uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Electrónica Industrial y Automática

2018-2019

Trabajo Fin de Grado

EMPLEO DE AFFORDANCES EN APRENDIZAJE PARA NAVEGACIÓN SEMÁNTICA

David Mesa López

Tutor

Jonathan Crespo Herrero

Lugar y fecha de presentación prevista: 14/10/2019 09:30

2.3.D05

RESUMEN

En este proyecto se va a presentar un método en el que se van a utilizar las *Affordances* de las acciones humanas, entendidas como las cualidades que permiten a una acción ser identificable, para poder entrenar un sistema que será capaz de categorizar la acción observada. En lugar de centrar los esfuerzos solamente en la detección de la postura corporal al realizar la acción, se utiliza también la detección de objetos, sea cual sea el objeto con el que se realiza la acción, para reconocer dichas características y utilizarlas en la clasificación. Se describe un modelo general que recoge las relaciones espaciales entre las ubicaciones de los puntos de interés tanto del cuerpo humano como del objeto, y se producen muestras que sirven cómo datos de entrenamiento para una máquina de soporte vectorial (SVM) que permite enseñar al sistema a categorizar diferentes acciones. Este modelo tiene varias propiedades notables: Recoge eficientemente todos los datos precisados para el reconocimiento, tiene una alta capacidad de integración, es fácil de utilizar y es capaz de tener altos porcentajes de éxito en la diferenciación de 5 acciones. Se demuestra cómo la pose humana y la localización de objetos basta en el reconocimiento de las *Affordances* de las acciones probadas, y se explica el procedimiento experimental seguido para que pueda ser probado con diferentes acciones. Se presentan los resultados experimentales sobre la categorización automática de acciones diferenciadas realizadas, en ocasiones, con el mismo objeto y se plantean diferentes caminos para poder mejorar nuestro proyecto.

Palabras clave: Affordance, Aprendizaje automatico, SVM, Reconocimiento cuerpo humano, Detección de objetos

ABSTRACT

In this project a method will be presented in which the *Affordances* of human actions will be used, understood as the qualities that allow an action to be identifiable, in order to train a system that will be able to categorize the observed action. Instead of focusing efforts solely on the detection of body posture when performing the action, object detection is also used, whatever the object with which the action is performed, to recognize these characteristics and use them in the classification. A general model is described that gathers the spatial relationships between the locations of the points of interest of both the human body and the object, and samples are produced that serve as training data for a vectorial support machine (SVM) that allows the system to learn how to differentiate different actions. This model has several remarkable properties: it efficiently collects all the data required for recognition, it has a high integration capacity, it is easy to use and it is capable of having high percentages of success in the differentiation of the 5 actions. It shows how the human pose and the location of objects suffices in the recognition of the Affordances of the tested actions, and explains the experimental procedure followed so that it can be tested with the different actions. The experimental results are presented on the automatic categorization of differentiated actions carried out, sometimes, with the same object, and different ways are proposed to improve our project.

Keywords: Affordance, Machine learning, SVM, Estimate human pose, Object detection

AGRADECIMIENTOS

Al prof. Jonathan Crespo Herrero por su guía en cada uno de los pasos de este proceso, el tiempo dedicado y su muy buena voluntad y disponibilidad. Gracias por ayudarme a entender como funciona la investigación científica y por haberme entendido en los pasos mas complicados de este trayecto. Su paciencia, su calma y su perfeccionismo le convierten en un ejemplo para mi.

Gracias a mi familia por ayudarme a superar los momentos mas difíciles y hacerme entender que era lo que estaba haciendo y que camino debía tomar.

Por último, gracias a mis amigos por ayudarme a avanzar y levantarme cuando nadie mas lo hacia, y a aliviar esos momentos de incertidumbre y presión que no me dejaban proseguir mi estudio.

Sin la ayuda de todos vosotros había sido complicado alcanzar el resultado obtenido, gracias.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En la vida cotidiana de cualquier persona se realizan un número muy elevado de acciones diferentes. Nosotros cómo seres humanos, de manera general, visualizamos estas acciones, y de manera inconsciente, a través de un análisis de las señales de movimiento biológico las categorizamos según los movimientos o características que las definen[1]. En el amplio mundo de la robótica, una de las diferentes cuestiones estudiadas en el marco actual es la de poder detectar y/o anticipar estas acciones; pues ser capaces de hacerlo en un entorno complejo, es una de las condiciones que puede ser importante en la interacción humano-máquina. Con esta cualidad un robot será capaz de predecir y planificar diversas acciones que proporcionen servicios en concreto [2], o que permitan incluso detectar y evitar diferentes tipos de accidentes o emergencias[3].

Dichas cualidades requieren de diferentes métodos de razonamiento y percepción en el cerebro humano. La reproducción artificial de esos métodos con las herramientas que el mundo actual proporciona, es uno de los múltiples retos actuales a los que la comunidad científica se enfrenta. El reconocimiento de estas acciones podría ser separado en tres diferentes procesos que permitiesen modular y simplificar dicho problema, de manera muy resumida:

1. Reconocimiento de entornos: Es importante ser capaz de categorizar en qué entorno se está realizando una determinada acción[4].

2. Reconocimiento de las affordances de las acciones: Es importante lograr que el robot sepa analizar las cualidades intrínsecas que definen las propias acciones que está observando.

3. Reconocimiento de la postura humana: Es importante que el robot sea capaz en todo momento de detectar la posición del humano al que observa, ya sea por el análisis de la acción que está realizando o para utilizar este parámetro en las acciones que llevará a cabo[5].

1.1. Motivación del trabajo

En este proyecto se pretende contribuir con un granito de arena a ese amplio campo de investigación del reconocimiento de *Affordances* para la navegación semántica. Se quiere participar en el avance de la ciencia dentro del campo de estudio de la interacción hombre-máquina, abordando la predicción de acciones humanas a través de un simple muestreo y análisis de los datos recogidos por instrumentos de percepción visual.

Este proyecto por lo tanto se centra en la creación de un sistema que ayude a la predicción de acciones y pueda participar en la interacción entre cierto robot social y el ser humano. Además se pretende que el sistema pueda aprender a identificar utilidades diferentes que puedan ser de ayuda para categorizar semánticamente un entorno en función de las acciones que pueden ser realizadas con los objetos existentes en él.

1.2. Objetivos del trabajo

Se va a aportar un enfoque en la navegación semántica a través del reconocimiento de las características intrínsecas o *Affordances* de las acciones que el ser humano realiza. El objetivo principal es investigar si es posible reconocer dichas características implementando un código en ROS [6](C++). Para ello vamos a utilizar una Asus Xtion Pro Live para recibir imágenes del entorno natural, estas imágenes serán procesadas por dos aplicaciones diferentes. La primera es capaz de proporcionar la localización de los puntos corporales trascendentes del ser humano y su orientación en el espacio. La segunda es capaz de proporcionar la localización y orientación del objeto que vamos utilizar en nuestra acción a través de una etiqueta.

Una vez recogidos estos datos, nuestro software es capaz de crear muestras que definan dichas acciones. Esta información se valida y agrupa con condiciones temporales, pues la información que llegue del reconocimiento del cuerpo humano y la información derivada del reconocimiento de objetos, ha de ser simultánea. Para la categorización de estas muestras se va a utilizar un técnica de aprendizaje automático.

Además de nuestro objetivo principal se han concentrado esfuerzos en diferentes objetivos secundarios, que han sido importantes de alcanzar para el éxito en la realización de la tesis:

- Encontrar la herramienta o sistema de aprendizaje que mejor se ajuste para la predicción en la categorización de las acciones que se pretenden reconocer.
- Elegir el hardware necesario para la implementación de la idea en investigación.
- Lograr recibir con éxito las imágenes de nuestro hardware y crear muestras de las mismas en el formato que las aplicaciones que se van a utilizar demanden, y así poder entrenar el sistema creado.
- Desde el principio se ha querido que nuestro sistema sea integrable para poder ser fácilmente probado en diferentes plataformas, es por ello que se ha desarrollado la aplicación en ROS.

1.3. Estructura del trabajo

Este trabajo va a estar definido de la siguiente manera: en la sección 2 se estudia el avance científico en las diferentes tecnologías que se van a utilizar, en la sección 3 se puede encontrar el funcionamiento conceptual del sistema implementado y se presentan las herramientas utilizadas, en la sección 4 se desarrolla todo el funcionamiento completo del sistema y los requerimientos precisados por el mismo, en la sección 5 se encuentran enumerados los diferentes experimentos llevados a cabo y sus propios resultados, y por último, en las sección 6 se concluye el trabajo incluyendo las reflexiones suscitadas y los trabajos futuros ideados.

2. ESTADO DEL ARTE

En esta sección del trabajo se va a tratar de mostrar los avances más importantes que se han logrado con respecto al tema tratado. Debido a la necesidad de la utilización de diversas tecnologías, este estado del arte se divide en cuatro puntos; así se puede entender de mejor manera qué son los *Affordances*[7], a la vez que se investigan los campos de la detección de objetos, la detección de la postura corporal y parte del conocimiento actual dentro del campo del aprendizaje automático.

2.1. Affordances

Affordance es un término utilizado en diferentes campos como la psicología cognitiva, la inteligencia artificial o los sistemas de interacción hombre-máquina. Apareció por primera vez en 'The theory of Affordances'[7] y fue descrito cómo *Todas las posibilidades que materialmente ofrece un objeto para reconocer cómo usarlo*. Hoy en día, referido al campo de la interacción hombre-máquina, se puede entender un affordance como esa posibilidad de acción que determinado objeto inspira inmediatamente en el usuario que lo percibe. Así, por ejemplo, un ser humano ante una lanza y una silla podría tomar la decisión de sentarse en la lanza o de lanzar la silla, puesto que es objetivamente posible. En cambio, hay una alta probabilidad de que lance la lanza y se siente en la silla. La cualidad que le hace a la lanza ser *lanzable* y a la silla ser *sentable* es a lo que llamamos affordance. Estas affordances las aprende temporalmente el usuario por estímulos inconscientes que recibe o por la propia experiencia vital, pero existen, y la ciencia las utiliza [8] para poder tomar partido en la interacción de la máquina con el mundo del que se rodea. William W. Gaver describe que se pueden diferenciar las affordances de tres maneras diferentes [9]; esta distinción es importante puesto que existirán affordance válidas y no válidas en los objetos:

1. Falsas: Este tipo de affordance se crea cuando el usuario piensa que es capaz de interaccionar de una determinada manera con cierto objeto cuando realmente no lo es. Hay que tener mucho cuidado reconociendo este tipo de affordances para poder

prevenir acciones no deseadas del sistema propuesto.

2. Ocultas: Este fenómeno se produce cuando el usuario no es capaz de percibir una interacción que realmente existe. La presencia de este tipo de affordance solo son detectables a través de la experimentación, siendo importante convertirlas en perceptibles para que nuestro sistema funcione de la manera deseada.

3. Perceptibles: Son las affordances que ofrecen un vínculo directo entre la percepción del objeto y una acción a realizar con el mismo.

El estudio de las affordances y los diferentes métodos informáticos para identificarlas de manera artificial, ha permitido abrir un campo enorme en el mundo de la robótica. En estudios como [10], donde se establecen exitosamente relaciones entre escenarios y acciones realizadas en ellos, [11], donde se utilizan dichas características para predecir exitosamente la acción que va a llevar a cabo un ser humano, o [12], donde se obtuvieron unos resultados positivos en el reconocimiento de los affordance de los objetos con ayuda de una red neuronal, se puede ver cómo la comunidad científica utiliza la detección de los affordances de los objetos en sus aplicaciones. Uno de los logros de la comunidad científica respecto a la inclusión del concepto de affordance en sus estudios ha sido el de mejorar los resultados de la predicción y reconocimiento, como se puede ver en los estudios anteriormente referenciados.

2.2. Detección de objetos

La detección y reconocimiento de objetos es un área de estudio dentro del amplio campo de la visión artificial o *Computer vision*. Cuando un ser humano ve un objeto, un animal o una fotografía, es inmediatamente capaz de definir con rapidez que es lo que esta observando de manera natural. El objetivo es dotar a un sistema informático de esta capacidad innata en el ser humano y lograr que adquiera cierta comprensión frente a la observación de imágenes. En los últimos años se han implementado diferentes sistemas en los que se combinan diferentes técnicas en la detección de objetos; véase [13] donde se hace una combinación contextual de localización y clasificación que muestra que la clasificación puede mejorar la detección y viceversa, además de proponer un método eficaz de localización de objetos con ventanas correderas en dos etapas que combina la

eficacia de un clasificador lineal con la robustez de un sofisticado método no lineal, y [14], donde se propone un método para identificar y localizar objetos en imágenes. En lugar de operar a nivel de píxeles, se aboga por el uso de superpíxeles cómo unidad básica de una segmentación de clases. La detección de objetos podría clasificarse en dos subtareas:

1. Localización de objetos: En esta tarea el sistema es capaz de identificar la ubicación y orientación de un objeto en el espacio. Un ejemplo de herramienta utilizada en este contexto es el software de libre distribución de VTT [15], que permite a través de etiquetas pegadas en los objetos, localizar la ubicación y orientación del mismo. En [16] se puede ver cómo se estudia la localización de objetos en la vía para facilitar la conducción de los coches autónomos.

2. Clasificación de objetos: Esta tarea se centra en predecir la clase de un objeto por su forma y poder atribuirle una etiqueta [17]. Dentro de este campo cabe mencionar la segmentación[18], esta técnica consiste en dividir las imágenes por grupos de píxeles y así crear imágenes más fáciles de analizar. Así es posible dividir los objetos y diferenciar las partes que permiten reconocerlo, de las partes que no ofrecen información valiosa a la hora de la categorización. En la figura 2.1 se observa como un software de segmentación es capaz de dividir en partes al objeto para identificarlo fácilmente.

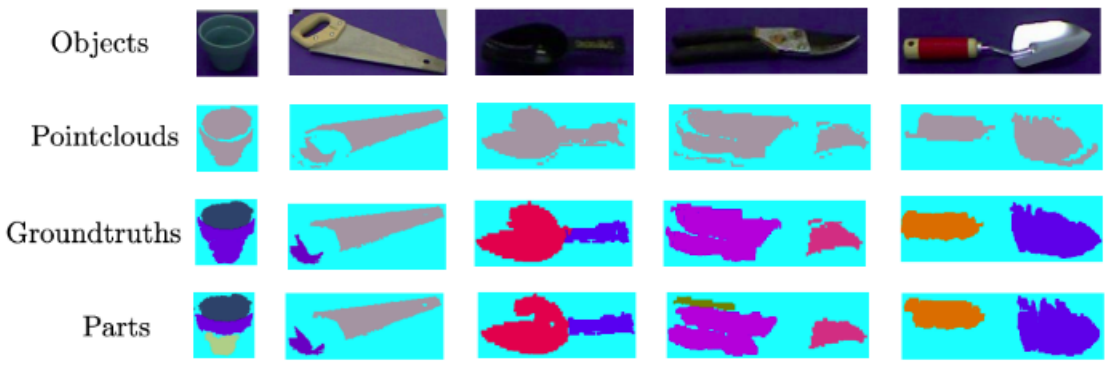


Fig. 2.1. Ejemplo de segmentación de objetos [19]

2.3. Detección de postura humana

La detección fiable de la postura del cuerpo humano a partir de imágenes ha sido un reto científico importante durante décadas debido a la alta empleabilidad de los sistemas

capaces de realizar dicha tarea con éxito. La capacidad de los seres humanos para percibir a otros seres humanos, es una cualidad que los robots podrían poseer. El lanzamiento en los últimos años de diferentes sensores de profundidad capaces de captar imágenes a tiempo real ha simplificado el reto [20], es por ello que hoy en día están a disposición un gran número de herramientas informáticas que permiten cumplir con esta misión; como skeleton_tracker, Nuitrack o Skeltrack.

La idea común en los algoritmos que tratan de dar solución a dicho problema, se encuentra en el intento de crear un módulo que simule las articulaciones y puntos principales del esqueleto un ser humano. Generalmente cada uno de estos puntos es conectado al punto más próximo, generando así una réplica aproximada de un esqueleto humano. En la figura 2.2 observa como un software es capaz de recoger los puntos del cuerpo en el espacio y dar como salida la posición de los mismos.

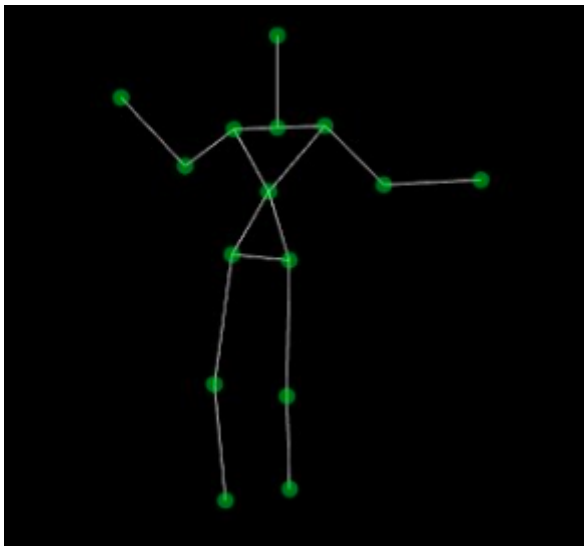


Fig. 2.2. Simulación del esqueleto humano

Actualmente existen tres métodos principales para el reconocimiento del esqueleto humano:

- Marco de estructuras pictóricas: La idea de este método es representar un todo a través de la suma de sus partes dispuestas en una configuración deformable. Cuando el sistema es capaz de determinar la ubicación y orientación de mencionadas partes, la estructura resultante y sus uniones, podrán representar la figura del esqueleto humano. Una de las herramientas que existe en este marco es un software de libre

distribución basado en los drivers para cámaras de Openni [21], capaz de recoger el modelo del ser humano a través de 15 puntos.

- Modelo de partes deformables: En estos modelos las partes del esqueleto humano vienen definidos por una plantilla que incluye plantillas de piezas que las definen, así conseguimos definir de mejor manera las articulaciones que en el modelo anterior. En [22] se observa como se detectan unidades deformables complejas mediante dicho método.

- Métodos basados en *Deep Learning*: El gran problema de los métodos clásicos es que modelan una parte pequeña de las interacciones de todo el cuerpo. Las ventajas principales que se observan en estos modelos son; primero que son capaces de capturar el contexto completo de todas las articulaciones del cuerpo y en segundo lugar no es necesario diseñar explícitamente una topología para cada parte del modelo, al contrario, los sistemas aprenden de manera completa a detectar el cuerpo humano como conjunto. En [23] se encuentra uno de los primeros estudios sobre el *Deep Learning* en este campo, donde se propone un método para la estimación de la pose humana a través de redes neuronales multicapa.

2.4. Aprendizaje automático

En el momento en el que se trata de diseñar la mayoría de robots o sistemas interactivos con el mundo real, se piensa en un producto independiente que sea capaz de aprender por sí mismo a diferenciar las circunstancias operativas en las que se va a encontrar. En la comunidad científica se denomina aprendizaje automático o *Machine learning* al estudio de los algoritmos y modelos estadísticos que permiten lograr que nuestros sistemas sean capaces de realizar una determinada tarea sin recibir ningún tipo de instrucción previa, es decir, de manera automática. Estos métodos dotan a nuestros sistemas de la capacidad de analizar una cantidad de datos masivos para poder predecir el comportamiento de diferentes variables controladas o no controladas que el usuario quiera monitorizar.

2.4.1. Modelos aplicados al aprendizaje automático

A continuación, se estudiarán 4 de los modelos que se utilizan en el contexto científico actual; obteniendo así una visión general[24]:

- Redes neuronales: La idea tras la implementación de una red neuronal es que muchas neuronas pueden unirse por enlaces para poder llevar a cabo cálculos complejos. Lo más común es utilizar un gráfico representativo en el que ilustramos dichas neuronas como nodos y cada una de las direcciones emergentes de dichos nodos son las conexiones que enlazan la salida de este nodo a la entrada del siguiente. Cada una de las neuronas puede definirse a través de una función escalar de diferente tipo, así la entrada de una neurona se obtendrá con el sumatorio de las salidas conectadas a ella. Se puede diseccionar esta red neuronal en tres capas según la finalidad de sus componentes; la capa inferior o de entrada, la capa escondida y la capa de salida. La capa inferior está formada por aquellos nodos que recogen la muestra del entorno, la capa escondida es la más profunda pues es la que determina la dimensión de la red, es aquella capa que conecta la entrada y la salida y que se encarga de dimensionar las operaciones necesarias para simplificar la toma de decisiones en un único sumatorio que le proporciona a una única neurona que forma la capa de salida. Esta neurona tiene una salida, que a su vez es la salida de la propia red neuronal. En la figura 2.3 se puede observar el funcionamiento orientativo de una red neuronal sencilla.

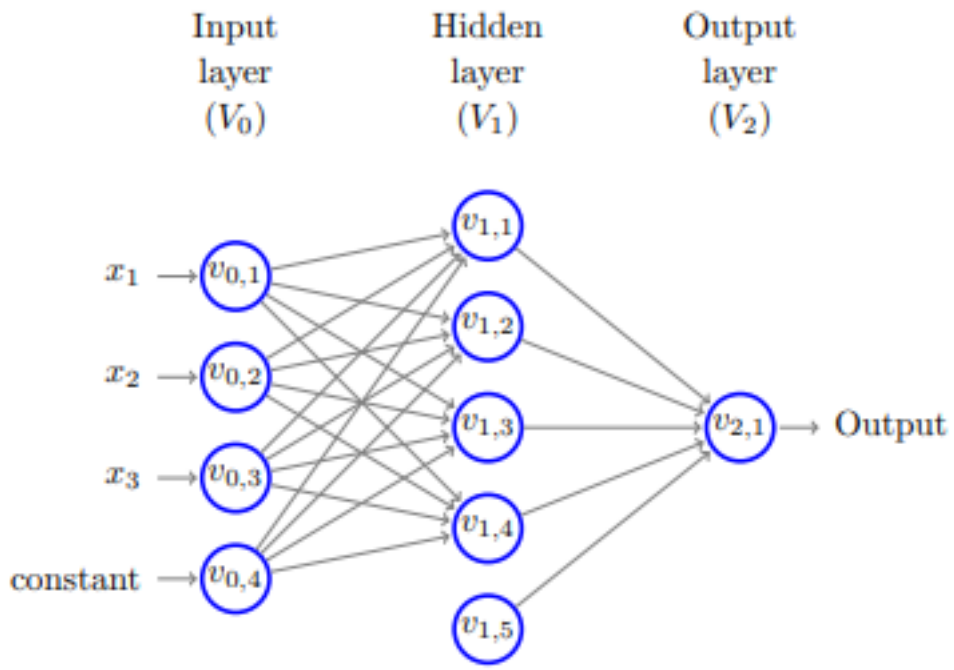


Fig. 2.3. Esquema de una red neuronal

Las redes neuronales son muy útiles para aplicaciones en las que se requiere algún tipo de aprendizaje profundo o *Deep Learning*[25]. Entre sus diferentes posibles aplicaciones se puede ver cómo se utiliza con éxito una red neuronal como apoyo para la predicción del comportamiento de conductores[26] o cómo se utiliza una red neuronal recurrente para la auto-predicción de comportamiento humano [27].

- Máquinas de soporte vectorial: El SVM es un tipo de aprendizaje basado en semi-espacios que utiliza algún tipo de conocimiento previo. Este modelo utiliza sets de datos de entrenamiento para lograr crear el modelo matemático que podrá diferenciar las distintas etiquetas que introduzca el usuario.

  Siendo *S = (x1, y1), . . . ,(xm, ym)* un conjunto de muestras, donde ym puede coger los valores +1,-1, se dice que es un conjunto categorizable donde ym serán las etiquetas para la muestra de datos. De manera más concreta el modelo matemático tiene como entrada nuestro set de datos y devuelve como salida un hiper-plano que es capaz de separar dichos datos de entrada con el mayor margen posible en categorías, a través de la siguiente formulación cuadrática; donde w viene definido por el mínimo conjunto de vectores que separan los datos.

$$(\mathbf{w}_0, b_0) = \underset{(\mathbf{w},b)}{\operatorname{argmin}} \|\mathbf{w}\|^2 \text{ s.t. } \forall i,\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

Una vez creado este modelo la máquina de soporte vectorial adquiere la capacidad para predecir la clase a la que correspondería una nueva entrada, ubicándola dentro de un súper plano u otro. En la figura 2.4 se muestra de manera más intuitiva cómo una máquina de soporte vectorial categoriza dos tipos de datos en dos hiper-planos.

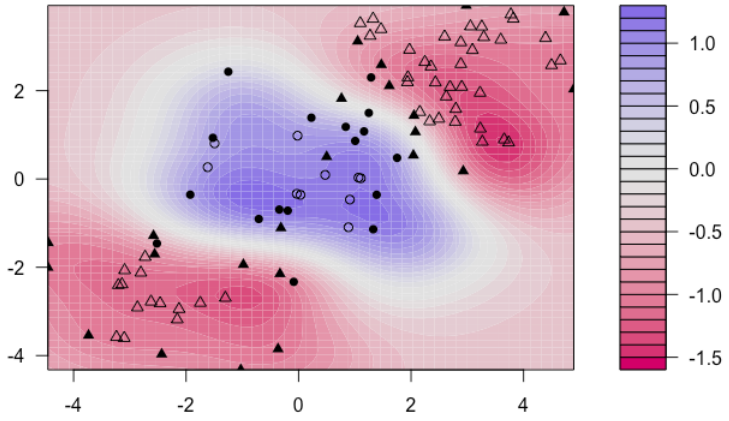


Fig. 2.4. Ejemplo de categorización en una SVM

En el contexto científico actual hay una multitud de proyectos de diferentes campos de estudio en los que una máquina de soporte vectorial forma parte. En [28] se puede observar cómo se utiliza dicha aplicación en el reconocimiento de actividades humanas en una casa inteligente utilizando una máquina de soporte vectorial para la clasificación, en [29] se utiliza una SVM para el reconocimiento facial entre diferentes personas humanas y en [30] se puede ver un artículo explicando cómo una SVM puede ser de gran ayuda en el campo de la bioinformática.

- Modelo oculto de Márkov: El modelo oculto de Márkov es un modelo estadístico que se puede considerar como una red bayesiana simple, su objetivo es encontrar los parámetros ocultos en un sistema a través de los parámetros observables. El método Bayesiano es un modelo probabilístico que presenta un cierto número de variables al azar y sus dependencias condicionales con el resto de variables. Cada variable estaría representada por un nodo, el cual vendría definido por una función de probabilidad que toma como entrada las probabilidades de los nodos padres, y

tras aplicarle su función probabilística devuelve la probabilidad representada por el propio nodo. En [31] se puede ver cómo se utiliza una red dinámica bayesiana para modelar actividades humanas a partir de una multitud de datos que categorizan los estados de los usuarios en tiempo real desde streams de vídeo, audio e interacción con los ordenadores (teclado y ratón).

- Clustering: Esta técnica es ampliamente utilizada en el análisis exploratorio de datos, ya que es una técnica ideal para lograr obtener los datos más significativos dentro de tus muestras. De manera intuitiva se puede entender esta técnica como un proceso que permite agrupar un conjunto de objetos de tal manera que objetos similares terminan en el mismo grupo y los objetos diferentes en grupos diferentes. El problema principal de este método es que si, por ejemplo, se tiene un número grande de datos, los datos pueden ser muy parecidos a sus vecinos dentro del mismo *Cluster* pero muy diferente de un dato más alejado dentro de esa misma secuencia. Otro problema relacionado con este método es que al ser una técnica de aprendizaje automático no supervisado, no existen etiquetas en lo que deseamos predecir y no existe ningún procedimiento claro para saber si se ha tenido éxito en nuestro proceso de agrupación de las muestras. Es difícil cerciorarse de haber hecho bien las cosas dentro del aprendizaje no supervisado, pues existen diferentes características implícitas de similitud entre objetos que dan como resultados diferentes *clusters* de salida, cómo se ve en la figura 2.5, que muestra de manera muy intuitiva el principal problema de esta técnica.

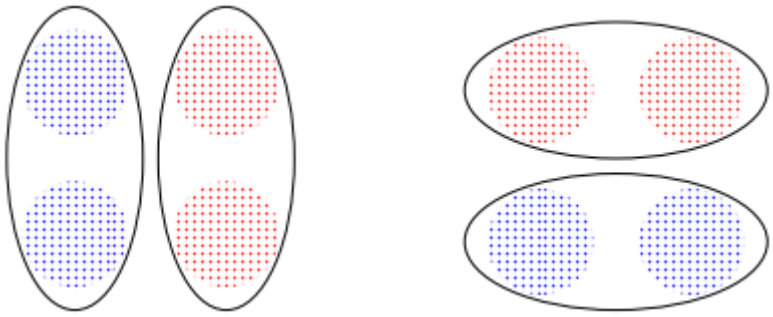


Fig. 2.5. Diferentes *clusters* para una misma muestra

Aún con los problemas que estos métodos presentan se pueden utilizar para diferentes aplicaciones, por ejemplo en [32] se puede ver como estos *clusters* se utilizan

como entrada de agrupación de una máquina oculta de Markov capaz de reconocer diferentes actividades. También se puede ver la utilización de este método dentro de la minería de datos [33].

3. ARQUITECTURA DEL SISTEMA

En este capítulo se va a explicar de manera conceptual el diseño y la implementación de la solución desarrollada en este trabajo. Posteriormente en el capitulo 4, se describe con mayor profundidad el funcionamiento de todas las diferentes herramientas necesitadas en la implementación del proyecto. El sistema está diseñado para reconocer acciones humanas, en la figura 3.1 se puede ver un simple ejemplo de ejecución en el que nuestro sistema se ha probado.

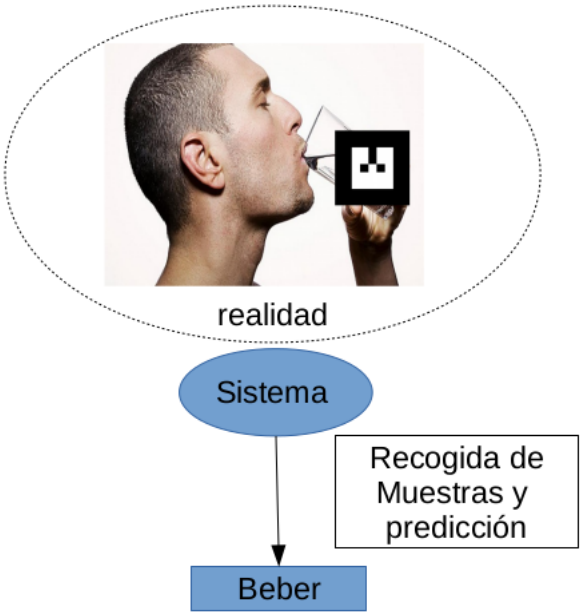


Fig. 3.1. Ejemplo de ejecución del sistema

Para poder llevar a cabo el objetivo del proyecto se han precisado de diferentes herramientas utilizadas a lo largo de todo el trabajo, estas herramientas de software son necesarias en el sistema que desee utilizar nuestra aplicación; además de todas aquellas que demanden a la hora de la instalación. Las herramientas principales utilizadas se pueden ver en la tabla 3.1.

SOFTWARE	UTILIDAD
ROS	Framework del sistema
Openni2	Drivers necesarios para la Asus
NiTe	Librería necesaria para el funcionamiento de cualquier modulo openni
tf2 y tf	Paquetes de ROS necesarios para utilizar macros en el espacio
geometry_msgs	Paquetes de ROS que definen los tipos de dato utilizados en la orientación espacial
openni2_camera	Paquete necesario para utilizar cualquier tipo de cámara Asus
perception_pcl	Point Cloud Library para aplicaciones en 3D
ar_track_alvar	Detección de objetos
skeleton_tracker	Detección del cuerpo humano
libsvm	Librería SVM utilizada

TABLA 3.1. TABLA DE SOFTWARES UTILIZADO

3.1. Estructura

De manera general es necesario un método que permita recoger simultáneamente muestras identificativas de las *Affordances* de las acciones y luego utilizar algún tipo de herramienta que permitiese identificarlas y clasificarlas. En la figura 3.2 se observa cómo funcionará de manera orientativa el sistema.

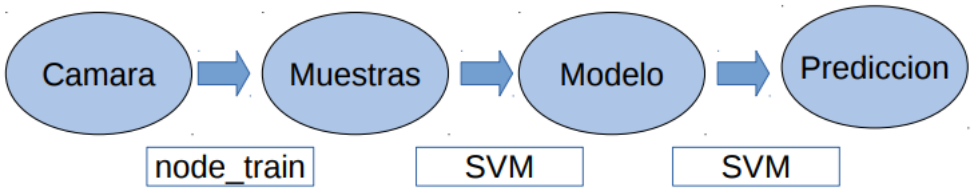


Fig. 3.2. Esquema conceptual del sistema

Tras escoger los software requeridos entre los disponibles en código abierto en internet, es preciso implementar código propio para poder utilizarlos. A continuación se define de manera breve y resumida el funcionamiento y objetivo principal de los diferentes módulos que aparecen en las figuras 3.3 y 3.4, para posibilitar el entendimiento de las mismas. Como se ha dicho anteriormente, en la fase de desarrollo se explica más concretamente el funcionamiento de cada etapa.

- skeleton_tracker: Este software lee las imágenes RGB-D de nuestra Asus y publica en un topic de ROS, un array de transformadas de /tf con la posición y la orientación de 15 puntos del ser humano, además publica también los nodos relativos a la imagen de la cámara y su información.

- ar_track_alvar: Este software lee las imágenes RGB del topic publicado por skeleton_tracker, con la frame de la cámara como referencia. En las imágenes es capaz de reconocer una etiqueta que va a estar adosada en el objeto que se requiere localizar y publica la posición y orientación de dicha etiqueta

- SVM_*: Todas las etapas que comiencen con el prefijo SVM_, son parte de la librería que se va a utilizar para crear el modelo utilizado en el aprendizaje automático[34]. En concreto, SVM_scale ayuda a escalar nuestros datos, SVM_train crea el modelo de entrenamiento y SVM_predict hace la predicción de la acción observada.

- tf_listener: Código implementado requerido para poder acceder a los datos de /tf.

- node_train: Código implementado requerido para escribir las muestras simultáneas en un fichero.

- ForTest: Código implementado requerido para dividir las muestras obtenidas.

- resultAnalize: Código implementado requerido para ayudar a construir de manera automática la matriz de confusión.

3.1.1. Recogida de muestras

Es necesario un sistema de recogida de muestras que obtenga de manera sincronizada los datos y poder así crear los datos de entrenamiento. En la figura 3.3 se puede observar el funcionamiento conceptual de dicho sistema, más adelante se explica el proceso:

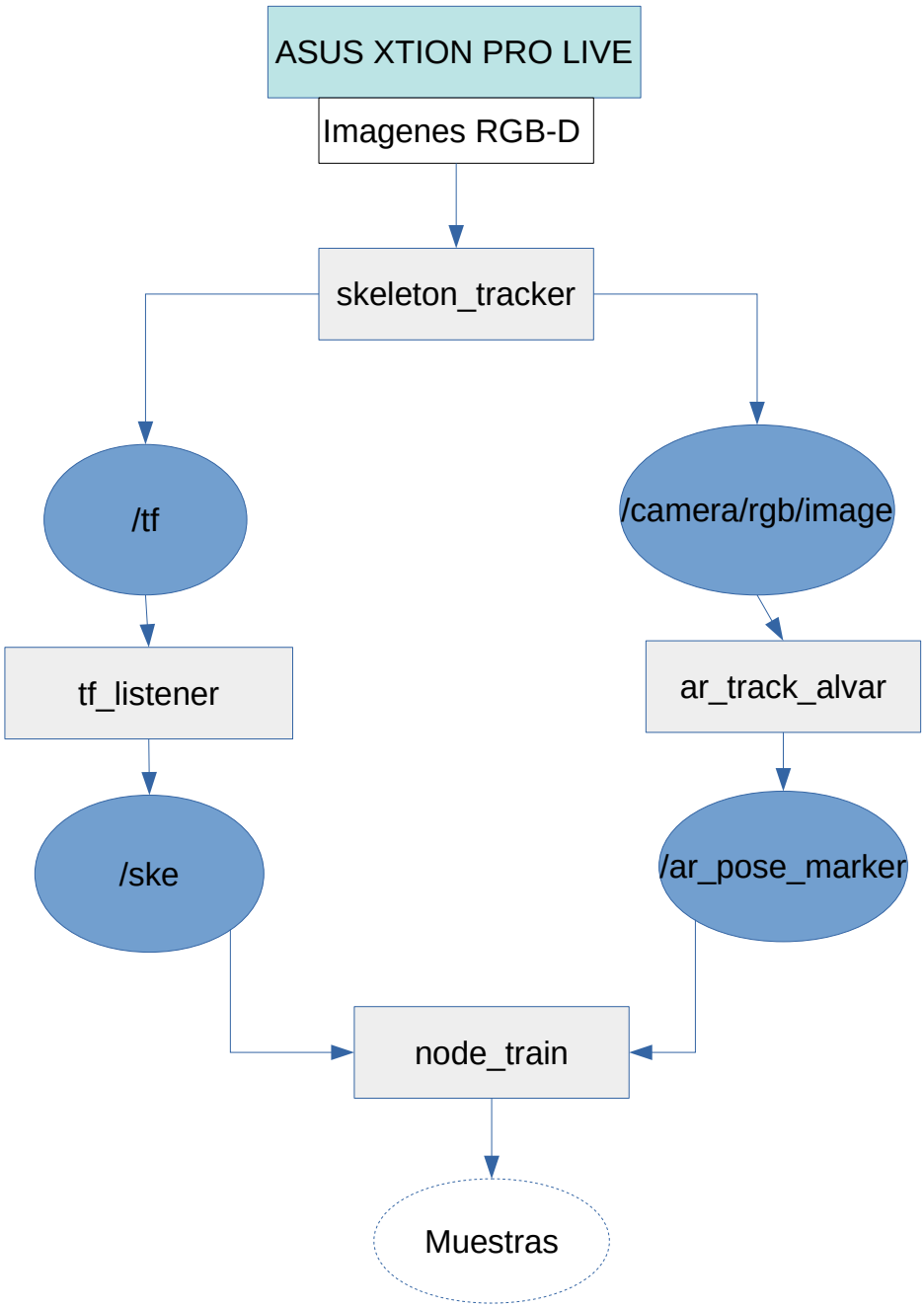


Fig. 3.3. Diagrama conceptual del método para recoger los datos de entrenamiento

3.1.2. SVM

Nuestra librería para la utilización de la SVM, no solo es capaz de crear un modelo; sino que también tiene otras sub-herramientas para escalar los datos (*SVM_scale*) y predecir etiquetas (*SVM_predict*). En la figura 3.4 se observa cómo se va a utilizar esta aplicación, mas adelante se desarrolla su utilización.

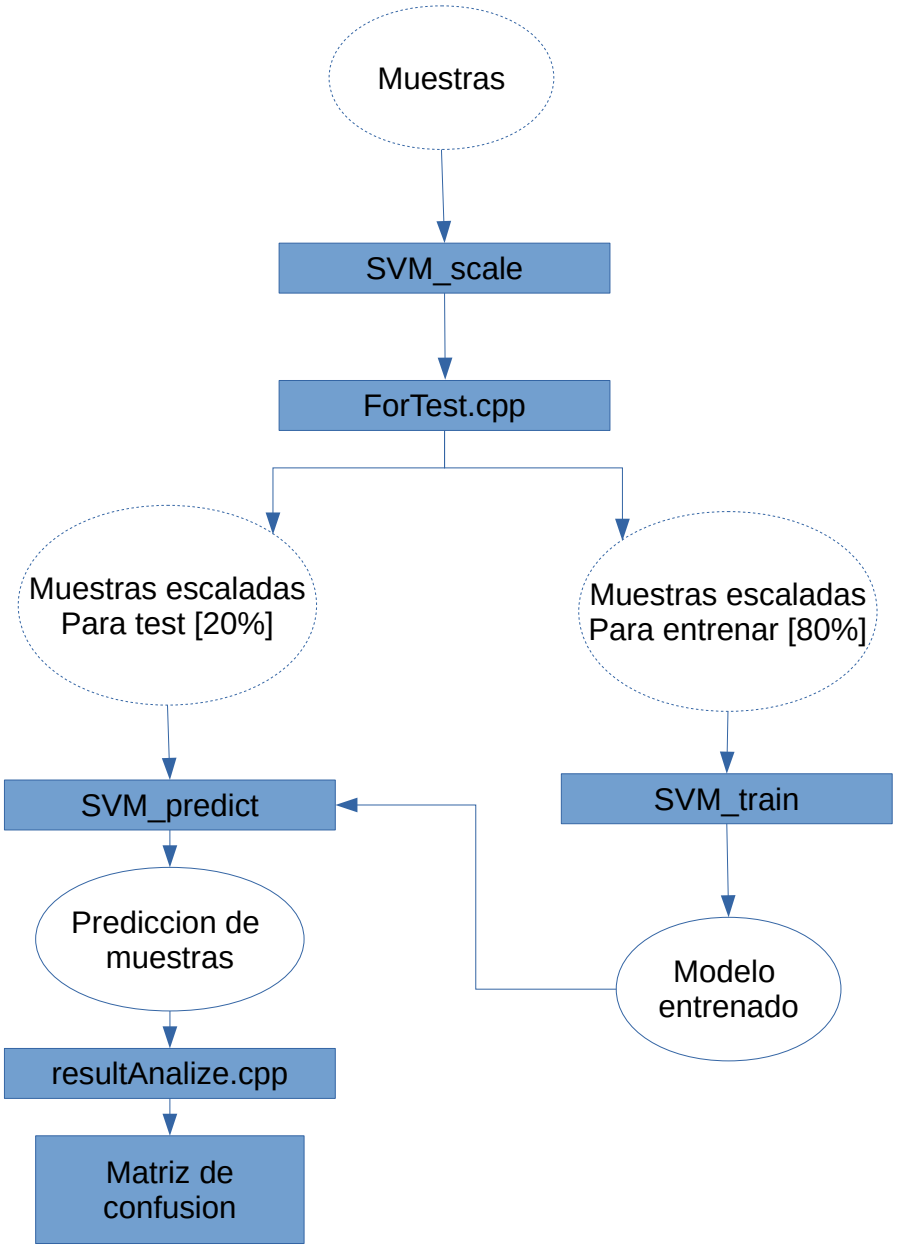


Fig. 3.4. Diagrama conceptual del funcionamiento de la SVM

3.2. Integración

Todo nuestra aplicación está desarrollada en ROS melodic y es por ello que es necesario que cualquier sistema computacional que quiera utilizarla deba tener acceso a un sistema operativo Linux, puesto que ROS melodic solo puede descargarse y funcionar correctamente sobre estas plataformas. Además se recomienda instalar una versión que no sea anterior a *Indigo* y que funcione con un sistema de compilación catkin, puesto que si no se pueden encontrar problemas de migración.

Aún con las facilidades que permite el framework elegido en el que trabajamos, ROS, la tarea de integración de todas las herramientas utilizadas en este trabajo ha sido el reto al que más tiempo se ha dedicado a lo largo su realización.

ROS reduce mucha carga de trabajo pues permite conectar librerías y módulos, de tal manera que el programador no tiene que preocuparse de escribir código de interconectividad a bajo nivel de los mismos, mas allá de los parámetros que ROS sugiere establecer en todos los nodos implementados. Además permite reutilizar aplicaciones y asi poder avanzar de manera más fácil en la investigación y desarrollo de la robótica.

Ahora bien, ROS utiliza un numero elevado de parámetros de configuración para la compilación y ejecución, tanto en sus librerías como en sus paquetes. Esto dará entonces diferentes problemas a la hora de compilar el sistema una vez instalado, es por ello que aquí se dan unas pequeñas recomendaciones para saber donde buscar cuando el usuario se encuentre con esos problemas:

- Es importante que se tenga en cuenta que la mayoría de *paths* o rutas que están configurados en todas las CmakeLists de los diferentes paquetes de ROS están diseñadas para encontrar aplicaciones o librerías que cada usuario puede tener en diferentes localizaciones. Es por ello que si su sistema catkin no encuentra cierta librería o paquete simplemente tenga que cambiar la ruta a donde el usuario tenga instalada dicha aplicación, dentro del archivo CMakeList.

- También es posible que cierto paquete no funcione debido a que ROS es incapaz de encontrar una aplicación o librería necesaria. Para evitar esto asegúrese de instalar todas las librerías y paquetes requeridos en la sección *find_package* del CmakeLists del paquete que de problema.

- Es muy recomendable que si va a usar y modificar estos archivos de configuración, tenga una idea general de qué esta modificando, puesto que después seria mucho mas difícil encontrar el fallo.

4. DESARROLLO

En esta sección del proyecto se va a describir detenidamente todas la partes de las que consta el proyecto. Se explica en profundidad cómo funciona el sistema y qué herramientas han sido necesarias para la implementación del mismo.

4.1. Ingeniería de requerimientos de hardware

Para la captura de imágenes de nuestro proyecto hay que escoger una cámara que dé la posibilidad de recoger tanto imágenes RGB, para poder utilizar skeleton_tracker, como imágenes Depth, para poder utilizar ar_track_alvar. Tras comprobar experimentalmente que el software no puede funcionar con la Asus Xtion Pro, se decide escoger la Asus Xtion Pro Live vista en la figura 4.1.

Fig. 4.1. Asus Xtion Pro Live

Asus Xtion Pro Live es la primera solución sensible al movimiento y los colores (RGB) para el desarrollo de software interactivo, su SDK compatible con OPENI y NITE facilita su uso en el desarrollo de aplicaciones. Emplea un sensor de infrarrojos, la detección adaptativa de la profundidad y el color para capturar los movimientos del usuario en

tiempo real.

4.2. Ingeniería de requerimientos de software

La carga principal del trabajo consiste en el desarrollo de los diferentes módulos de software que permitan cumplir el objetivo. El lenguaje que se utiliza para la implementación de las diferentes herramientas es C++, un lenguaje muy potente y robusto que permite integrar sin problema nuestros sistemas dentro de aplicaciones mucho más complejas.

4.2.1. ROS

ROS es un framework flexible modulado ideal para implementar aplicaciones de software en sistemas robóticos. Es una colección de herramientas y librerías que tienen como misión simplificar la creación de módulos implementados para el control del comportamiento de sistemas robóticos en diferentes plataformas. El objetivo principal de la utilización científica de este software es apoyar la reutilización de código en la investigación y desarrollo de robótica para que se pueda encontrar una multitud de paquetes diferentes en un sistema integrado. De manera muy resumida los sistemas implementados en este software funcionan basándose en tres conceptos:

- Nodos: Son programas ejecutables con un propósito único. Trabajan de manera individual, es decir, cada uno de ellos es compilado, ejecutado y manejado de manera individual. Todo nodo puede suscribirse a diferentes topics y/o publicarlos, y así crear una red de nodos que realizan una determinada tarea de manera modular.

- Topics: Los topics o temas, son el medio que utilizan los nodos para comunicarse entre sí. Solo pueden manejar un tipo de mensaje, es decir, ya sea para publicar o para recibir un mensaje, un nodo debe manejar un tipo de dato concreto para poder comunicarse.

- Mensajes: Los mensajes son la estructura de dato que definen a los topics. Pueden ser de cualquier tipo de estructura de dato compuesto por float, integer, strings,

boolean y cualquier tipo de array de estas clases. Vendrán definidos en los archivos .msg.

- Servicios: Los servicios son transacciones instantáneas entre nodos que utilizan la estructura servicio/cliente con requerimientos de respuesta. Vendrán definidos en los archivos .srv.

El componente protagonista de este sistema es el nodo *master*, su función principal consiste en permitir a los nodos en ROS localizarse unos a otros y manejar su comunicación. Registra todos los nodos, servicios y topics que se encuentran en ejecución y permite su utilización. En la figura 4.2 se observa el funcionamiento general en la comunicación de cualquier nodo de ROS y el master.

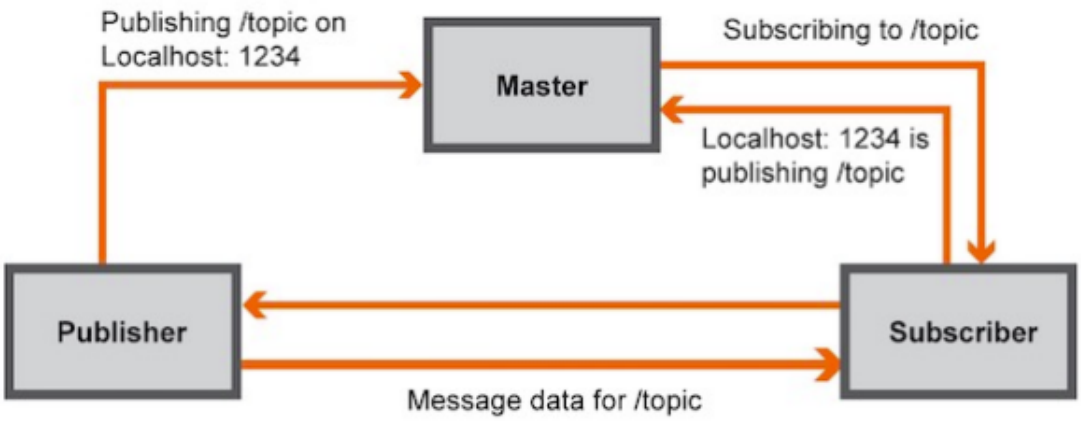


Fig. 4.2. Nodo master de ROS

4.2.2. Paquetes de ROS utilizados

Para la implementación de nuestro código se han requerido diferentes software de código libre y librerías implementadas por terceras personas dentro de este framework. Esencialmente estos módulos se encargaran de gestionar nuestra Asus y hacer posible la recogida y utilización de nuestras muestras. Se ha necesitado instalar muchas librerías diferentes a lo largo del camino, prácticamente todas eran requeridas por los diferentes nodos que describimos a continuación, y que son la elección escogida como herramienta para cumplir nuestros objetivos:

- perception_pcl: La libreria PCL *(Point Cloud Library)* es una librería codificada en C++ que permite el procesamiento de imágenes de 3D su propósito es acelerar el trabajo algorítmico 3D en percepción, para su uso en aplicaciones robóticas. Esta librería será utilizada principalmente por AR para poder detectar y ubicar las etiquetas en el espacio.

- openni2_camera: Openni2_camera es la herramienta principal que permite utilizar la Asus. Esta librería es necesaria para cualquier software que utilice openni2 que sea capaz de recoger la información necesaria de la cámara y publicarla en una multitud de topics que permiten recoger diferente información.

- ar_track_alvar: Para poder reconocer las *Affordances* de las acciones que vamos a categorizar, hay que fijarse en dos características principales; el movimiento de los objetos que vamos a utilizar en nuestras acciones y el movimiento del cuerpo del ser humano que las realiza. Para cumplir con el primero de nuestros dos objetivos vamos a utilizar ar_track_alvar. Este software funciona reconociendo en el espacio una etiqueta que ubicada en cualquier clase de objeto. El nodo de dicho software se suscribe al topic que se determine en su archivo .launch para recoger la imagen RGB ofrecida por skeleton_tracker y publica en un array de tipo *PoseStamped*, una estructura de dato característica de geometry_msgs, la ubicación y orientación de nuestra etiqueta. El topic en el que se recogen todos los datos publicados es denominado ar_pose_marker y tendrá la posición de nuestra etiqueta en el espacio respecto a la frame estática ubicada en el sensor de la cámara. Es importante mencionar la necesidad de cambiar el archivo .launch de este nodo para que pueda suscribirse al topic que tu deseas.

- skeleton_tracker: El segundo objetivo planteado para el reconocimiento de acciones es el de detectar la posición del cuerpo humano. Para lograrlo se ha utilizado skeleton_tracker. Este software funciona reconociendo la ubicación y orientación de 15 puntos característicos del cuerpo humano, de la cabeza a los pies, literalmente. Además es capaz de publicar todos los topics correspondientes a la información de la cámara como la imagen que emite. A diferencia de ar_track_alvar, cogerá la información requerida directamente de la cámara y publicará en un topic de tf un array con las 15 frames de los puntos del cuerpo humano respecto a la posición de

la imagen *Depth* de la cámara. Este array almacenará 15 estructuras de dato de tipo *TransformStamped* en un topic de tf, por lo tanto se debe crear un programa capaz de reconocer las frames y convertirlas en tipos de dato de geometry_msgs, si se quiere ser capaces de acceder a los datos y manipularlos.

4.2.3. Software propio

Nuestro objetivo principal del proyecto consiste en crear unas muestras fiables, en el formato que la SVM demande, para poder reconocer los *Affordances* de las acciones y categorizarlas automáticamente. Para lograrlo se han utilizado aplicaciones de software en ROS que han precisado de código propio que permitiese recoger los datos que se ofrecían, lo que ha supuesto la mayor carga intelectual del trabajo. Además a lo largo del trayecto se han presentado diferentes oportunidades para implementar ejecutables que ayudasen a manejar de una manera más eficiente los archivos de texto con nuestras muestras. Cómo resultado se puede hacer una enumeración del código que se ha necesitado generar:

- tf_listener: tf es una librería capaz de mantener la relación entre frames y permite al usuario manejar transformadas de puntos y vectores en cualquier momento temporal[35]. Fue diseñada cómo un paquete de ROS y está orientada al manejo de datos para la orientación en un espacio tridimensional de aplicaciones robóticas. tf almacena los datos de los puntos en el espacio que se requieran, pero lo hace individualmente y es por ello que necesitamos crear un *listener* de esta herramienta para poder referenciar las distancias y orientaciones entre dos puntos del espacio. Lo que conseguimos con este ejecutable es que, a través de la función *lookupTransform* que ofrece esta librería tf, buscamos la transformada entre dos frames que interesen y convertimos esos datos en una estructura de dato de geometry_msgs que se publica en el topic que se desea, en nuestro caso el topic /ske, que contendrá los datos del esqueleto humano recogido. Además la recogida de información del topic de tf estara implementada en un sistema try/catch, que se suele utilizar para sistemas que puedan fallar, esto se hace asi para que si skeleton_tracker no es capaz de publicar datos en un cierto instante, tf_listener sea capaz de no acceder a memoria inexistente y por lo tanto haya un error *Segmentatio Fault* en el sistema. En la figura 4.3 se puede observar el funcionamiento general de la función *lookupTransform*.

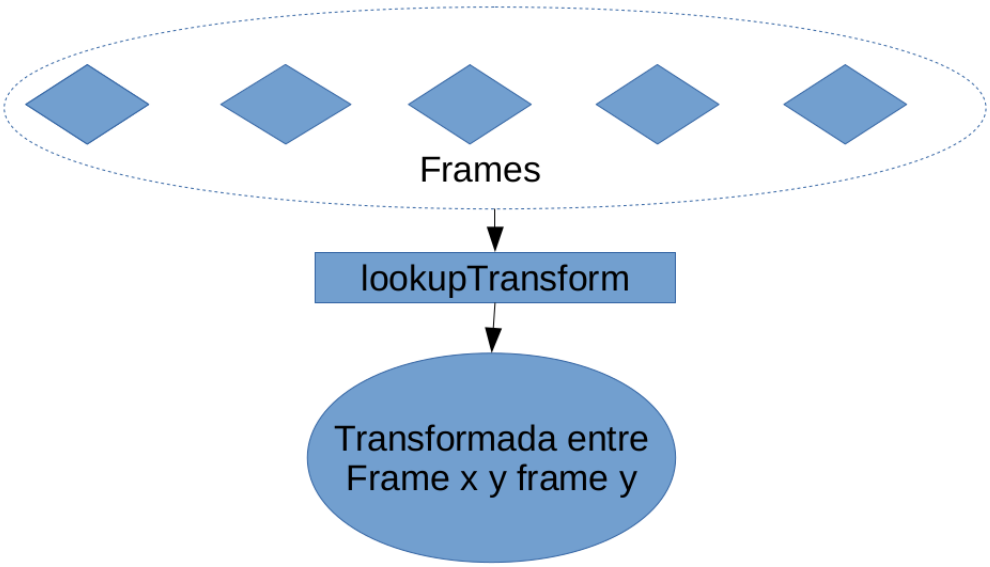


Fig. 4.3. Funcionamiento tf_listener

Para poder trasmitir de manera eficiente los datos del esqueleto humano se creó un tipo de mensaje .msg *ske* que contendría 9 datos tipo *TrasnformStamped* fáciles de leer en nuestro nodo principal.

- node_train: El objetivo principal de nuestro trabajo es crear unas muestras que podamos entrenar dentro de la máquina de soporte vectorial. Este nodo de ROS será el encargado de esta misión. Este nodo tendrá dos clases implementadas que funcionarán de manera similar, una definirá los objetos creados a partir de los datos recogidos por skeleton_tracker y la otra definirá los objetos creados a partir de los datos recogidos en ar_track_alvar. Cada una de estas clases será capaz de recoger estos datos como parámetros de los objetos, que leerán del topic requerido cuando el nodo master de ROS les indique que hay un nuevo mensaje en el topic al que se han suscrito. Contarán con un parámetro temporal adicional implementado para determinar el tiempo exacto en el que el sistema ejecuta la función *Callback* presente en cada uno. Dentro de la función *Callback* del objeto que se encarga de atender el nodo creado por ar_track_alvar, se ha implementado un sistema para saber si el array publicado esta vacío y no acceder a memoria inexistente, provocando un error de *Segmentatio Fault* que bloquearía el sistema. Además contarán con una función capaz de convertir los datos muestreados a parámetros en strings. Luego dentro del main, que es donde se ejecutaran las suscripciones a los topics y el bucle principal de atención a mensajes, se ejecutará un pequeño menú con el que elegimos que

acción vamos a muestrear, en el bucle de atención a los topics simplemente se comprueba que los mensajes llegados de las dos herramientas son simultáneos, y si es así, se juntan dichas muestras y se escriben como dato en nuestro fichero. Además se utiliza la etiqueta de ar_track_alvar como método de inicio de toma de muestras, es decir, se incluye un flag en el *Callback* que permite saber si está percibiendo la etiqueta y solo se escribe en los ficheros cuando esta etiqueta sea detectada; así se tendrá pleno control de cuando se comienzan a tomar las muestras y cuando se detienen.

- forTest: Este ejecutable simplemente ayuda a separar las muestras recogidas por node_train. Leerá el archivo de texto linea a linea y aleatoriamente alojará las muestras en el archivo .t o .train de la accion reconocida.

- resultAnalaizer: Este ejecutable simplemente leerá el archivo .predict generado por la SVM y clasifica los resultados para ayudarnos a crear la matriz de confusión, que permite no solo saber cuantas veces a categorizado bien nuestro sistema, sino también, que acciones confunde con que otras acciones .

Juntando las diferentes herramientas anteriormente descritas se forma un sistema que es capaz de recoger las muestras y posteriormente predecir su identidad. A continuación se muestra de manera esquemática el funcionamiento del sistema de ROS completo para la recogida de muestras en la figura 4.4. Luego de este sistema de comunicación la SVM, en el proceso que se ha comentado anteriormente, ayuda a identificar nuestras acciones.

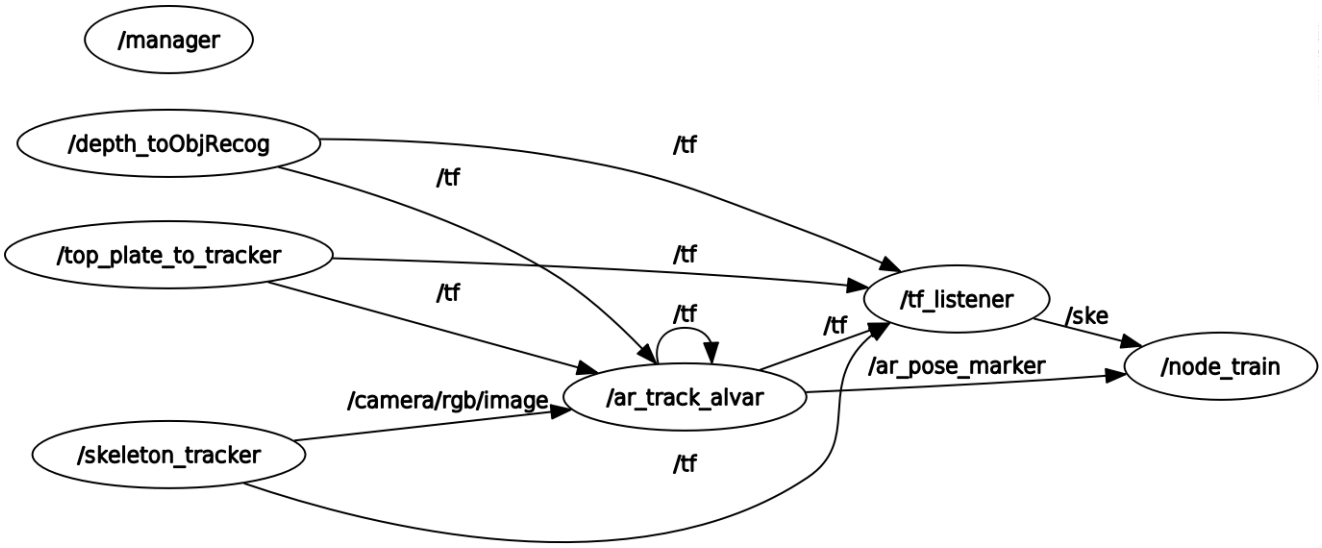


Fig. 4.4. Comunicaciones en ROS

Cada muestra tomada viene definida por 70 datos numéricos diferentes. Estos datos tienen dos claras partes diferenciadas, los primeros 63 números representan los datos tomados de los puntos en movimiento del cuerpo humano y los últimos 7 representan los datos de la ubicación en el espacio de la etiqueta identificada por ar_track_alvar. Estos primeros 63 números son característicos de 9 puntos diferentes del ser humano; la cabeza, el cuello, el torso, la mano izquierda y derecha, el codo izquierdo y derecho, y el hombro izquierdo y derecho. Cada punto viene definido por 7 datos numéricos; 3 puntos que localizan su posición en el el espacio y otros 4 que determinan su orientación. De esta misma manera los datos desde el numero 63 al numero 70 representan la posición y la orientacion del objeto. El formato de la muestra ha de ser cuidado para que pueda ser utilizado en nuestra SVM, este formato se observa en la figura 4.5.

```
 1:0.100664 2:-0.215157 3:-1.3442 4:-0 5:-0 6:-0 7:1 8:0.0702402
9:-0.452356 10:-1.33423 11:-0 12:-0 13:-0 14:1 15:0.097433
16:-0.0355161 17:-1.31453 18:-0 19:-0 20:-0 21:1 22:0.238844
23:-0.20869 24:-1.36831 25:-0 26:-0 27:-0 28:1 29:0.308242
30:0.0431876 31:-1.38476 32:-0 33:-0 34:-0 35:1 36:0.273308
37:0.201401 38:-1.08365 39:-0 40:-0 41:-0 42:1 43:-0.0375166
44:-0.221624 45:-1.3201 46:-0 47:-0 48:-0 49:1 50:-0.0827581
51:-0.0279666 52:-1.35294 53:-0 54:-0 55:-0 56:1 57:-0.148898
58:0.269385 59:-1.25174 60:-0 61:-0 62:-0 63:1 64:0.121984
65:0.0632511 66:0.566854 67:0.935665 68:-0.150343 69:0.175222
70:0.175222
```

Fig. 4.5. Ejemplo de muestra individual

4.2.4. SVM

Se ha escogido una máquina de soporte vectorial, debido a que sus ventajas dentro del aprendizaje automático parecían idóneas para nuestro proyecto. Hoy en día se consideran mucho más fáciles de utilizar que una red neuronal y su eficacia para implementaciones de esta dimensión es perfecta. Entre las diferentes herramientas que encontramos en código abierto en internet escogimos LIBSVM. LIBSVM es una de las herramientas más utilizadas en la última década como apoyo a proyectos que precisan de una máquina de soporte vectorial. En la tabla 4.1 se pueden ver algunas de las principales aplicaciones en las que se ha utilizado.

DOMINIO	**TRABAJOS REPRESENTATIVOS**
Visión artificial	LIBPMK [36]
Procesamiento de lenguaje	Maltparser [37]
Neuroimaging	PyMVPA [38]
Bioinformatica	BDVal [39]

TABLA 4.1. CAMPO CIENTIFICO LIBSVM

El funcionamiento interno de esta herramienta se puede ver en [34]. En este proyecto se va a describir de manera resumida cómo vamos a utilizar las diferentes posibilidades que LIBSVM ofrece. Es muy importante tener muy en cuenta el formato de dato que LIBSVM requiere y generar las muestras, con nuestros nodos de ROS, de tal manera que permita ejecutar sin problema la máquina de soporte vectorial. Las cuatro herramientas que vamos a utilizar son:

- svm-scale: Esta herramienta se utiliza para escalar los datos de nuestras muestras, una nota importante a tener en cuenta es que todos los datos se van a escalar con los mismos parámetros, tanto el conjunto de datos de entrenamiento como el conjunto de datos de test. Escalar los datos introducidos en una máquina de soporte vectorial es altamente recomendable, pues evita que los atributos numéricos más grandes dominen los rangos numéricos más pequeños. Además evitan las complejidades a la hora del cálculo , debido a que los valores del kernel usualmente dependen de los productos internos de los vectores de característica. En este proyecto se utilizan

etiquetas entre [-2,+2] para facilitar el entrenamiento de la máquina. El ejecutable genera dos tipos de salida, un archivo .scale con las muestras escaladas y un archivo .range con todos los parámetros utilizados en el proceso.

- svm-train: Este es el ejecutable que se encargará de entrenar las muestras creadas en nuestro nodo, una vez pasadas por svm-scale. Este programa necesita que las muestras ya hayan sido etiquetadas para poder crear el modelo. Este modelo será un determinado número de datos parametrizados que, a través de un determinado tipo de procedimiento *(kernel)*, generara una distribución matemática capaz de categorizar futuras muestras no entrenadas. La salida del programa sera un archivo .model con todos los parámetros utilizados y un conjunto de características que genera el sistema, dependiendo de las muestras introducidas.

- svm-predict: Finalmente svm-predict se encargará de analizar los datos no etiquetados ni entrenados y deducir una categorización de cada una de las muestras. En el caso de no ser capaz de decidir la etiqueta de una muestra el sistema la elimina de la lista de predicciones, esto es a lo que es llamado *accuracy*, sera el porcentaje de muestras que el sistema ha sido capaz de categorizar. El ejecutable tendrá como salida un archivo .predict con todas las etiquetas categorizadas con éxito. Un apunte que hay que hacer es que las etiquetas atribuidas no tienen por qué estar correctamente etiquetadas, de ahí se obtendrán los datos para nuestra matriz de confusión.

- grid.py: Este programa es un código generado en python que permite sacar valores más concretos antes de entrenar nuestras muestras y así facilitar y mejorar los resultados.

- easy.py: Esta herramienta permite ejecutar de una sola vez las herramientas anteriormente mencionadas, producirá todas las salidas mencionadas en las diferentes etapas, y además, generará un gráfico conceptual del modelo generado.

Una vez introducidas las herramientas a utilizar vamos a explicar cual es el procedimiento que vamos a utilizar para la separación y categorización de muestras.

1. En primer lugar se separan las muestras de cada archivo en una proporción 20-80. El 20 % será utilizado como datos de test y el otro 80 % como datos de entrenamiento.

2. En segundo lugar se van a escalar los datos con svm-scale, para así tener los datos de entrenamiento y test igualmente escalados. Los datos estarán etiquetados por nuestro nodo node_train, esta categorización será un etiquetado con valores en el rango [-2,+2] los más separadamente posible entre sí. Los datos de test también estarán etiquetados, esto se debe a que el sistema utilizará esa etiqueta solo para saber que comprobación se ha hecho bien o mal y así poder sacar un porcentaje automáticamente.

3. Luego se juntan los datos de los archivos de entrenamiento de cada acción en un solo archivo. Así se permite que svm-train pueda generar un modelo que sea capaz de categorizar las diferentes etiquetas utilizadas y tengamos por fin nuestro archivo .model.

4. Por último se utiliza svm-predict con cada uno los diferentes archivos de test .t de las diferentes acciones que se han separado al principio. Esta predicción se hace con el modelo generado previamente para cada uno de los archivos .t y así se es capaz de generar un archivo .predict para cada una de las acciones. Cómo nosotros sabemos que etiqueta debería producirse a la salida de cada uno de los archivos, se construirá la matriz de confusión, y no solo obtener la información de cuanto se equivoca o no nuestro sistema; sino el ratio con el que confunde cada una de las acciones con el resto.

5. EXPERIMENTOS

En esta sección vamos a describir las pruebas que se han llevado a cabo para saber la eficacia de la solución implementada, planteando diferentes escenarios. Nuestro objetivo principal es poder reconocer las *Affordances* de los objetos a través de nuestro método y poder identificar las acciones que realizamos en frente de la Kinect. Para poder probar nuestro software vamos a escoger acciones que involucren el movimiento de una persona y la utilización de algún objeto. Para así poder coger ambas muestras simultáneamente y predecir la acción que se esta realizando. Las acciones escogidas en cuestión serán dos diferentes que se realizarán con un vaso; beber y vaciar, y dos diferentes que se realizarán con un tenedor; comer y remover, además se elegirá otra acción individual que no tenga nada que ver con las anteriores, lanzar.

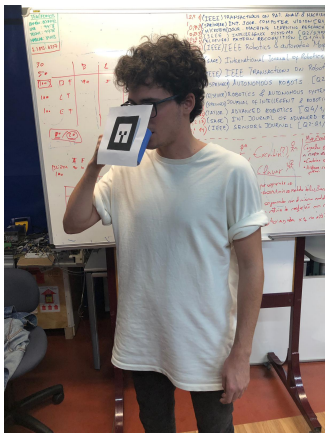
5.1. Procedimiento

Se va a colocar la Asus Xtion Pro Live a 1 metro del suelo y tras comprobar que es capaz de detectarnos a cuerpo entero, se va a dejar una marca a 1,2 metros de distancia del sensor habiendo comprobado que el software no tiene problemas en detectar la etiqueta del objeto y en localizar los puntos requeridos del cuerpo humano. Tras escoger la acción a muestrear se ejecutará en primer lugar skeleton_tracker y se esperará hasta el momento que la aplicación muestre Üser #1: Tracking!", en ese momento se iniciará ar_track_alvar con el launch file deseado modificado; para luego ejecutar a la vez los programas tf_listener y node_train. Entonces nuestro sistema comenzará a recoger los datos deseados de la acción que queramos grabar. Cada vez que se deseaba cambiar de acción se paraba completamente el sistema y se volvían a iniciar los 4 nodos otra vez.

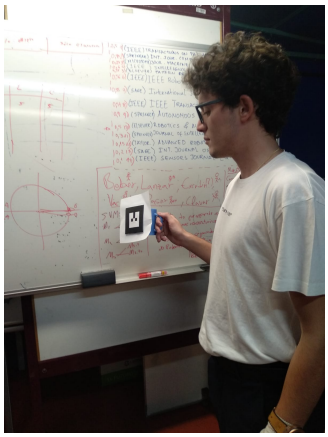
5.2. Pruebas y resultados

En esta sección se pueden observar las diferentes pruebas que se han llevado a cabo y sus resultados. Se han hecho 3 tipos de pruebas diferentes debido a las diferentes resultados que se han ido obteniendo. En la figura 5.1 se puede observar como se van a realizar

las acciones frente a la cámara.

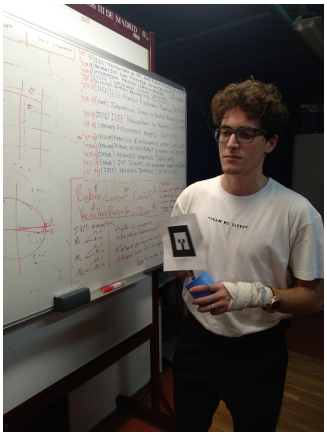


(a) Beber

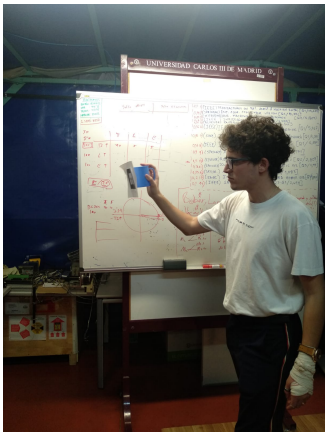


(b) Vaciar

(c) Comer



(d) Remover



(e) Lanzar

Fig. 5.1. Usuario realizando diferentes acciones

5.2.1. Set 1 de pruebas

En las primeras pruebas experimentales llevadas a cabo se trató de seguir el procedimiento descrito anteriormente. Para este primer experimento se tomaron 200 muestras de cada acción para entrenar la máquina de soporte vectorial, y 50 muestras de cada acción para testear el modelo generado por la máquina de soporte vectorial.

En estas condiciones encontramos que la máquina de soporte vectorial es capaz de reconocer todas las acciones en el 100 % de las ocasiones. Estos resultados se acogieron con escepticismo, cabía la posibilidad de que se estuviera contemplando un escenario demasiado idealizado. Esto dio lugar a diseñar más experimentos orientados a confirmar el éxito de los resultados, tratando de plantear retos mayores al sistema, dado que los

resultados obtenidos no son los esperados para un sistema que se ha probado por primera vez.

5.2.2. Set 2 de pruebas

Tras los resultados obtenidos en el procedimiento anterior decidimos que el problema podría ser producido porque al tomar las muestras de las diferentes acciones reiniciando el sistema, la colocación al realizar la siguiente acción era diferente, y nuestro sistema tomaba como *Affordance* de la acción realizada el posicionamiento con respecto a la cámara. Para evitar esto decidimos reiniciar únicamente el módulo node_train teniendo especial cuidado con realizar todas las acciones exactamente en la misma posición, tratando que las acciones se pareciesen lo máximo posible entre sí en cuanto a posicionamiento y lugar de realización. Los resultados fueron similares, encontramos un reconocimiento entre el 98-100 %.

Decidimos investigar con las muestras obtenidas y cambiar parámetros. Se observa que introduciendo menos muestras de entrenamiento por cada acción en la aplicación de la máquina de soporte vectorial que crea el modelo, se obtienen predicciones mas razonables en torno al 60-80 % dependiendo de la acción; y que además, según se sube el número de muestras de entrenamiento del modelo, las predicciones se acercan al 100 %. Se comprueba entonces que los resultados se ven altamente condicionados por la cantidad de muestras con las que se realiza el modelo de predicción. Nuestro objetivo no es comprobar el funcionamiento de la SVM para nuestras muestras y comprobar bajo que parámetros mejora su funcionamiento. Además se continúa viendo los resultados con escepticismo dado que los resultados de un 100 % no son los esperados.

5.2.3. Set 3 de pruebas

Se decidió que se iba a probar el sistema de una manera mas real y con una perspectiva frente al cuidado en realizar las acciones en el mismo punto mucho mas aleatoria, evitando así que el sistema encontrase *Affordances* falsos en nuestro posicionamiento frente a la cámara.

Es por ello que se tomaron las muestras con el usuario en constante movimiento, o in-

cluso cambiándose los objetos de mano y moviéndose de manera aleatoria por el entorno. En estas condiciones se probaron modelos creados con diferentes números de muestras y los resultados fueron muy parecidos. Estos resultados se acogieron con más credibilidad, dado que la SVM no podría categorizar las acciones por la ubicación respecto a la cámara pues era constantemente aleatoria. Además el cambio de manos de los objetos aportaba mas realismo a los resultados. Estos datos se obtuvieron introduciendo 100 muestras de cada acción para la creación del modelo y 50 muestras de cada acción para hacer el test. En la tabla 5.1 se pueden ver los resultados obtenidos en la matriz de confusión, las palabras en negrita representan las acciones simuladas y las no negrita las acciones reales. Por ejemplo la primera fila representa que para la acción real realizada beber, se ha identificado un 90.74 % de las muestras correctamente, un 3.72 % de las muestras se han confundido con la acción comer y un 5.54 % de las muestras se han confundido con la acción remover.

%	**Beber**	**Vaciar**	**Comer**	**Remover**	**Lanzar**
Beber	90.74	0	3.72	5,54	0
Vaciar	0	95.46	2.27	2.27	0
Comer	0	2.22	84.45	13.33	0
Remover	5.26	0	13.16	81.58	0
Lanzar	0	0	0	0	100

TABLA 5.1. MATRIZ DE CONFUSIÓN DE LOS RESULTADOS DE LA PRUEBA COLECTIVA

5.2.4. Set 4 de pruebas

En vista de los resultados anteriores se decidió realizar un cuarto set de pruebas probando el sistema con modelos creados a partir de aquellas acciones que más confundieran a la SVM. Observando que la acción de lanzar era siempre bien identificada por el sistema, debido a ser la más diferente de las acciones, se decidió separar para estos experimentos. Los experimentos se realizaron con aquellas acciones que se confundieran en más de un 5 % de las ocasiones, es decir beber-remover y remover-comer. En la tabla 5.2 y 5.3 podemos observar los resultados.

%	**Beber**	**Remover**
Beber	96	4
Remover	4	96

TABLA 5.2. MATRIZ DE CONFUSIÓN DE LOS RESULTADOS PARA BEBER Y REMOVER

%	**Comer**	**Remover**
Comer	82	18
Remover	8	92

TABLA 5.3. MATRIZ DE CONFUSIÓN DE LOS RESULTADOS PARA COMER Y REMOVER

5.2.5. Set 5 de pruebas

Se han realizado pruebas para comprobar como se reconocerían estas acciones con las datos de la localización del cuerpo humano y de la localizacion del objeto por separado. Lo único que se ha hecho ha sido separar las muestras; en los datos recogidos para el cuerpo humano (los primeros 63 datos de cada muestra) y en los datos recogidos para la lozalización del objeto (los ultimos 7 datos de cada muestra). Se ha utilizado la SVM de la misma manera, y con las 5 acciones a la vez. En las tablas 5.4 y 5.5 se pueden ver los resultados obtenidos.

%	**Beber**	**Vaciar**	**Comer**	**Remover**	**Lanzar**
Beber	83.33	0	8.33	8.33	0
Vaciar	0	100	0	0	0
Comer	16.67	0	66.67	16.66	0
Remover	0	0	0	76.47	23.53
Lanzar	14.3	0	0	0	85.7

TABLA 5.4. MATRIZ DE CONFUSIÓN DE LOS RESULTADOS DE LA PRUEBA CON RECONOCIMIENTO DE OBJETOS

%	**Beber**	**Vaciar**	**Comer**	**Remover**	**Lanzar**
Beber	41.67	33.33	16.67	8.33	0
Vaciar	0	83.34	8.33	8.33	0
Comer	8.33	0	83.34	8.33	0
Remover	0	15.38	15.38	69.24	0
Lanzar	0	0	0	0	100

TABLA 5.5. MATRIZ DE CONFUSIÓN DE LOS RESULTADOS DE LA PRUEBA CON RECONOCIMIENTO CORPORAL

6. CONCLUSIONES

Después de la implementación y desarrollo de la solución propuesta podemos concluir que se han cumplido los diferentes objetivos del trabajo. Se ha conseguido encontrar una herramienta que se ha ajustado de manera notoria a nuestras necesidades, se ha encontrado el software y hardware de apoyo necesario para la realización del proyecto, y se ha logrado integrar con éxito todas estas herramientas; además al haber realizado con éxito la aplicación en ROS, se ha conseguido crear un sistema fácilmente integrable.

Nuestro objetivo principal, el reconocimiento de acciones de la vida cotidiana a través de las *Affordances* intrínsecas en las mismas, ha sido relativamente cumplido puesto que hemos logrado identificar esas acciones, con un mínimo del 80 % para la acción menos identificable. Además hemos comprobado como se relacionan los resultados a la hora de enfrentar directamente las acciones más realizadas. Por último, se han presentado resultados experimentales que prueban la mejoría del reconocimiento de estas *Affordance* al juntar los datos recogidos de la ubicación espacial del objeto y de la ubicación de los puntos del ser humano, a excepción de la acción de vaciar donde observamos un reconocimiento del 100 % con los datos del objeto. Esto era de esperar puesto que la etiqueta del objeto en esta acción, es en la única en la que está dada la vuelta, facilitando enormemente al sistema la identificación de la acción.

6.1. Trabajos futuros

A lo largo de la implementación del código, la integración de las herramientas utilizadas o en la sección de experimentos nos hemos encontrado con problemas que nos han hecho desviarnos o descartar diferentes objetivos que nos habíamos plantado en un primer momento. Mi intención es desarrollarlas en un futuro y por eso dejo de su constancia en esta sección del trabajo:

1. SVM: La librería SVM que utilizamos es perfecta para aplicaciones offline como la que se ha realizado, pero deja poca maniobrabilidad si se quiere usar como parte de nuestro código fuente en vez de como aplicación. Es por ello que es posible mejorar

el sistema encontrando una librería SVM que puedas implementar como código en tus ejecutables y así tener un completo control en tiempo real del entrenamiento de los datos y de la predicción, además esto permitiría entrenar y predecir en nuestro propio ejecutable sin tener que abrir otros procesos para poder ejecutar diferentes aplicaciones; cómo es necesario con la librería escogida. En nuestro caso habíamos pensado implementar otro nodo llamado node_predict que ejecutase las acciones con *libsvm* precisadas, pero nos dimos cuenta de que esta situación no era eficiente y nos quedamos sin tiempo para indagar en cómo hacerlo.

2. Mejorar reconocimiento de Affordances: Una idea que tengo en la cabeza que habría mejorado sin duda alguna el reconocimiento de las *Affordances* es implementar otra característica que cuantificase el movimiento temporal de los puntos más importantes de cada acción y así se lograría tener en cuenta no solo un conjunto de posiciones individuales sino también su cambio en el tiempo. tf es capaz de hacer esto, pues te permite saber que diferencia hay entre la misma transformada en dos puntos diferentes de tiempo. Por ejemplo se puede crear una variable con la suma de las diferencias de la frame del objeto principal con el que se realiza la acción cada X tiempo.

3. Más números de acciones: Nuestro software puede ser utilizado para tratar de encontrar las *Affordances* de muchas acciones diferentes, pero seria óptimo lograr implementar un código que integrase mejores herramientas de captación; como softwares de segmentación y reconocimiento de objetos, no únicamente de localización.

4. node_predict: Un objetivo primordial que no se ha podido terminar realizando para este proyecto es el reconocimiento de *Affordances* on-line. Para ello es necesario implementar otro nodo en ROS que utilice la SVM ya entrenada para categorizar las acciones al mismo tiempo que se detectan, el cuerpo de este nodo sería muy parecido al del nodo node_train implementado en este proyecto, pues recogería las muestras de la misma manera, pero utilizaría el modelo de la SVM entrenada para reconocer on-line la acción que se está realizando. Para esto es importante lograr el apartado uno de esta sección.

6.2. Marco regulador

En este apartado vamos a estudiar el marco regulador en el que se ha realizado el proyecto, atendiendo tanto a a la legislación requerida como a la propiedad intelectual del sistema.

6.2.1. Legislación

Respecto al proyecto elaborado a continuación existen diferentes cuestiones éticas y legislativas que habría que tener en cuenta para el desarrollo futuro de aplicaciones en este campo.

Principalmente existe legislación propuesta para la implementación de la robótica en la sociedad y los riesgos que ello supone en la integridad y dignidad personal. El pasado 12 de Febrero de 2019 se aprobó en el parlamento europeo el siguiente texto que habla con profundidad sobre una política industrial global europea en materia de inteligencia artificial y robótica [40].

También se puede destacar la existencia de legislación aprobada que regula sobre la grabación pública de seres humanos y la utilización de dicho material; método por el cual nuestros robots sociales serian capaces de percibir y reconocer las acciones que se están realizando. Estas cuestiones están reguladas en el BOE, recogidas en Leyes Organicas disponibles on-line; se pueden ver los documentos reguladores de la protección de datos personales y garantía de los derechos digitales [41].

Para la implementación de este proyecto, se ha utilizado herramientas informáticas sujetas a sus correspondientes licencias:

a) Ubuntu como sistema operativo que esta sujeto a su licencia GNU GPL, que dá a los usuarios libertad de usar, estudiar, compartir y modificar dicho software.

b) ROS como framework de la aplicación, limitado bajo licencia BSD, que permite libertad al usuario para uso comercial e investigación de la aplicación.

c) skel_tracker es usada como aplicación de la detección de seres humanos y esta regulada por una licencia BSD, permitiendo libertad de uso comercial e investigación.

d) ar_track_alvar es usada como aplicación para la detección de objetos, esta regulada por una licencia GNU Lesser General Public License, que garantiza la libertad de compartir y modificar el software cubierto por ella.

En este proyecto se cumplirán los requisitos del Reglamento de organización y funcionamiento del Comité de Ética en Investigación de la Universidad Carlos III de Madrid en donde ver las recomendaciones y compromisos para la realización de las actividades de investigación.

6.2.2. Propiedad intelectual

El objetivo de este trabajo es la investigación teórica del reconocimiento de *Affordances* para la navegación semántica, por lo tanto no estará sujeto a ningún tipo de pantentabilidad pues sera desarrollado como un trabajo teórico, aplicable tras su completa implementación, a futuros desarrollos de software.

6.3. Entorno socio-económico

Se va a estudiar de manera breve el impacto que este trabajo puede tener en la sociedad actual.

6.3.1. Presupuesto

El coste aproximado de este proyecto se calculará en las siguientes tablas. Las cuantías económicas se han valorizado a través de estimaciones basadas en los salarios del BOE oficial [42] y se dividen en: directo y los costes indirectos.

PUESTO	**TIEMPO**	**SALARIO POR HORA**	**TOTAL**
Graduado	500	15.75	7875
Profesor	60	35	2100
Total €			9975

TABLA 6.1. COSTES LABORALES

HERRAMIENTA	CONCEPTO	TOTAL
Acceso bibliografía	Coste de acceso	1100
Total €		1100

TABLA 6.2. COSTES INFORMÁTICOS

CONCEPTO	TOTAL
Costes laborales	9975
Costes informáticos	1100
Costes indirectos (40 %)	4430
Impuestos (21 %)	3256.05
Total €	18761.05

TABLA 6.3. COSTES TOTALES

6.3.2. Impacto social

El uso del reconocimiento de *Affordances* para la navegación semántica tiene numerosas aplicaciones en el campo de la robótica; concretamente tiene una gran utilidad para la robótica social. En la robótica social la aplicación centra su atención en la interacción con el individuo, es decir, los recursos utilizados se enfatizan en permitir que los robots se integren con la sociedad humana. Esta interacción se dá con el objetivo de contribuir con el ser humano en diversos aspectos de su vida. Siguiendo este camino nuestra aplicación podría ayudar a permitir al robot poder reconocer acciones que el ser humano realiza para poder coexistir con él, siendo una herramienta especialmente valiosa para los robots asistenciales y los utilizados en el sector servicios.

Esta clase de robots impactaría de manera importante en la sociedad porque siendo plenamente operativos, podrían realizar tareas de las que ahora se encarga el ser humano; como ser camarero/a, cocinero/a, dependiente o azafato/a. Económicamente podría suponer un cambio en estas posiciones laborales, pero crearía otros puestos de trabajo encargados de su supervisión y mantenimiento.

BIBLIOGRAFÍA

[1] K. Pelphrey y J. Morris, 'Brain Mechanisms for Interpreting the Actions of Others From Biological-Motion Cues', *Current directions in psychological science*, vol. 15, pp. 136-140, jul. de 2006. DOI: 10.1111/j.0963-7214.2006.00423.x.

[2] T. Lan, T.-C. Chen y S. Savarese, 'A Hierarchical Representation for Future Action Prediction', en *ECCV*, 2014.

[3] C. Vondrick, H. Pirsiavash y A. Torralba, 'Anticipating Visual Representations from Unlabeled Video', *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 98-106, 2015.

[4] U. Orozco-Rosas, K. Picos, V. H. Díaz-Ramírez, O. Montiel y R. Sepulveda, 'Visual environment recognition for robot path planning using template matched filters', en *Optical Engineering + Applications*, 2017.

[5] M. D. et al., 'Spacetime Pose Representation for 3D Human Action Recognition', *ICIAP Workshop on Social Behaviour Analysis,*, 2013.

[6] *«ROS: Robot Operation System». En linea*, Available: https://www.ros.org. Ultimo acceso: sept 2019.

[7] J. J. Gibson, 'The theory of Affordances', *The Ecological Approach to Visual Perception, chapter 8*, 1997.

[8] M. Cakmak y M. Doùgar, 'Affordances as a Framework for Robot Control', ene. de 2007.

[9] W. Gaver, 'Technology Affordances', ene. de 1991, pp. 79-84. DOI: 10.1145/108844.108856.

[10] T.-H. Vu, C. Olsson, I. Laptev, A. Oliva y J. Sivic, 'Predicting Actions from Static Scenes', en *ECCV*, 2014.

[11] H. S. Koppula y A. Saxena, 'Anticipating Human Activities Using Object Affordances for Reactive Robotic Response', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 14-29, 2016.

[12] D. K. et al., 'Object-Based Affordances Detection with Convolutional NeuralNetworks and Dense Conditional Random Fields', *IEEE-RSJ International Conference on Intelligent Robots and Systems*, 2017.

[13] H. Harzallah, F. Jurie y C. Schmid, 'Combining efficient object localization and image classification', *2009 IEEE 12th International Conference on Computer Vision*, pp. 237-244, 2009.

[14] B. Fulkerson, A. Vedaldi y S. Soatto, 'Class Segmentation and Object Localization with Superpixel Neighborhoods', nov. de 2009, pp. 670-677. DOI: 10.1109/ICCV.2009.5459175.

[15] *«ar track alvar». En linea*, Available: http://wiki.ros.org/ar_track_alvar. Ultimo acceso: sept 2019.

[16] S. Bag, 'Deep Learning Localization for Self-driving Cars', *Rochester: Rochester Institute of Technology*, 2017.

[17] K. He, X. Zhang, S. Ren y J. Sun, 'Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification', *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026-1034, 2015.

[18] S. Barbon, A. P. Barbon, N. Valous y D. Barbin, 'Image Segmentation', en. feb. de 2019, pp. 35-43. DOI: 10.1201/9781315209203-3.

[19] S. R. Lakani, A. J. Rodríguez-Sánchez y J. H. Piater, 'Can Affordances Guide Object Decomposition into Semantically Meaningful Parts?', *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 82-90, 2017.

[20] Z. Zhang, 'Microsoft Kinect Sensor and Its Effect', *IEEE MultiMedia*, vol. 19, pp. 4-10, 2012.

[21] *«openni2 tracker». En linea*, Available: https://github.com/ros-drivers/openni2_tracker. Ultimo acceso: sept 2019.

[22] Y. Yang y D. Ramanan, 'Articulated Human Detection with Flexible Mixtures of Parts', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2878-2890, 2013.

[23] A. Toshev y C. Szegedy, 'DeepPose: Human Pose Estimation via Deep Neural Networks', *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653-1660, 2014.

[24] S. B.-D. Shai Shalev-Shwartz, *UNDERSTANDING MACHINE LEARNING From Theory to Algorithms*, 1.ª ed. 32 Avenue of the Americas, New York: Cambridge University Press, 2014.

[25] S. K. Sarvepalli, *Deep Learning in Neural Networks: The science behind an Artificial Brain*, oct. de 2015. DOI: 10.13140/RG.2.2.22512.71682.

[26] S. Gite y H. Agrawal, 'Early Prediction of Driver's Action Using Deep Neural Networks', *IJIRR*, vol. 9, pp. 11-27, 2019.

[27] A. A. y Gorka Azkune, 'Predicting Human Behaviour with Recurrent Neural Networks', *DeustoTech-Deusto Foundation, University of Deusto*, 2017.

[28] I. Fatima, M. Fahim, Y.-K. Lee y S. Lee, 'A Unified Framework for Activity Recognition-Based Behavior Analysis and Action Prediction in Smart Homes', en *Sensors*, 2013.

[29] P. J. Phillips, 'Support Vector Machines Applied to Face Recognition', *Advances Neural Inform. Processing Systems, vol. 48*, 2001.

[30] E. Byvatov y G. Schneider, 'Support vector machine applications in bioinformatics.', *Applied bioinformatics*, vol. 2 2, pp. 67-77, 2003.

[31] N. Oliver, A. Garg y E. Horvitz, 'Layered representations for learning and inferring office activity from multiple sensory channels', *Computer Vision and Image Understanding*, vol. 96, pp. 163-180, 2004.

[32] P. Rashidi y D. J. Cook, 'COM: A method for mining and monitoring human activity patterns in home-based health monitoring systems', *ACM TIST*, vol. 4, 64:1-64:20, 2013.

[33] M. S. Pang-Ning Tan y V. Kumar, *Introduction to Data Mining*, 1.ª ed. Pearson Addison-Wesley, 2005.

[34] C.-C. Chang y C.-J. Lin, 'LIBSVM: A library for support vector machines', *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 27:1-27:27, 3 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[35] T. Foote, ‘tf: The transform library’, en *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ép. Open-Source Software workshop, abr. de 2013, pp. 1-6. DOI: 10.1109/TePRA.2013.6556373.

[36] K. Grauman y T. D. T, ‘The pyramid match kernel: Discriminative classification-with sets of image features’, *InProceedings of IEEE International Conference on-Computer Vision*, 2005.

[37] J. N. et al., ‘MaltParser: A language-independent system for data-driven dependencyparsing’, *Natural Language Engineering*, 7(1):37-53, 2009.

[38] M. Hanke et al., ‘PyMVPA: a Python Toolbox for Multivariate Pattern Analysis of fMRI Data’, *Neuroinformatics*, vol. 7, pp. 37-53, 2008.

[39] F. C. et al., ‘BDVal: repro-ducible large-scale predictive model development and validation in high-throughputdatasets’, *Bioinformatics*, 26(19):2472-2473, 2010.

[40] *Una política industrial global europea en materia de inteligencia artificial y robótica*, Available: http://www.europarl.europa.eu/doceo/document/TA-8-2019-0081_ES.html#def_1_4. Ultimo acceso: sept 2019.

[41] *Protección de Datos Personales y garantía de los derechos digitales*, Available: https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673&p=20190625&tn=2. Ultimo acceso: sept 2019.

[42] BOE, "BOE n.45,"2018.