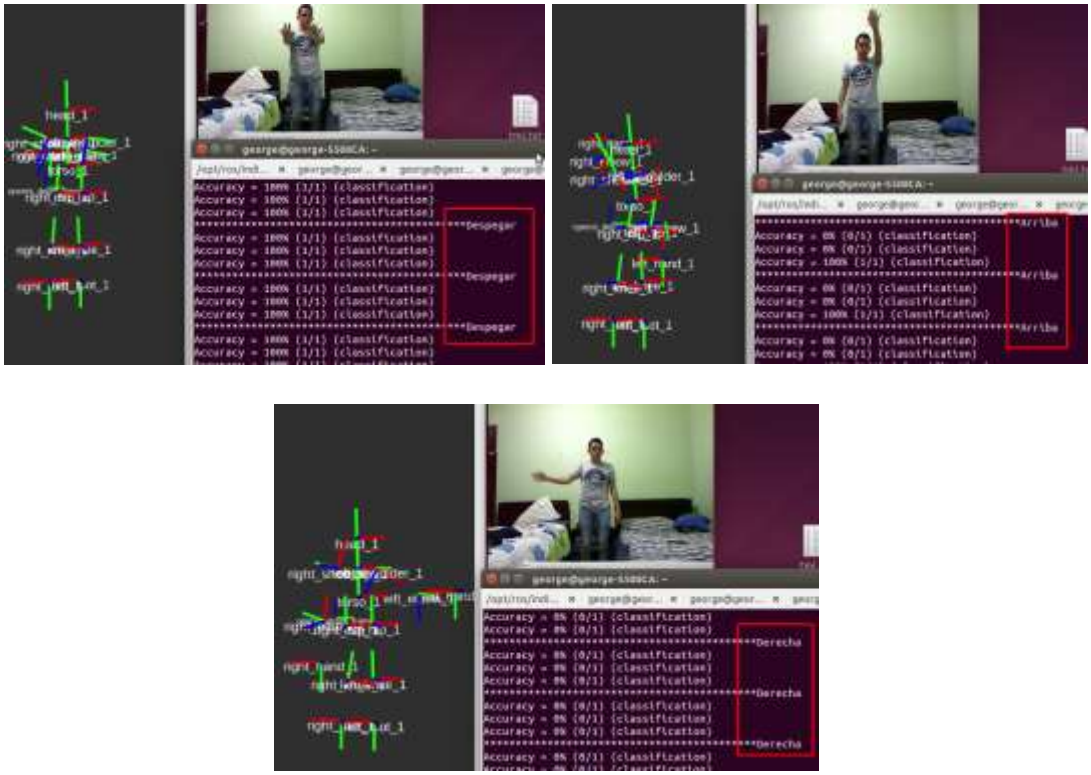


Figura 51. Prueba del nodo de reconocimiento de gestos corporales

CAPÍTULO V

5 COMUNICACIÓN CON MICRO VEHÍCULO AEREO Y DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO

5.1 Instalación de librerías de micro vehículo aéreo multirotor

Con el fin de poder iniciar el micro vehículo aéreo multirotor Parrot Bebop como un nodo ROS y que tenga la capacidad de recibir mensajes publicados en un tópico por el script de reconocimiento de gestos caporales, es necesario realizar la instalación de la librería "bebop_autonomy". Su instalación se la realiza en el espacio de trabajo creado para el proyecto llamada "catkin_ws" y los comandos utilizados para su instalación son los siguientes:

1. "cd catkin_ws/src"
2. "git clone https://github.com/AutonomyLab/bebop_autonomy.git src/bebop_autonomy"
3. "catkin make"
4. "rosdep update"

Finalizada este proceso se debe verificar la instalación, por lo que se debe realizar la conexión del computador con el micro vehículo aéreo. El drone funciona como router por lo que crea una red Wi-Fi es decir una red con estándar 802.11ac que opera en frecuencias de 2.5 y 5 GHz. La red que crea es una red tipo C y en la Tabla 9 se muestran las características de la conexión.

Tabla 9.
Direccionamiento IP del router del bebop drone

Dirección IP asignada a usuario	192.168.42.11
Puerta de enlace predeterminada	192.168.42.1
Mascara de subred	255.255.255.0

Una vez que se realiza la conexión se debe ejecutar el nodo del bebop drone para verificar su correcta instalación. Al realizar la ejecución deberá mostrar un resultado como el que se muestra en la Figura 52. El comando para realizar la ejecución es:

- “roslaunch bebop_driver bebop_node.launch”

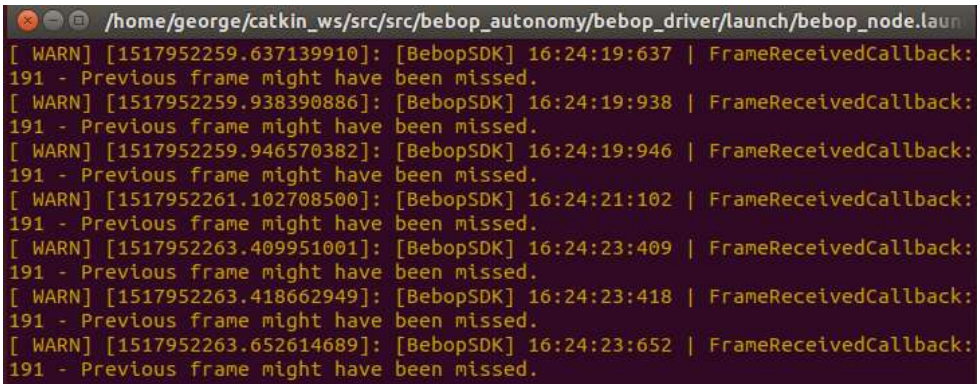


Figura 52. Funcionamiento de nodo de bebop drone

Se verifica que los nodos el nodo del bebop drone se inicie en ROS y debe presentar una estructura como la que se muestra en la Figura 53.

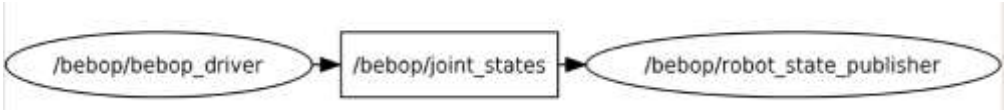


Figura 53. Nodo de bebop drone

5.2 Desarrollo de algoritmo para comunicación de gestos con bebop drone

Es necesario desarrollar un script el cual dependiendo el gesto detectado por el modelo de predicción enviar una serie de comandos que permitan pilotar el bebop drone. Para que el drone realice movimientos es necesario publicar en el nodo del bebop drone diferentes tipos de mensajes como se muestran en la Tabla 10.

Tabla 10.
Tipo de mensajes a publicar para generar movimientos en bebop drone

Movimiento	**Tipo de mensaje**	**Tópico**
Despegar	std_msgs/Empty	Takeoff
Aterrizar	std_msgs/Empty	Land
Emergencia	std_msgs/Empty	reset
Adelante	geometry_msgs/Twist	linear.x (+)
Atrás	geometry_msgs/Twist	linear.x (-)
Izquierda	geometry_msgs/Twist	linear.y (+)
Derecha	geometry_msgs/Twist	linear.y (-)
Arriba	geometry_msgs/Twist	linear.z (+)
Abajo	geometry_msgs/Twist	linear.z (-)

Los mensajes linear.x,y,z deben ser valores comprendidos entre +1 y -1 donde los valores más cercanos al límite representan una mayor velocidad en el drone y para los mensajes “Empty” se envía únicamente una orden de activación para realizar tareas específicas como despegar o aterrizar.

La estructura del script desarrollado para realizar el pilotaje del bebop drone con los gestos corporales sigue una secuencia como se muestra en el diagrama de flujo de la Figura 54.

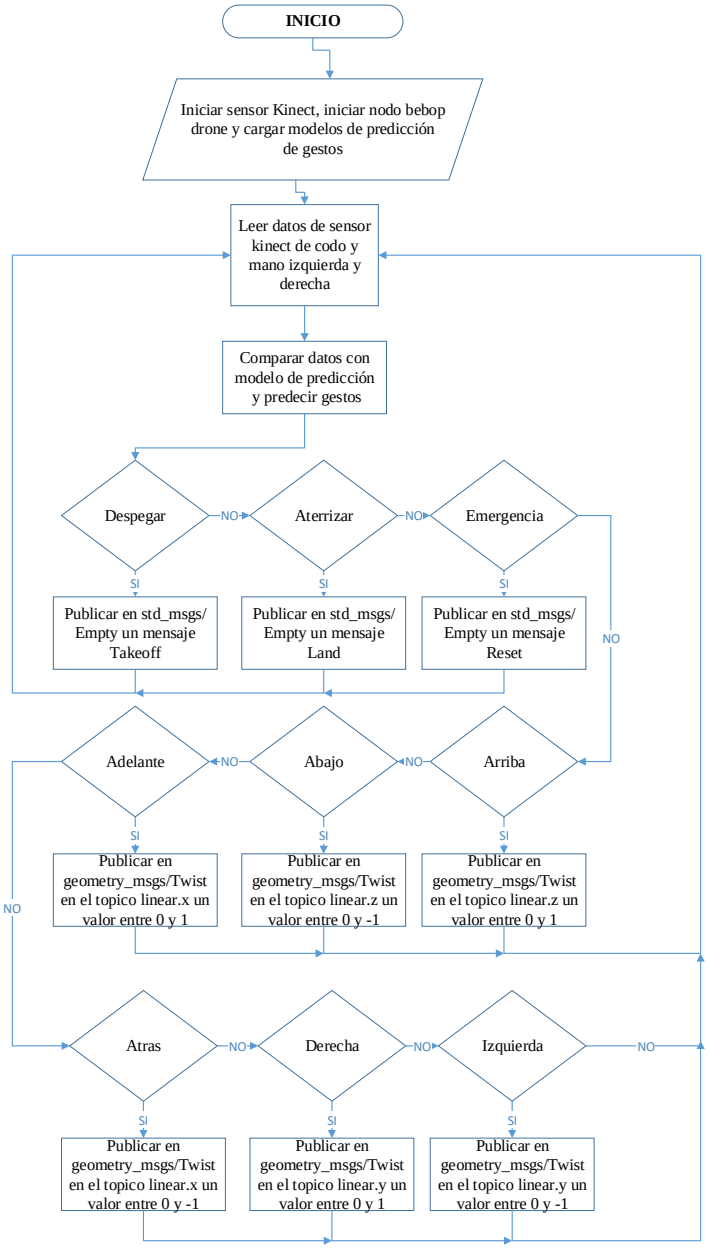


Figura 54. Diagrama de flujo de control de bebop drone con gestos

Para el script es necesario realizar la importación de las librerías “std_msgs.msg” y “geometry_msgs.msg”, específicamente “Empty” y “Twist” como se muestra en la Figura 55.

```
import std_msgs.msg
import geometry_msgs.msg
from std_msgs.msg import Empty
from std_msgs.msg import String
from geometry_msgs.msg import Twist
```

Figura 55. Importación de librerías

Luego se realiza el reconocimiento de gestos como se mostró en la sección 4.3 y cuando se reconozca un gesto corporal será necesario invocar a una función la cual realice la publicación del mensaje para que realice el movimiento. En las Figura 56 y 57 se muestra como realizar la publicación de un mensaje tipo Empty y un mensaje tipo Twits respectivamente.

```
if p_label[0] == 1:
    print('****************************************Despegar')
    rate.sleep()
    elevar()
```

```
def elevar():
    i = 0
    pub = rospy.Publisher('/bebop/takeoff', Empty,queue_size=10)
    rate = rospy.Rate(10)
    while i<=1:
    pub.publish()
    i=i+1
    rate.sleep()
```

Figura 56. Publicación de mensajes tipo Empty en bebop drone

```
if p_label1[0] == 1:
print('****************************************Adelante')
rate.sleep()
movimientoadelante()
```

```
def movimientoadelante():
    i=0
    pub = rospy.Publisher('/bebop/cmd_vel', Twist,queue_size=10)
    rate = rospy.Rate(10)
    while i<=2:
    i=i+1
    msg.linear.x = 0.5
    msg.linear.y = 0
    msg.linear.z = 0
    pub.publish(msg)
    rate.sleep()
```

Figura 57. Publicación de mensajes tipo Twist en bebop drone

5.3 Diseño de la interfaz gráfica de usuario

La interfaz gráfica para el sistema de reconocimiento de gestos corporales para controlar un micro vehículo aéreo multirotor, funciona como como un nodo independiente, el cual se suscribe al tópico de los mensajes publicados por el script reconocimiento de gestos y dependiendo la información recibida desplegara información de las instrucciones predichas por el modelo de clasificación.

La interfaz de usuario por lo tanto debe ser intuitiva, de fácil manejo y comprensión, además de, mostrar únicamente información relevante para poder realizar el control de micro vehículo aéreo de una manera óptima y sin distracciones. Por ello se realiza una interfaz en la cual se cuente con una ventana de información y una ventana de control de vuelo.

5.3.1 Ventana de información

Esta ventana cuenta con instrucciones claras y concisas de los gestos que se deben realizar para controlar el micro vehículo aéreo multirotor. Sirve como ventana de entrenamiento y aprendizaje de gestos de control del sistema para los usuarios. En la Figura 58 se muestra el diseño de la ventana.

Figura 58. Diseño de ventana de información

5.3.2 Ventana control de vuelo

Esta ventana mostrara la información de vuelo, por lo que debe dar información relevante con la que se pueda manipular el drone, con el fin de cumplir con estas especificaciones se toma en cuenta las siguientes consideraciones para el diseño de esta ventana:

- Transmisión en tiempo real de video desde la cámara integrada en el drone hacia la interfaz para permitir el control de vuelo estando en la misma o en diferentes habitaciones.
- Presentación clara de la instrucción de movimiento que se está enviando al drone. Aquí se debe tomar en cuenta que la distancia de funcionamiento ideal para el reconocimiento de gestos por el sensor Kinect es en un rango de 2 a 2.5 metros, por lo que el mensaje mostrado tiene que ser de un tamaño considerable.
- Presentación en tiempo real de la extracción de características del cuerpo humano, así como, presentación de posición del usuario para que se facilite el posicionamiento del usuario frente al sensor Kinect.

Tomando en cuenta las consideraciones mencionas en la Figura 59 se muestra el diseño para la ventana de control de vuelo.

Figura 59. Diseño de ventana de control de vuelo

5.3.3 Implementación de interfaz gráfica de usuario

Para la implementación de la interfaz gráfica se utiliza la librería “Tkinter “de Python y el paquete “rqt” de ROS. La herramienta TKinter sirve para desplegar la información de las instrucciones de vuelo y con la herramienta rqt se realiza la obtención en tiempo real de video de la cámara del drone, la presentación de la cámara RGB del sensor Kinect y la presentación de la imagen caracterizada del cuerpo humano.

La interfaz se crea como un nodo independiente de ROS que se suscribe al tópico de reconocimiento de gestos para desplegar las instrucciones, además se suscribe a los nodos Openni Tracker, sensor Kinect y bebop drone para mostrar toda la información necesaria. En las Figura 60 y 61 se muestra las ventanas de información y de la interfaz de vuelo en funcionamiento respectivamente.

Figura 60. Ventana de información

Figura 61. Ventana de interfaz de vuelo

5.4 Creación de paquete gestControlBC

El paquete gestcontrolBC llamado así para hacer referencia a “Gesture Control by Bryan Cobeña” es un paquete de ROS el cual tiene los scripts necesarios para que funcione el sistema, además de, un ejecutable “init.launch” donde puede iniciar cada una de las librerías necesarias para el funcionamiento del sistema de reconocimiento de gestos basado en SVM y sensor RGB-D y así poder controlar el micro vehículo aéreo multirotor. La creación de este paquete se realiza en el espacio de trabajo “catkin_ws” y para crearlo se utilizan los siguientes comandos:

1. “cd ~/catkin_ws/src”
2. “catkin_create_pkg gestcontrolBC std_msgs rospy roscpp”
3. “cd ~/catkin_ws”
4. “catkin_make”

Al finalizar la instalación se tiene un directorio como el que se muestra en la Figura 62.

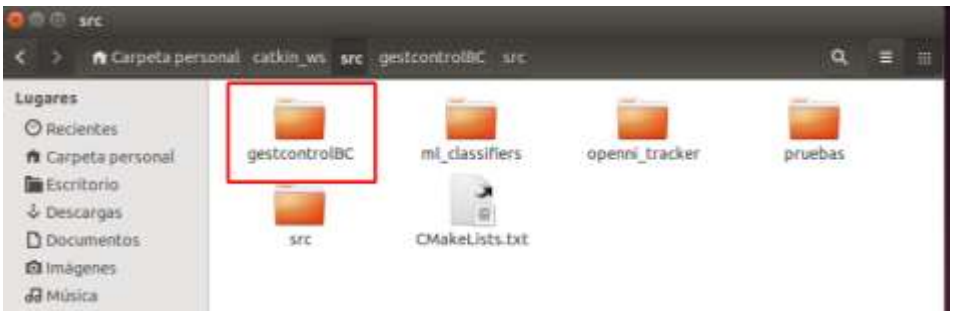


Figura 62. Creación de Paquete gestcontrolBC

Dentro de este paquete se encuentra el archivo “init.launch” el cual tiene una estructura como la mostrada en la Figura 63 y sirve para iniciar los módulos del sensor Kinect, el bebop drone y la interfaz de usuario para realizar el control de vuelo.

```
1  <launch>
2      <group name="openni">
3          <include file="$(find openni_launch)/launch/openni.launch" />
4      </group>
5      <group name="bebop">
6          <include file="$(find bebop_driver)/launch/bebop_node.launch" />
7      </group>
8      <group ns="interfaz">
9          <node pkg="rqt_gui" name="rqt_gui" type="rqt_gui"/>
10     </group>
11 </launch>
```

Figura 63. Archivo init.launch

Dentro del mismo paquete como se muestra en la Figura 64 se encuentran como ejecutables de python los archivos correspondientes a la interfaz de vuelo, al paquete de reconocimiento de gestos y envió de comandos al bebop llamado “gestocontrolBC.py” y el programa utilizado para la extracción de características del esqueleto humano.

Figura 64. Ejecutables del paquete gestcontrolBC

Finalmente se realiza la ejecución de todo el sistema y en la Figura 65 se muestran los paquetes, nodos y tópicos activos para el funcionamiento del sistema de reconocimiento de gestos corporales basado en SVM y sensor RGB-D para el control de micro vehículos aéreos multirotor.

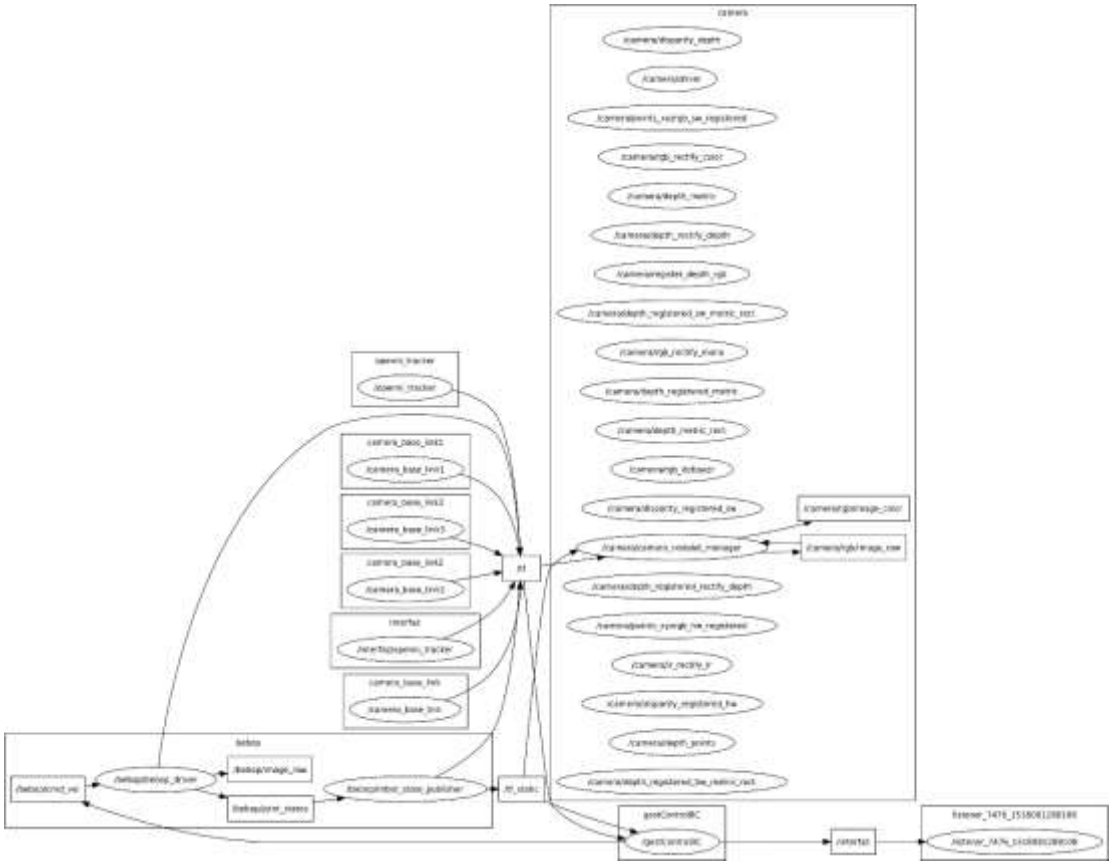


Figura 65. Nodos activos en ROS en la ejecución de gestcontrolBC

CAPÍTULO VI

6 PRUEBAS DE FUNCIONAMIENTO

Las pruebas realizadas se describen para conocer el funcionamiento del sistema de reconocimiento de gestos corporales para controlar un micro vehiculó aéreo multirotor y de esa manera determinar la capacidad de funcionamiento ante diferentes usuarios y en diferentes condiciones.

Las pruebas han sido realizadas con una metodología dinámica y consta de las siguientes fases:

- Primera fase: Pruebas de funcionamiento del sensor Kinect
- Segunda fase: Pruebas de funcionamiento del sistema de reconocimiento de gestos corporales con el bebop drone
- Tercera fase: Realizar una comparación con el sistema de reconocimiento de gestos y el sistema tradicional para controlar el drone.
- Curta fase: Trabajos futuros

6.1 Pruebas de funcionamiento del sensor Kinect

En estas pruebas se valida la distancia a la que debe estar el usuario para el reconocimiento correcto de los gestos corporales. De manera que cada uno de los gestos pueda ser identificado sin ningún problema y se utiliza una altura constante del sensor Kinect de 0.8 metros, debido a que es la distancia en la cual se realizó la recolección de datos para su posterior entrenamiento con SVM.

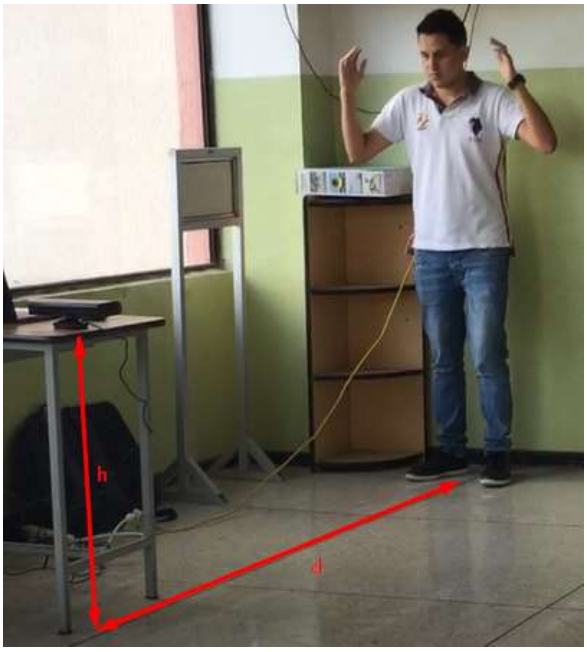


Figura 66. Pruebas de distancia

Tabla 11.
Pruebas de distancia

Prueba	Distancia (m)	Altura (m)	Observación
1	1.4 – 1.7	0.8	El sistema no detecta los gestos
2	1.7 – 2.0	0.8	El sistema no detecta gestos de: Adelante, atrás
3	2.0 – 2.3	0.8	El sistema detecta todos los gestos
4	2.3 – 2.6	0.8	El sistema detecta todos los gestos
5	2.6 – 2.9	0.8	El sistema no detecta gestos
6	2.9 - 3.1	0.8	El sistema no detecta los gestos

Con las pruebas realizadas para el sensor Kinect con una altura de 0.8m y a diferentes distancias se comprueba que la distancia ideal para utilizar el sistema de reconocimiento de gestos corporales, está en el rango de los 2.0 m hasta los 2.6 m.

Otra prueba que se realiza con el sensor Kinect tiene que ver con los grados de luxes para determinar las condiciones de iluminación en la que puede trabajar el sistema de reconocimiento de gestos. Con esta prueba se comprueba que el sistema funciona correctamente hasta un nivel de iluminación de 1000 luxes, pasado este nivel el sistema empieza a detectar con error los gestos o simplemente no los detecta. En la tabla 12 se muestra un resumen.

Tabla 12.
Pruebas de distancia

Prueba	Nivel de iluminación (lux)	Observaciones
1	60 - 600	Sistema funciona correctamente
2	601 – 1000	Sistema funciona correctamente
3	1000 – 1300	Sistema funciona con fallas
4	1300 en adelante	Sistema no funciona

6.2 Prueba del funcionamiento sistema

En esta fase pruebas se valida que las ventana de control de vuelo cumpla con el diseño del capítulo anterior y que funcione de manera correcta permitiendo detectar cada uno de los gestos correctamente y poder pilotear el drone.

En la figura 67 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para despegar.

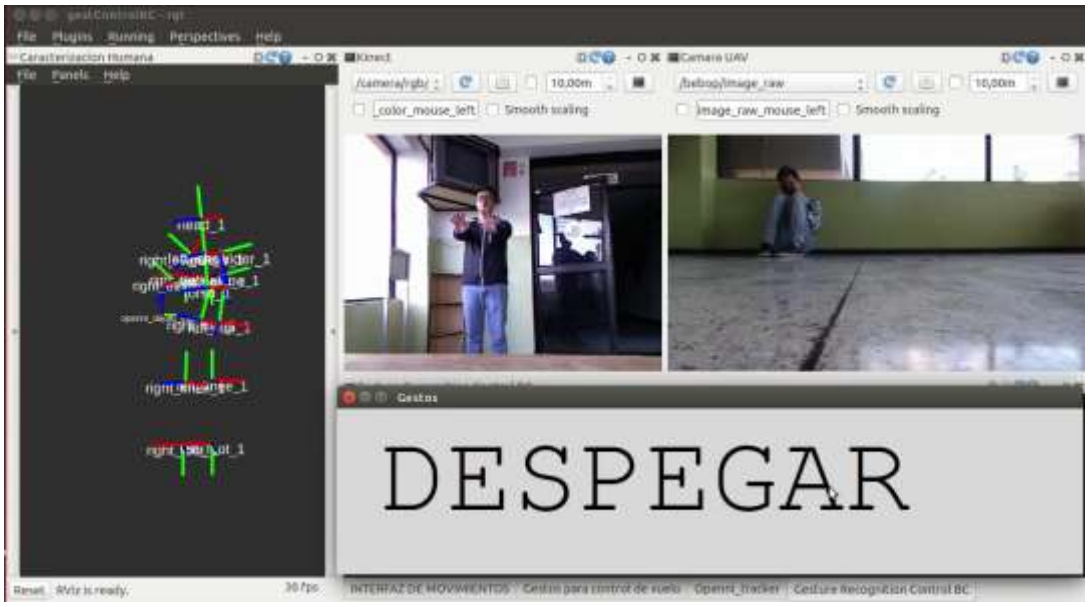


Figura 67. Prueba de la interfaz con bebop drone con instrucción despegar

En la figura 68 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para aterrizar.

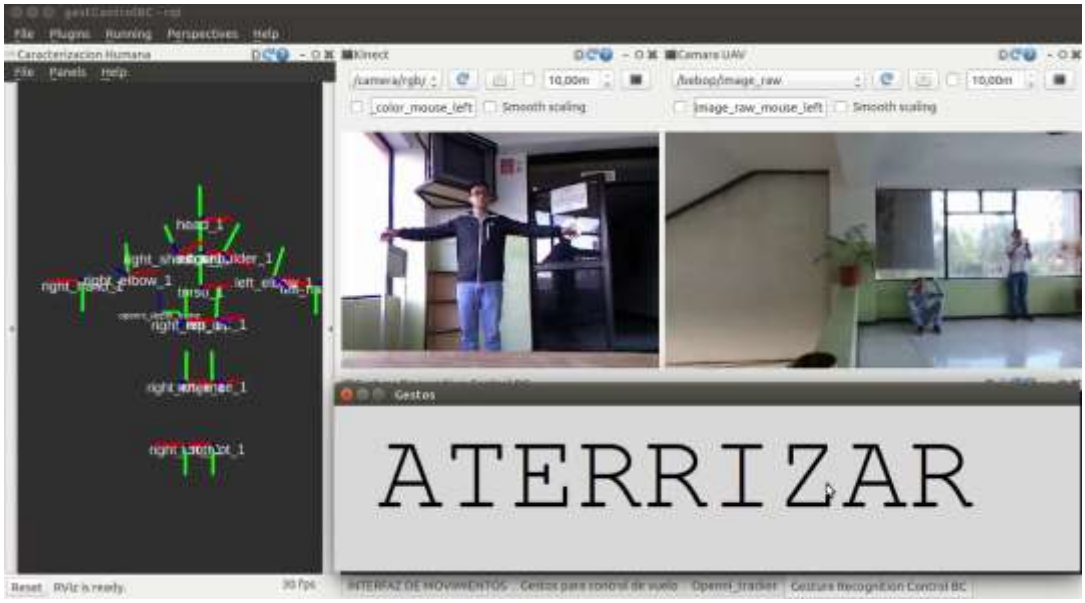


Figura 68. Prueba de la interfaz con bebop drone con instrucción aterrizar

En la figura 69 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para que el drone realice un movimiento hacia arriba.

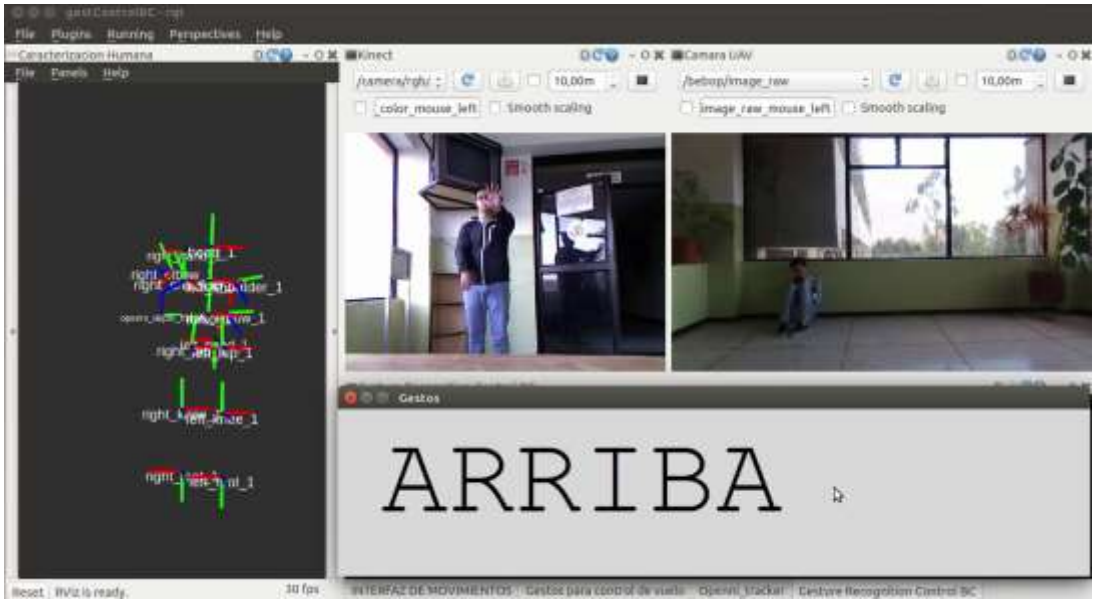


Figura 69. Prueba de la interfaz con bebop drone para mover hacia arriba

En la figura 70 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para que el drone realice un movimiento hacia abajo.

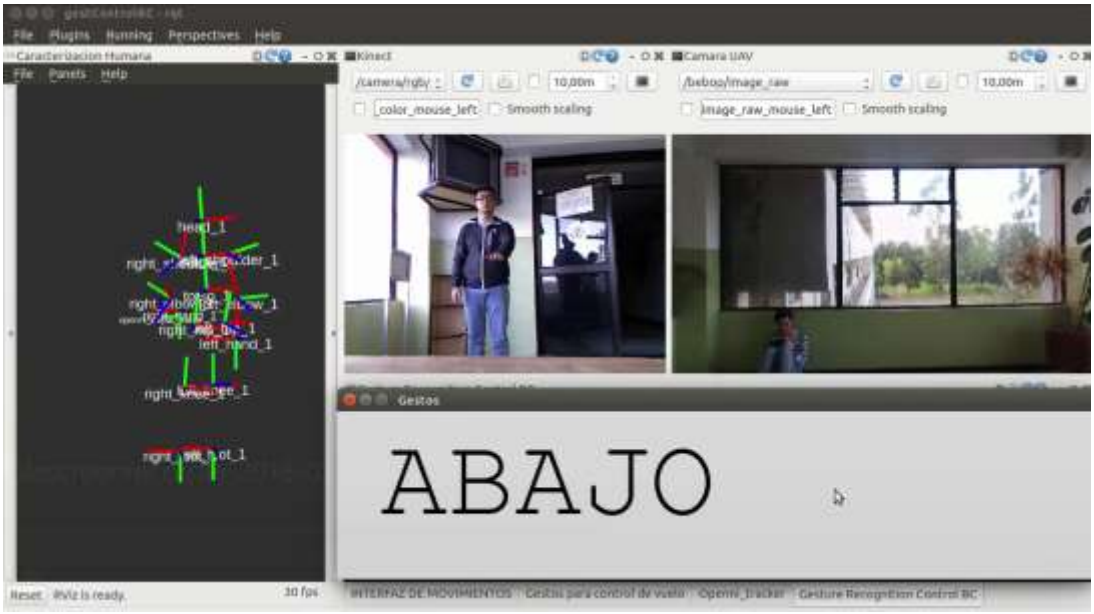


Figura 70. Prueba de la interfaz con bebop drone para mover hacia abajo

En la figura 71 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para que el drone realice un movimiento hacia adelante.

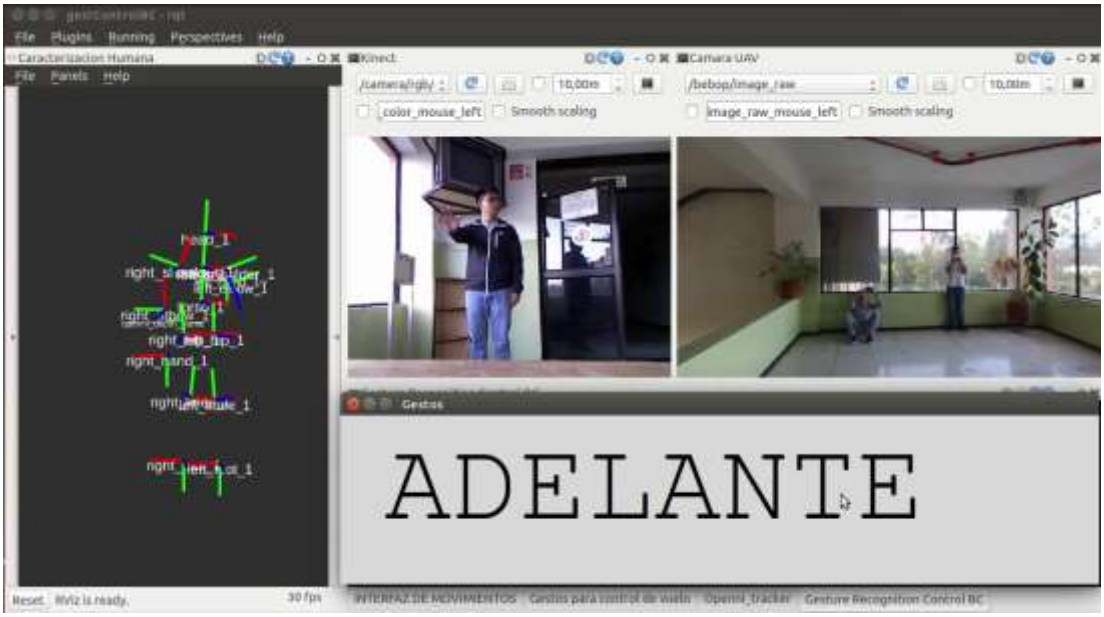


Figura 71. Prueba de la interfaz con bebop drone para mover hacia adelante

En la figura 72 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para que el drone realice un movimiento hacia atrás.

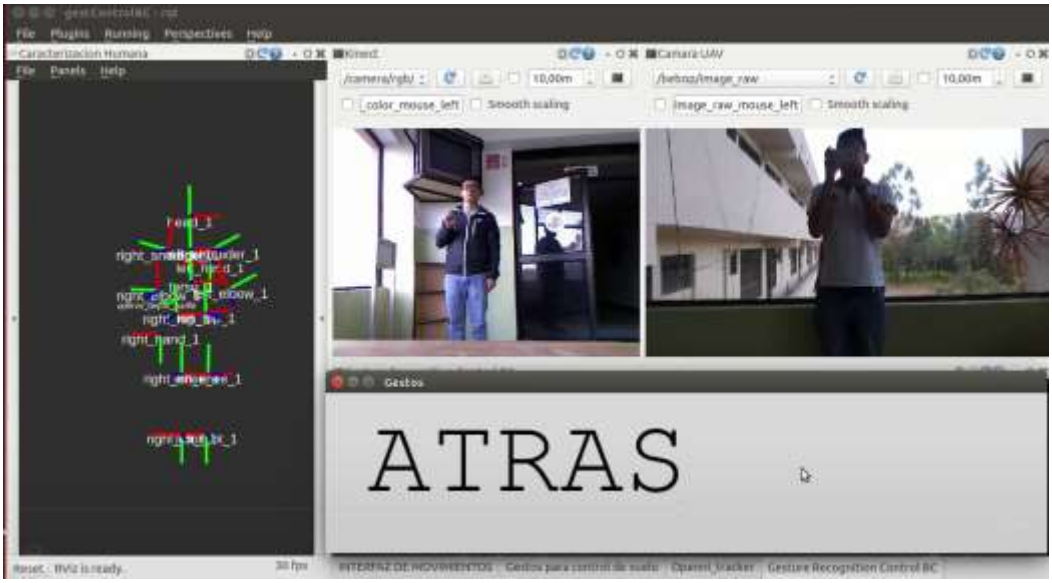


Figura 72. Prueba de la interfaz con bebop drone para mover hacia atrás

En la figura 73 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para que el drone realice un movimiento hacia la derecha.

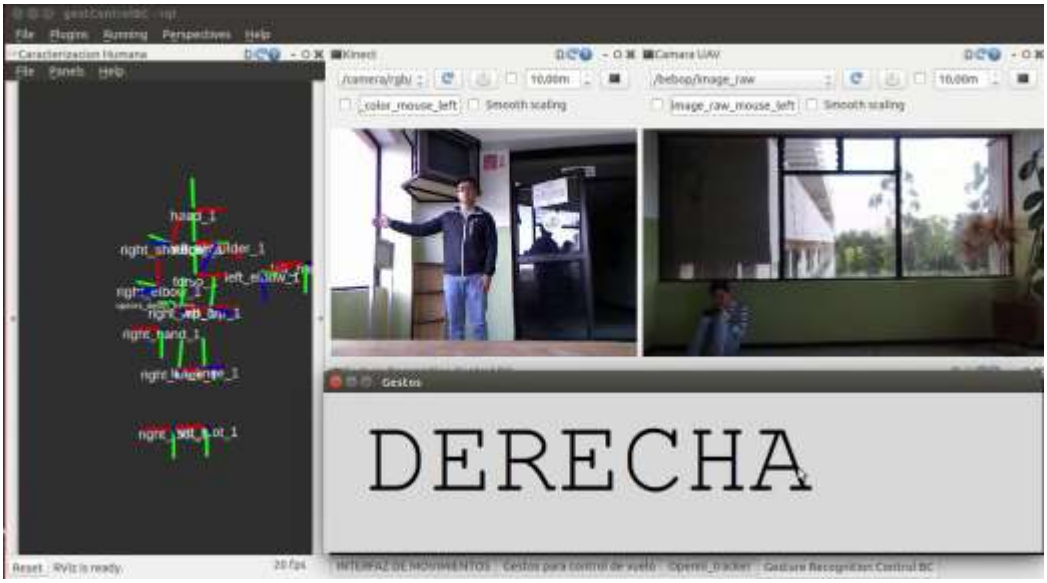


Figura 73. Prueba de la interfaz con bebop drone para mover hacia la derecha

En la figura 74 se muestra la ventana de control de vuelo cuando el usuario realiza el gesto para que el drone realice un movimiento hacia la izquierda.

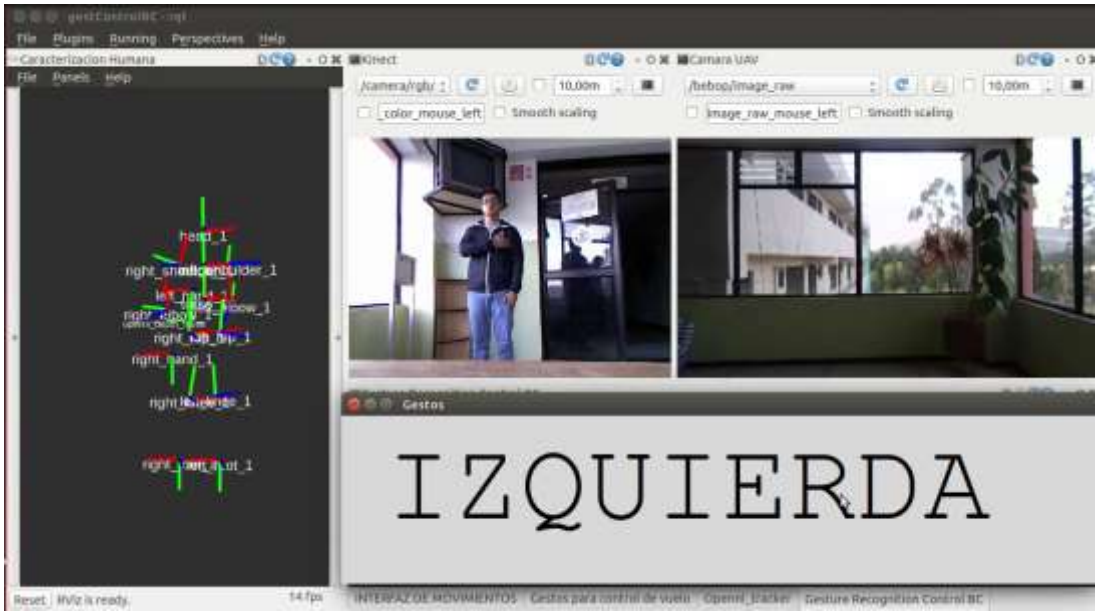


Figura 74. Prueba de la interfaz con bebop drone para mover hacia la izquierda

6.2.1 Prueba del sistema con diferentes usuarios

Esta prueba se consideran usuarios que fueron parte del entrenamiento del sistema, pero también se realizó pruebas con usuarios que no intervinieron en el entrenamiento para probar el funcionamiento del entrenamiento de maquina supervisado.

Estas pruebas tienen una serie de fases para completar que son las que se muestran a continuación:

- Primera fase: Capacitar a los usuarios a utilizar el sistema de reconocimiento de gestos corporales por medio de la ventana de información de la interfaz gráfica de usuario, para que tengan claro los gestos con los cuales se controla el drone

- Segunda fase: Los usuarios prueban el sistema y realizan los gestos para realizar acciones de despegar, aterrizar, emergencia, adelante, atrás derecha, izquierda, arriba y abajo con el drone.

En la Tabla 13 se resumen las pruebas realizadas con un total de 6 usuarios.

Tabla 13.
Prueba del sistema con diferentes usuarios

Usuario	Participo del entrenamiento	Tiempo de uso del sistema (min)	Observaciones
1	Si	3.23	Reconoce todos los gestos
2	Si	1.12	Reconoce todos los gestos
3	Si	2.20	Reconoce todos los gestos
4	No	1.16	Reconoce todos los gestos
5	No	3.15	Reconoce todos los gestos
6	No	4.30	Reconoce todos los gestos

En la figura 75 se muestran pruebas realizadas con diferentes usuarios

Figura 75. Prueba con diferentes usuarios

Al finalizar esta prueba se observa que el sistema de reconocimiento de gestos corporales logra predecir correctamente los gestos de los usuarios, inclusive realiza una correcta predicción de usuarios que no fueron involucrados en el entrenamiento y clasificación mediante SVM.

6.3 Comparación con método tradicional del control

En esta prueba los seis usuarios de la sección anterior realizan el pilotaje del drone por medio de la aplicación de celular Freeflight 3 para poder comparar con el sistema de reconocimiento de gestos corporales. Al finalizar se realiza una serie de preguntas que son las siguientes:

- ¿Con que método se familiarizo más rápido para controlar el drone?
- ¿Cómo le fue más fácil controlar el drone con la aplicación móvil o con el sistema de reconocimiento de gestos?
- Para personas que no están relacionadas con la tecnológica ¿Qué método de control de vuelo recomendaría?

De las respuestas impartidas por los usuarios se tiene los siguientes resultados mostrados en la Figura 76:

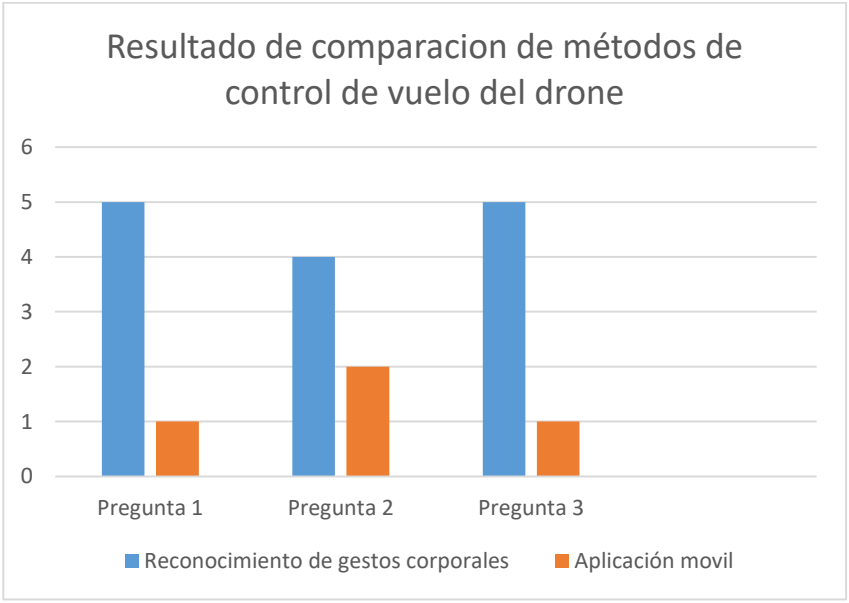


Figura 76. Resultado de comparación de método de reconocimiento de gestos con método tradicional de vuelo

6.4 Trabajos Futuros

Con el fin de hacer más robusto el sistema de reconocimiento de gestos basado en SVM y el sensor RGB-D a futuro se prevé realizar una nueva clasificación y entrenamiento donde participen una mayor cantidad de usuarios para tener un modelo de predicción más robusto.

Entre otras de las aplicaciones que se desea añadir a futuro está el añadir más gestos para el control del sistema con los cuales se pueda realizar acciones involucradas con el entretenimiento como el tomar fotos o empezar a grabar videos, dando mayor utilidad al este sistema.

Debido a que el sistema ha sido probado únicamente con el Parrot Bebop Drone se provee como trabajo a futuro realizar pruebas con otros modelos de micro vehículos aéreos multirotor siempre y cuando cuenten con los paquetes y librerías para que se pueda realizar su comunicación con el sistema operativo ROS.

CAPÍTULO VII

7 CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

Se desarrolló un algoritmo con el cual se puede determinar la posición en x, y, z de puntos de referencia del cuerpo humano utilizando el sensor Kinect y la librería openni tracker completando así la primera fase de la interacción humano máquina que es la extracción de características.

Con el fin de utilizar gestos intuitivos y fáciles de realizar para cada persona con capacidad de análisis se seleccionó diferentes gestos deícticos que representen órdenes de vuelo para poder controlar los cuatro grados de libertad del drone.

Se realizó la recolección de datos de gestos de cinco usuarios para realizar el entrenamiento de maquina supervisado basado en SVM y obtener tres modelos de predicción que representen las ordenes realizadas con el brazo izquierdo, derecho y los dos brazos.

Se realizó la clasificación de los gestos corporales basados en SVM utilizando el kernel en función de base radial luego de analizar que con un menor número de interacciones y de vectores soporte lograba realizar la clasificación, reduciendo así el coste computacional y el tiempo de entrenamiento.

Se diseñó una interfaz gráfica de usuario fácil e intuitiva de utilizar la cual cuenta con una ventana de información donde los usuarios pueden realizar el reconocimiento de los gestos para controlar el drone y una ventana de control de vuelo donde se presenta únicamente información necesaria para que el usuario pueda controlar el sistema de manera adecuada.

Se realizó la comunicación con el bebop drone utilizando un estándar 802.11 a/c también conocido como Wi-Fi donde el drone funciona como router y asigna una dirección IP al computador que se encarga de publicar los mensajes en la red por medio de los tópicos de ROS para que el drone realice los movimientos.

Se desarrolló un paquete de ROS que fue llamado “gestcontrolBC”, el cual contiene los archivos de inicio y los scripts para realizar la ejecución del sistema de reconocimiento de gestos corporales para controlar el bebop drone.

Se realizaron pruebas para probar el funcionamiento del sensor Kinect en las que se pudo determinar que para un funcionamiento ideal la altura a la cual debe estar ubicado el sensor Kinect es de 0.8m y el usuario debe estar a un rango de distancia de 2.0 m hasta los 2.6 m.

Se realizaron pruebas en diferentes intensidades del luz para probar el reconocimiento de los gestos corporales en los cuales se pudo determinar que el valor máximo para que el sistema de reconocimiento de gestos funcione sin inconvenientes es de 1000 luxes.

El sistema fue probado con usuarios que no intervinieron en el entrenamiento del sistema y su funcionamiento fue correcto demostrando allí la potencialidad del aprendizaje de maquina basado en SVM.

Se realizó una comparación con el método tradicional de control de vuelo del bebop drone, por medio de un cuestionario realizado a los usuarios que probaron el sistema y de esto se concluyó que, el sistema de reconocimiento de gestos corporales representa una alternativa ideal para las personas que no están relacionadas con la tecnología, además, los gestos utilizados en el sistema son intuitivos y fáciles de utilizar por lo que les resulto más fácil realizar el pilotaje.

7.2 Recomendaciones

Se recomienda a la hora de utilizar el sistema tener en cuenta el tiempo de autonomía de vuelo del drone es de 12 minutos, para evitar posibles accidentes por la descarga el bebop drone.

Se recomienda probar el sistema de reconocimiento de gestos corporales basado en SVM y sensor RGB-D con otros modelos de micro vehículos aéreos multirotor, siempre y cuando, estén disponibles sus paquetes para el sistema operativo de robótica ROS.

Se recomienda como trabajo a futuro realizar el entrenamiento y clasificación de gestos con una mayor cantidad de usuarios para tener un sistema de reconocimiento de gestos más robusto.

Se recomienda tener en cuenta las pruebas de funcionamiento acerca de la distancia y la intensidad de luz en las cuales funciona de manera adecuada el sensor Kinect para evitar el mal pilotaje del bebop drone.

Se recomienda realizar una capacitación a futuros usuarios del sistema para que tengan claro los gestos con los cuales se controla el drone y evitar accidentes con el drone.

REFERENCIAS BIBLIOGRÁFICAS

Afthoni, Rizal, & Susanto. (2013). Proportional derivative control based robot arm system using Microsoft Kinect. *IEEE*.

Aguilar, M., & Galicia, I. (2016). *Estudios de la lingüística aplicada*. Obtenido de http://ela.enallt.unam.mx/index.php/ela/article/view/625/706

Aguilar, W. G., & Angulo, C. (2012). Compensación de los Efectos Generados en la Imagen por el Control de Navegación del Robot Aibo ERS 7. *Memorias del VII Congreso de Ciencia y Tecnología ESPE 2012*, (págs. 165 - 170).

Aguilar, W. G., & Angulo, C. (2014). Robust Video Stabilization based on Motion Intention for low-cost Micro Aerial Vehicles. *Systems Signals and Devices (SSD), 2014 11th International Multi-Conference* (págs. 1 - 6). IEEE.

Aguilar, W. G., & Angulo, C. (2014). Estabilización de vídeo en micro vehículos aéreos y su aplicación en la detección de caras. *IX Congreso de Ciencia y Tecnología ESPE 2014*, (págs. 155 - 160).

Aguilar, W. G., & Angulo, C. (2014). Real-time video stabilization without phantom movements for micro aerial vehicles. *EURASIP Journal on Image and Video Processing.*

Aguilar, W. G., & Angulo, C. (2016). Real-time model-based video stabilization for micro aerial vehicles. *Neural processing letters*, (págs. 459 - 477).

Aguilar, W. G., & Morales, S. (2017). 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms. *Electronics*, 5(4), 70.

Aguilar, W. G., Casaliglla, V. P., & Polit, J. L. (2017). Obstacle Avoidance for Low-Cost UAVs. *Semantic Computing (ICSC), 2017 IEEE 11th International Conference* (págs. 503 - 508). IEEE.

Aguilar, W. G., Casaliglla, V., & Pólit, J. (2017). Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles. *Electronics*.

Aguilar, W. G., Casaliglla, V., Pólit, J., Abad, V., & Ruiz, H. (2017). Obstacle avoidance for flight safety on unmanned aerial vehicles. *International Work-Conference on Artificial Neural Networks* (págs. 575-584). Springer, Cham.

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Parra, H., & Ruiz, H. (2017). Pedestrian Detection for UAVs Using Cascade Classifiers with Meanshif. *IEEE 11th International Conference on Semantic Computing, ICSC 2017* (págs. 509 - 5014). IEEE.

Aguilar, W. G., Luna, M., Moya, J., Abad, V., Ruiz, H., Parra, H., & Angulo, H. (2017). Pedestrian detection for UAVs using cascade classifiers and saliency maps. *International Work-Conference on Artificial Neural Networks* (págs. 563 - 574). Springer, Cham.

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). RRT* GL based optimal path planning for real-time navigation of UAVs. *International Work-Conference on Artificial Neural Networks* (págs. 585-595). Springer, Cham.

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). RRT* GL based optimal path planning for real-time navigation of UAVs. *International Work-Conference on Artificial Neural Networks* (págs. 585 - 595). pringer, Cham.

Aguilar, W. G., Rodríguez, G. A., & Álvarez, L. (2017). On-Board Visual SLAM on a UGV Using a RGB-D Camera. *International Conference on Intelligent Robotics and Applications* (págs. 298 - 308). Springer, Cham.

Aguilar, W. G., Rodríguez, G., Álvarez, L., Sandoval, S., & Quisaguano, F. (2017). Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing. *International Conference on Augmented Reality, Virtual Reality and Computer* (págs. 410 - 419). Springer, Cham.

Aguilar, W. G., Rodríguez, G., Álvarez, L., Sandoval, S., & Quisaguano, F. (2017). Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing. *International Work-Conference on Artificial Neural Networks* (pág. 596 606). Springer, Cham.

Aguilar, W. G., Salcedo, V. S., Sandoval, D. S., & Cobeña, B. (2017). Developing of a Video-Based Model for UAV Autonomous Navigation. *Latin American Workshop on Computational Neuroscience. Communications in Computer and Information Science* (págs. 94 -105). B. C. Barone D., Teles E. (Ed.): Springer, Cham.

Aguilar, W. G.; Luna, M.; Moya, J.; Abad, V.; Ruiz, H.; Parra, H.; Lopez, W. (2017). Cascade Classifiers and Saliency Maps Based People Detection. *International Conference on Augmented Reality, Virtual Reality and Computer* (págs. 501 - 510). Springer, Cham.

Aprendizaje automático. (2017). Obtenido de https://la.mathworks.com/discovery/aprendizaje-automatico.html

Cai, Q., Gallup, D., Zhang, C., & Zhang, Z. (2010). 3D deformable face tracking with a commodity depth camera. *Computer Vision* (págs. 229-242). Greece: Crete.

Calle, J. L., Balseca, J. M., & Medina, R. P. (2015). Test Wisc IV: Una Mirada desde la herramienta kinect. *Vínculos*.

Carmona, E. (2014). Tutorial sobre Máquinas de Vectores Soporte (SVM). *UNED*.

Chang, C.-C., & Lin, C.-J. (2017). *LIBSVM: Una librería para maquinas de vectores soporte*. Obtenido de https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Chmaj, G., & Selvaraj, H. (2015). Distributed Processing Applications for UAV/drones: A Survey. *Springer*.

Conley, K., & Foote, T. (2017). *ROS Robots*. Obtenido de http://robots.ros.org/

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Springer*.

Ding, I.-J., & Chang, C.-W. (2015). An adaptive hidden Markov model-based gesture recognition approach using Kinect to simplify large-scale video data proccessing for humanoid robot imitation. *Springer Science*.

Erle Robotics. (2017). Obtenido de http://docs.erlerobotics.com/robot_operating_system/ros

Erle Robotics. (2017). Obtenido de http://docs.erlerobotics.com/robot_operating_system/ros/basic_concepts

Estrada, B., & Mera, C. (2017). *Support Vector Machine*. Obtenido de https://www.yumpu.com/es/document/view/14334412/support-vector-machine-ejemplos-aprendest-022011

Gallo, L., Placitelli, A. P., & Ciampi, M. (2011). Controller-free exploration of medical image data: experiencing the Kinect. *24th Interna*.

Gonzáles, W. (2017). *Diseño y contrucción de un vehículo aéreo no tripulado (UAV) del tipo drone cuadricóptero de carreras.* Colombia.

Ho, Y., & Joc. (2013). Challenging technical issues of 3D video processing. *J Converg*.

Ibañez, R. (2014). Evaluación de técnicas de Machine Learning para el reconocimiento de gestos corporales. *XLIII Jornadas Argentinas de Informática e Investigación Operativa*.

Integración de dispositivos Leap y Kinect. (26 de 04 de 2013). Obtenido de http://animusproject.wix.com/web/apps/blog/integraci%C3%B3n-de-dispositivos-leap-y-kinect

Jiménez, R. (2015). Tracking Humano mediante kinect para control de robots. *Clepsidra*.

Kang, H., Woo, L. C., & Jung, K. (2004). Recognition-based gesture spotting in video games. *Elsevier*.

Kurakin, A., Zhang, Z., & Liu, Z. (2012). A Real Time System For Dynamic Hand Gesture Recognition with A Depth Sensor. *Eusipco*.

Kurtulmus, F., & Kavdir, I. (2014). Detecting corn tassels using computer vision and support vector machines. *Elsevier*.

Le, T.-L., Nguyen, M.-Q., & Nguyen, T.-T.-M. (2013). Human posture recognition usinf human skeleton provided by Kinect. *IEEE*.

Lejune, A., Piérard, S., Droogenbroeck, M. V., & Verly, J. (2011). A new jump edge detection method for 3D cameras. *IEEE*.

Mayorga, R. (2009). *Sistema de Navegación para Vehículos Aéreos Cuadricópteros.*

Mcneill, D. (1992). *Hand and mind: What gestures reveal about thought.* The University of Chicago Press.

Microsoft. (2017). *Microsoft Skeletal Tracking*. Obtenido de https://msdn.microsoft.com/en-us/library/hh973074.aspx

Microsoft. (2017). *Seguimiento de usuarios con Kinect Skeletal Tracking*. Obtenido de https://msdn.microsoft.com/en-us/library/jj131025.aspx

Ñecato, D. (2014). *Diseño e implemntacion de un sistema de teleoperacion para controlar un robot humanoide mediante un sensor Kinect.* Ecuador.

Openni Tracker. (2017). Obtenido de http://wiki.ros.org/openni_tracker

Parrot Bebop Drone. (2017). Obtenido de http://global.parrot.com/au/products/bebop-drone/

Parrot Bebop Drone y Skycontroller. (2017). Obtenido de https://www.tecno-renting.com/producto/parrot-bebop-drone-y-skycontroller/

Quigley, M., Gerkey, B., Conley, K., & Faust, J. (2010). ROS: an open-source Robot Operating System. *Computer Science Department, Stanford University*.

Quiroga, F. (2014). *Reconocimiento de Gestos Dinámicos.* Buenos Aires.

Real, L. (2013). *Introduccion a los UAV y una visita al Salón de Seguridad y Defensa "HOMSEC13"*. Madrid.

Ren, Z., Meng, J., & Yuan, J. (2011). Depth Camera Based Hand Gesture Recognition and its Applications in Human Computer Interaction. *IEEE*.

Ren, Z., Meng, J., & Yuan, J. (2011). Depth Camera Based Hand Gesture Recognition and its Applications in Human-Computer-Interaction. *IEEE*.

Ren, Z., Yuan, J., Meng, J., & Zhang, Z. (2013). Robust Part-Based hand Gesture Recognition Using Kinect Sensor. *IEEE*.

ROS Documentation. (2017). Obtenido de http://wiki.ros.org/es

Sandoval, D., & Trujillo, A. (2016). *Sistema replicador de movimiento articular de extremidad superior derecha en brazo robótico industrial por estudio electromiográfico y uso de Kinect.* Sangolquí.

Shotton, J., Fitzgibbon, A., Cook, M., & Blake, A. (2011). Real-time human pose recognition in parts from sigle depth images. *Computer Vision and Pattern Recognition.* Colorado : Colorado Springs.

Weinberfer, K., & Lawrence, S. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 207-244.

Widenhofer, B. (29 de 11 de 2010). *EETIMES*. Obtenido de https://www.eetimes.com/document.asp?doc_id=1281322