Universidad Zaragoza
1542

Trabajo Fin de Grado

Grado en Ingeniería Electrónica y Automática

Reconocimiento de alimentos y manipulación con redes neuronales

Autor/es

Alicia Tierz Latasa

Director/es

Carlos Sagüés Blázquiz

Escuela de Ingeniería y Arquitectura

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Alicia Tierz Latasa,

con nº de DNI 17768381F en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado en Ingeniería Electrónica y Automática, (Título del Trabajo) Reconocimiento de alimentos y manipulación con redes neuronales

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 10 de Septiembre de 2018

Fdo: Alicia Tierz Latasa

Reconocimiento de alimentos y manipulación con redes neuronales

RESUMEN

Los asistentes virtuales son una herramienta cada vez más utilizada en el día a día de las personas. Los hay de muchos tipos y tienen campos de aplicación muy diferentes, pero en el ámbito de la cocina no son muy comunes. Este tipo de tecnología tiene un gran futuro, aunque aún hay muchos aspectos por desarrollar.
Sería deseable que una acción tan rutinaria como cocinar fuera fácil para todo el mundo, minimizando así el consumo de comidas ya preparadas y fomentando la comida sana y saludable.

En este trabajo de fin de grado se ha implementado un asistente de cocina que es capaz de, mostrar por pantalla los ingredientes necesarios o que faltan en una determinada receta y cuáles son los pasos que seguir para poder llevarla a cabo.
El asistente detecta los alimentos y utensilios de cocina que aparecen sobre la encimera o mesa en la que se está cocinando y las acciones que el usuario o usuarios están realizando a partir de imágenes que va captando por medio de la *Kinect.*

Para ello, utilizando imágenes con información de color, profundidad y seguimiento de personas, se hace un proceso previo segmentando los objetos a observar y se clasifican mediante una red neuronal entrenada con categorías referentes a la cocina.
Una vez clasificados los alimentos y utensilios se les aplica un filtro de Kalman para quitar el ruido que aparece, sobre todo en los objetos en movimiento y se detecta que acción de las siete que se han programado puede estar realizando el usuario y muestra por pantalla cuantos pasos han sido realizados y cual es el siguiente a llevar a cabo.

También es capaz de preguntar por pantalla, si un paso ha sido realizado para recordárselo al usuario o para apuntarlo como completado, si el asistente no ha sido capaz de reconocerlo.

Este sistema fue programado en un principio para un solo cocinero, aunque más tarde se ha extendido hasta dos personas.

Contenido

1. Introducción 5

1.1. Motivación 5

1.2. Estado de la materia y trabajo previo 5

1.3. Herramientas de trabajo 7

1.4. Estructura de la memoria 7

2. Red Neuronal utilizada 9

2.1. Selección de la red pre-entrenada 9

2.2. Entrenamiento de la red 13

2.3. Clasificación 17

3. Seguimiento de objetos 20

3.1 Filtro de Kalman aplicado al seguimiento de objetos 20

4. Seguimiento de personas 22

4.1. Seguimiento de una persona 22

4.2. Seguimiento de dos personas 25

5. Elaboración supervisada de alimentos 27

5.1 Creación de la receta 27

5.2 Seguimiento de la receta 27

6. Pruebas y experimentos 29

7. Conclusiones y líneas futuras de trabajo 32

Bibliografía 33

Tabla de ilustraciones 34

1. Introducción

Hoy en día, el uso de asistentes virtuales se ha incrementado sustancialmente, no es extraño oír como un móvil le dice a su propietario el tiempo que va a hacer mañana o como un conductor aumenta el volumen de la canción solo con un gesto de la mano sin tener que apartar los ojos de la carretera [1].

La interacción con máquinas que trabajan con inteligencia artificial es cada vez más frecuente ya que facilita muchas acciones de la vida cotidiana.

Uno de los campos que todavía no está muy explotado es en el ámbito de la cocina. Es fácil encontrar en las casas robots de cocina en los que introduces los alimentos necesarios para una receta y estos los cocinan el tiempo necesario, o bases de datos que te sugieren recetas, pero no cámaras u otros tipos de sensores que reconozcan las acciones de una persona y den consejos o sugerencias a partir de ellas.

Por ello este trabajo se centra en el diseño de un asistente de cocina que sea capaz de poderle introducir una receta y que te vaya guiando en la elaboración de esta.

Para ello el asistente tiene que ser capaz de detectar alimentos y utensilios, así como las acciones que está realizando el usuario.

1.1. Motivación

Este trabajo se ha realizado en colaboración con la empresa BSH Electrodomésticos España. Esta empresa es líder en su sector y apuesta por la investigación y la innovación. Es por ello que uno de los proyectos en colaboración con la Universidad de Zaragoza que tienen abierto es el desarrollo de un asistente de cocina, los objetivos y alcance son a largo plazo y bastante abiertos.

Por ahora lo que se pretende es el diseño de un sistema capaz de seguir a una persona mientras realiza una receta y poder guiarla dándole las instrucciones necesarias.

1.2. Estado de la materia y trabajo previo

Los asistentes han sido un campo de trabajo muy estudiado últimamente, y existen asistentes que ayudan en actividades muy variadas.

Algunos de los más conocidos hoy en día pueden ser *Google Assistant* (de *Google*), *Siri* (de *Apple*), *Cortana* (de *Microsoft*) o *Alexa* (de *Amazon*) [2].

Estos son asistentes por voz, suelen ir instalados en dispositivos como móviles, tabletas u ordenadores y su función es automatizar y realizar tareas con la mínima interacción hombre-máquina de una forma lo más natural posible. El usuario se comunica por voz, el asistente lo procesa, interpreta y responde de la misma forma.
Algunas acciones que se les pueden pedir son enviar mensajes de texto, realizar llamadas o dar indicaciones de cómo llegar a un lugar.

Este proyecto se centra en el entorno de la cocina en el cual también ha habido numerosos estudios sobre asistentes.
Algunos se centran en la preparación de recetas y los tiempos que conlleva cada paso para realizarlas [3]. Otros han desarrollado un sistema de diálogo hablado el cual da asistencia en la lectura del procedimiento y detalla todos los pasos que no están claros para usuarios con falta de experiencia [4].
También ha habido un estudio que ha desarrollado un sistema que ayuda a la hora de comprar ingredientes y planear las comidas que se van a llevar a cabo a lo largo de la semana para evitar el malgasto de comida [5].
Algo más parecido a este trabajo de fin de grado es un asistente que en base a las imágenes captadas por una cámara es capaz de reconocer las acciones que está llevando a cabo el usuario y avisar si falta alguna acción de realizar como guardar la leche en la nevera [6].

Este trabajo está contenido en un proyecto en colaboración con BSH en el que participan otras personas. Se empezó haciendo un estudio de las diferentes redes neuronales pre-entrenadas que había disponibles en *Matlab* eligiendo finalmente *ResNet-101* por ser la que en menos tiempo obtenía unos resultados de *precision* y *recall* mejores.
El siguiente paso fue el procesamiento de la imagen para segmentar los alimentos y objetos que el individuo llevaba en las manos y los que estaban encima de la mesa.
Para la detección en las manos se usó la tecnología de la cámara *Kinect V2* la cual es capaz de dar las coordenadas de 25 articulaciones con profundidad, mientras que para la detección en la mesa dados cuatro puntos introducidos por el usuario pertenecientes a la superficie se calcula el plano que la contiene y se detectan los objetos que hay por encima de dicho plano.
Una vez recortados y procesados se tratan con la red neuronal seleccionada, modificada y entrenada con las nuevas categorías referentes al entorno de la cocina.

1.3. Herramientas de trabajo

Para la captación de los videos se ha utilizado la cámara *Kinect v2* que es producida por *Microsoft* para uso en *Xbox 360, Xbox One* y en ordenadores *Microsoft Windows* y permite al usuario controlar e interactuar con su consola/PC con el uso de gestos y comandos por voz . Figura 1.

La elección de esta cámara es debida a su gran resolución a la hora de tomar imágenes tanto en color como en profundidad, así como que también es capaz de dar las coordenadas en 3D de 25 articulaciones de hasta seis personas simultáneamente [7].

Figura 1: Sensor Kinect v2

Para la programación tanto de la estructura de la red neuronal como de su entrenamiento y para el procesamiento de los videos se ha utilizado la herramienta de *software* matemático *Matlab.* Este programa ha sido elegido entre otras cosas por su compatibilidad con el sensor *Kinect v2* y por el gran número de librerías en los campos de redes neuronales y de visión por computador que contiene.

Para poder entrenar la red con las categorías necesarias se ha recurrido a la base de datos *ImageNet* [8] la cual está organizada siguiendo la estructura *WordNet.* Cada concepto significativo en *WordNet* es llamado "*synset*" y hay más de 100.000, mayoritariamente sustantivos. La base de datos *ImageNet* los recoge todos y ofrece un mínimo de 1000 imágenes por categoría o "*synset*". Estas imágenes se etiquetan mediante acción humana.

1.4. Estructura de la memoria

La estructura de la memoria está ordenada en función como se ha ido realizando el trabajo.

- El primer capítulo trata la red neuronal utilizada, el estudio previo sobre qué red pre-entrenada escoger y su entrenamiento.

- El segundo capítulo explica el seguimiento de objetos y el uso del filtro de Kalman y su funcionamiento en este trabajo.
- El tercer capítulo comenta el algoritmo que detecta que acción está llevando a cabo una persona o varias personas.
- El cuarto capítulo muestra los pasos para realizar una receta y como el asistente hace le seguimiento de dicha receta
- El quinto capítulo explica las pruebas a las que ha sido sometido el algoritmo y las dificultades encontradas.
- El último capítulo trata sobre las conclusiones a las que se ha llegado y las líneas futuras de trabajo.

2. Red Neuronal utilizada

Una red neuronal es un conjunto de capas formadas por "neuronas" e interconectadas entre sí. Hay una capa de entrada de dimensiones igual a los datos de entrada, una capa de salida de dimensiones igual al número de categorías que queremos clasificar y una serie de capas ocultas de las cuales su número de neuronas se considera un hiperarámetro. Estás neuronas que forman las capas tienen asociado un peso que se va actualizando mientras la red está siendo entrenada.
Cuando se utiliza una red lo que se hace es pasar una serie de datos, una imagen en nuestro caso, por todas las capas de la red las cuales extraen las características de dichos datos y acaban dando en la última capa las probabilidades sobre a qué categorías podría pertenecer.

Antes de comenzar a entrenar la red hay que separar el conjunto de imágenes con el que se va a trabajar en tres grupos diferentes: Imágenes de entrenamiento, imágenes de validación e imágenes de test.
Las primeras son el conjunto más amplio y sirven para estimar los parámetros del modelo. Las imágenes de validación comprueban como está siendo entrenado la red y corrigen algunos parámetros y el ultimo subconjunto se utiliza para una vez entrenada la red comprobar como de bien funciona con imágenes que nunca ha visto.
La división del conjunto de imágenes se ha hecho mediante *Matlab,* haciendo primero una partición aleatoria del 25% de cada categoría que se reserva como subconjunto de test y del 75% restante se ha realizado otra partición del 10% para imágenes de validación. Las imágenes restantes son las que forman el subconjunto de entrenamiento.

Para la clasificación de objetos en este trabajo se ha decidido utilizar una red pre-entrenada ya que es mucho más fácil y rápido que entrenar una red desde cero donde todos los pesos son iniciados de manera aleatoria. En una red pre-entrenada se pueden transferir características ya aprendidas para un nuevo uso utilizando un pequeño número de imágenes de entrenamiento.

2.1. Selección de la red pre-entrenada

Matlab ofrece una serie de redes neuronales convolucionales pre-entrenadas. Por ello, para escoger entre ellas se ha realizado un pequeño estudio con cuatro redes.

1. *AlexNet*
2. *GoogleNet*
3. *Resnet101*
4. *Inception-ResNet-v2*

Todos estos modelos han sido entrenados con un subconjunto de imágenes de la base de datos *ImageNet.* El estudio se ha llevado a cabo solo con cinco categorías diferentes para que no se incrementara demasiado el tiempo de entrenamiento de cada red.
Estas categorías han sido: *Banana*, *Broccoli*, *Garlic*, *Knife* y *Tomato.*

Para poder trabajar con estas redes hay que reemplazar las tres últimas capas de cada una ya que contienen información de cómo combinar las características que la red extrae en probabilidades sobre a qué clase o etiqueta corresponde.
Las tres nuevas capas que se añaden son: una capa totalmente conectada, una capa *softmax*, la cual contiene una función de normalización exponencial, y la capa de clasificación de salida.

Antes de entre pasar al entrenamiento hay que redimensionar las imágenes de entrada.
Por ejemplo, *AlexNet* requiere imágenes de 227x227x3 mientras que *GoogleNet* las requiere de 224x224x3. Para ello se ha modificado el tamaño sin variar la proporción inicial de la imagen y se ha añadido el espacio que quedaba en negro. Figura 2.

Figura 2: Redimensionado de una imagen

El estudio de las cuatro redes pre-entrenadas se ha hecho con un nivel de paciencia igual a cuatro. Esto quiere decir que cuando las pérdidas de validación no decrezcan en más de cuatro iteraciones el proceso de entrenamiento se para.

Los resultados del estudio de las cuatro redes han sido:

AlexNet:

Tiempo de entrenamiento: 4 min 10 sec

Número de iteraciones:1023

Precisión sobre las imágenes de test: 0.9338

GoogleNet:

Tiempo de entrenamiento: 14 min 21 sec

Número de iteraciones: 2139

Precisión sobre las imágenes de test: 0.9675

ResNet101:

Tiempo de entrenamiento: 39 min 21 sec

Número de iteraciones: 1336

Precisión sobre las imágenes de test: 0.9741

Inception-Resnet-v2:

Tiempo de entrenamiento: 70 min 31 sec

Número de iteraciones: 748

Precisión sobre las imágenes de test: 0.9657

Inception- v3:

Tiempo de entrenamiento: 144 min 47 sec

Número de iteraciones: 4429

Precisión sobre las imágenes de test: 0.9740

Para tener más información sobre los cuatro modelos también se ha sacado sus matrices de confusión para las cinco categorías entrenadas. Figura 3.

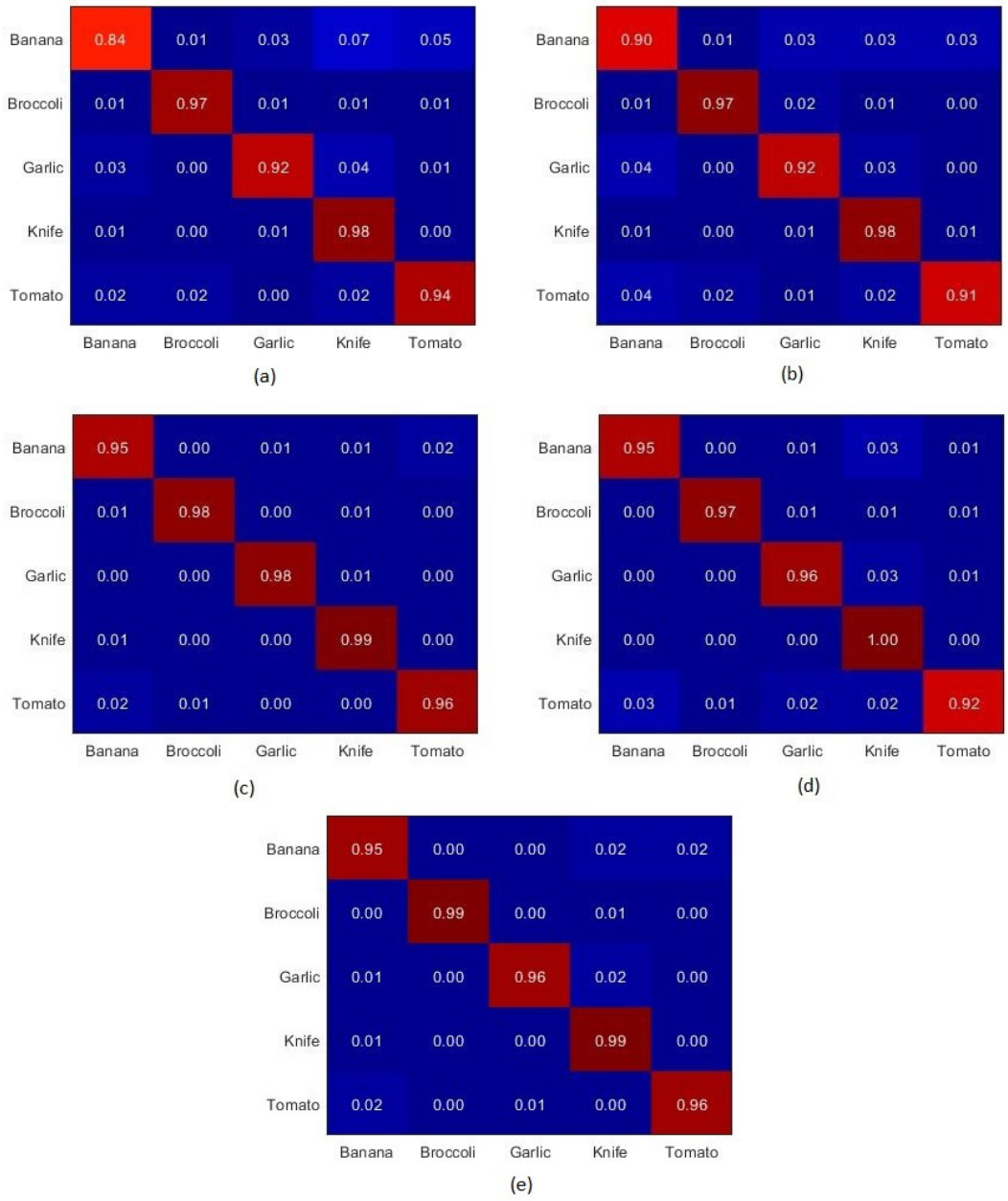


Figura 3: Matrices de confusión. (a) AlexNet. (b) GoogleNet. (c) ResNet101. (d) Inception-ResNet-v2. (e) Inception-v3

Con todos estos datos se puede observar que, aunque la red que menos tiempo tarda en converger es *AlexNet,* su precisión también es bastante menor y funciona peor que el resto.

Por otro lado, la segunda red que más ha tardado en converger ha sido *Inception-Resnet-v2,* aunque solo tiene mejor precisión que *AlexNet*.

El entrenamiento que más tiempo ha llevado ha sido con la red *Inception- v3.* Podría ser una buena elección ya que su precisión es casi tan buena como la de ResNet101, pero siendo que no es mejor que otras más rápidas también es descartada.
Solo quedan por tanto dos redes entre las que elegir, *GoogleNet* y *ResNet101.* La segunda es la que mejor precisión tiene, pero *GoogleNet* tampoco se diferencia mucho y es bastante más rápida.
Como el tiempo de entrenamiento de la red no es un requisito se ha optado por elegir en función de la precisión. Por ello la red convolucional pre-entrenada escogida ha sido *ResNet10.*

2.2. Entrenamiento de la red

Una vez seleccionada la red a utilizar, hay entrenarla con todas las categorías que se van a manejar.
Como se ha comentado en la introducción las imágenes han sido sacadas de una base de datos que se encuentra en internet llamada *ImageNet.*
Todas las categorías con las que se trabaja se pueden dividir en dos grupos: Ingredientes y utensilios de concina. Tabla 1.

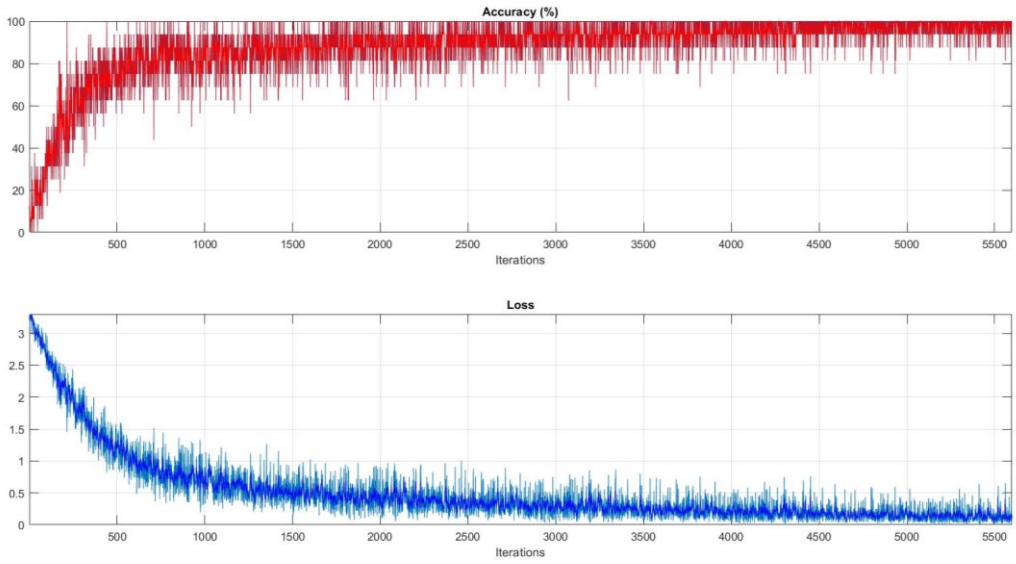
Ingredientes		**Utensilios**	
Artichoke	*Garlic*	*Bottle*	*Scissors*
Asparagus	*Lemon*	*Eggbeater*	*Spoon*
Banana	*Lettuce*	*Glass*	*Squeezer*
Bread	*Mushroom*	*Knife*	
Broccoli	*Pepper*	*Pan*	
Egg	*Tomato*	*Plate*	
Eggplant	*Zucchini*	*Pot*	

Tabla 1: Categorías utilizadas en el entrenamiento de la red neuronal

Las categorías de ingredientes han sido seleccionadas para tener una variedad de alimentos, en su mayoría verduras y frutas, que son fáciles de tener en casa y poder manipular.

En lo referente a los utensilios se han elegido en base a las acciones que se van a reconocer como cortar, exprimir, batir…

El entrenamiento de la red se ha realizado en 03:20 horas y 5598 iteraciones consiguiendo una precisión de validación del 88.89% y unas pérdidas de validación de 0.3833



Tras el entrenamiento, usando las imágenes que se han reservado para test, se puede calcular la *precision* total aplicando la red a este grupo. En este caso la *precision* es 0.9268

Para estudiar mejor cómo funciona la red con cada categoría se puede extraer la matriz de confusión. Figura 4.

En el eje vertical se muestran las etiquetas verdaderas mientras que en el eje horizontal aparecen las etiquetas que la red predice que son. Esto quiere decir que en la diagonal se encuentran los casos en los que la red acierta y cuanto mayor sea este número mejor será nuestra red.

Por el contrario, los porcentajes fuera de dicha diagonal muestran la proporción total de veces que la red se ha equivocado en cada categoría y que etiqueta ha dicho erróneamente que era.

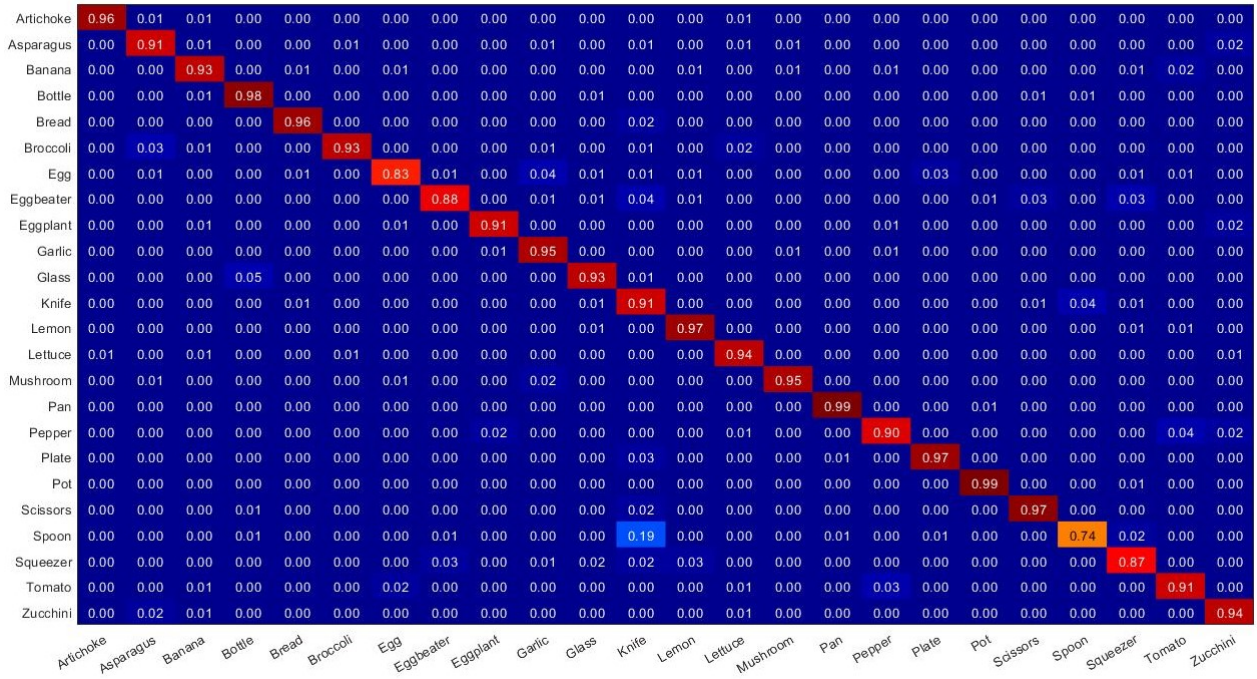


Figura 4 Matriz de Confusión

Se puede observar como en la Figura 4 hay categorías que tienen un muy buen resultado de acierto como por ejemplo, *Bottle* o *Pot* que tienen un 98 y 99% de acierto respectivamente. Mientras que otras como Spoon no funcionan muy bien. Para comprobar donde estaba el fallo se repasó las imágenes de test que habían sido confundidas viéndose que en varias imágenes aparecían, aparte de cucharas, cuchillos como se puede observar en la Figura 5.

Figura 5: Imágenes de la categoría Spoon mal predichas debido a que en ellas aparecen también objetos de la categoría Knife

Otros parámetros que se pueden observar sobre cada categoría son la *precision* y el *recall* o exhaustividad. Figura 6.

La *precision* es el porcentaje de verdaderos positivos sobre el total de positivos predichos mientras que el *recall* es el porcentaje de verdaderos positivos que se logran identificar.

En nuestro caso los verdaderos positivos son los elementos de la diagonal, los falsos positivos son la suma de los porcentajes de las columnas menos el valor de acierto y los falsos negativos corresponden con la suma de los porcentajes de las filas menos el valor de los aciertos.

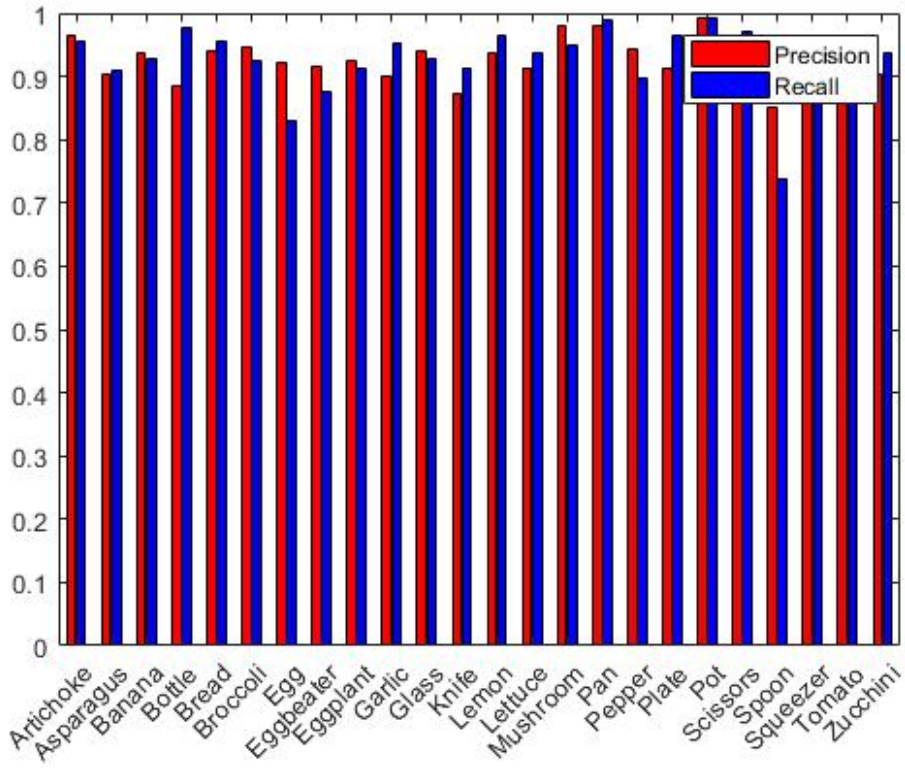


Figura 6: Precision y Recall En el caso de Recall se puede observar cómo es bastante bajo para la categoría Spoon.

Otro parámetro observado ha sido el índice de Kappa Cohen el cual ajusta el efecto del azar en la proporción de la concordancia observada y es usado para validar la calidad de los resultados [9].

Para calcularlo se calcula la probabilidad de acertar *po* menos la probabilidad de que acierte por azar *pe* y se divide todo entre uno menos la probabilidad del azar.

$$k = \frac{po - pe}{1 - pe}$$

Para nuestra red y con las imágenes de test el índice de Kappa Cohen es igual a 0.9243.

Antes de entrenar la red definitiva, se ha hecho otra de prueba con diferentes categorías para comprobar que funcionaba.
Una vez que decididas definitivamente las acciones, utensilios y alimentos necesarios se entrenó la red neuronal que se utilizaría finalmente.

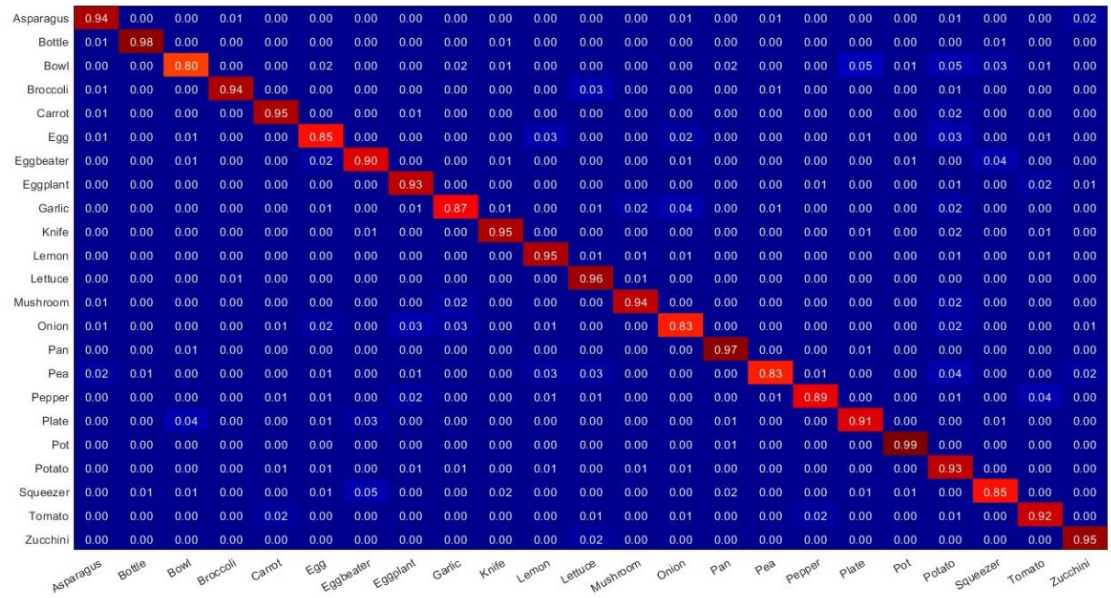


Figura 7: Matriz de confusión de la primera red entrenada

En la Figura 7 se muestra la matriz de confusión del primer entrenamiento.
Al no estar la categoría de cuchara parece que funcionaría bastante mejor pero su precisión era del 0.9286, la cual no dista mucho del 0.9268 de la otra red.

2.3. Clasificación

El código de segmentación está basado en un trabajo previo [10] como se ha mencionado en la Introducción aunque se han añadido algunas modificaciones.

Para clasificar los alimentos y utensilios que aparecen en la imagen primero se dirige la atención a las manos del usuario que aparece en el video. Este dato lo da la Kinect ya que es capaz de detectar 25 articulaciones de la persona a la que sigue. Figura 8.

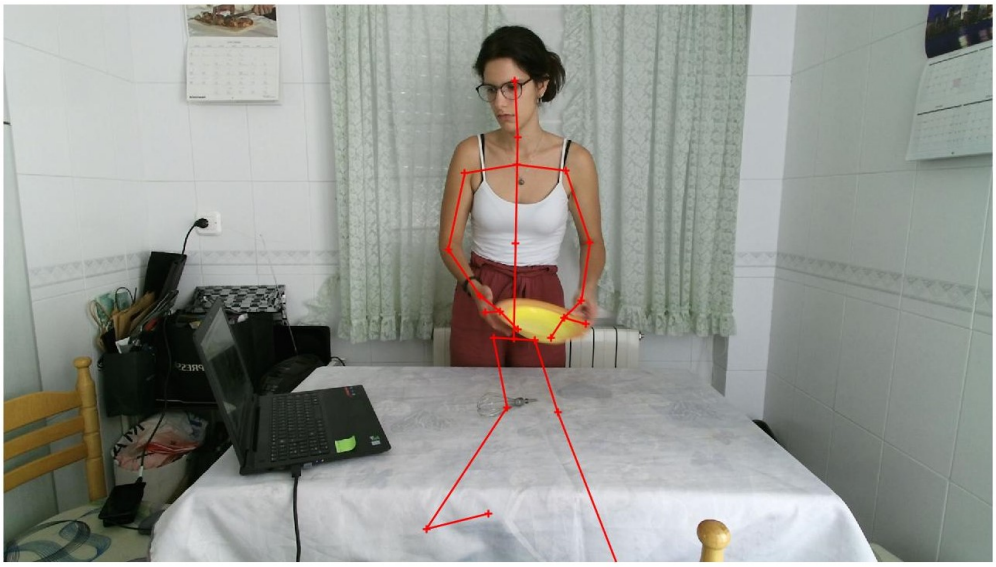


Figura 8: Seguimiento del usuario

A partir de los centroides de las manos y de su profundidad se saca una *BoundingBox*, que se redimensiona de la misma manera que las imágenes de la base de datos y se pasa por la red neuronal para que la etiquete.
Hay que tener en cuenta que si se lleva un objeto con dos manos el programa lo contaba dos veces, por lo que ha habido que hacer algunas modificaciones para que sea capaz, mediante la etiqueta del objeto y la distancia entre los centroides de las manos, de detectar que es el mismo objeto.

Una vez observadas las dos manos del usuario la atención se centra en los objetos encima de la mesa.
Para sacar el plano de la mesa antes de empezar a procesar los *frames* del video hay que marcar cuatro puntos que pertenezcan a dicho plano mediante una "calibración" previa. Una vez realizada esta acción se cogen todos los puntos que estén por encima de ese plano, se restan los pertenecientes al cuerpo del usuario (Figura 9), se sacan las *BoundingBox* de cada uno y se clasifican con la red neuronal.

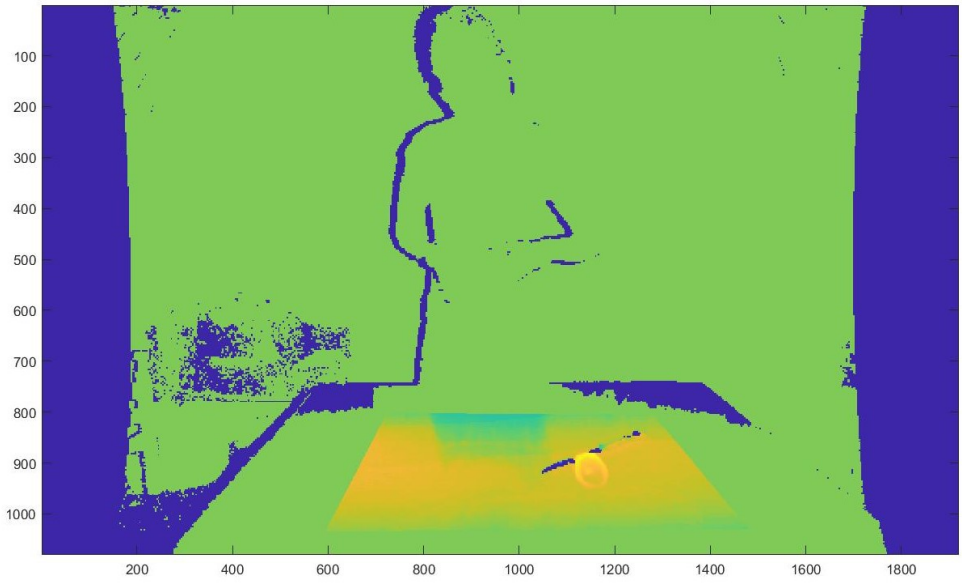


Figura 9: Detección de los objetos encima de la mesa

Cuando se tienen todos los alimentos y utensilios de la escena detectados, se almacenan en unas estructuras llamadas *tracks,* las cuales se explican en el siguiente capítulo.

3. Seguimiento de objetos

Una vez que obtenidos todos los objetos que aparecen en la escena hay que hacer un seguimiento de ellos para poder comprender las acciones que se pueden estar llevando a cabo con ellos.
Para poder hacer este seguimiento de manera correcta y sin que problemas de ruido afecten al resultado se ha aplico un filtro de Kalman.

3.1 Filtro de Kalman aplicado al seguimiento de objetos

Según el libro Kalman Filtering: Theory and Practice with MATLAB [11] el filtro de Kalman es un algoritmo que estima el estado de un sistema a partir de los datos medidos. Este algoritmo tiene dos pasos, el primero es la predicción del estado del sistema y el segundo utiliza las mediciones de ruido para ajustar la estimación del estado del sistema. Uno de los usos más comunes de este tipo de filtros es, en Visión artificial, el seguimiento de trayectorias que predice la ubicación futura del objeto a seguir.

En este proyecto se ha utilizado para el seguimiento de los alimentos y utensilios detectados tanto en las manos del usuario como en la mesa. Este problema se puede dividir en dos partes: la detección de los objetos en movimiento en cada *frame* y la asociación de las detecciones correspondientes al mismo objeto a lo largo del tiempo.

A cada objeto se le asocia una estructura llamada *track*. Esta estructura contiene, un numero identificador, el centroide del objeto, su *BoundingBox,* su etiqueta predicha por la red neuronal, su edad (la cual es el número de veces seguidas que ha sido visto), el número de veces seguidas que ha sido invisible y un índice que clasifica en que parte se encuentra, distinguiendo entre si lo lleva una persona o está encima de la mesa. El objetivo de esta estructura es mantener el estado del objeto seguido.

Mediante el filtro de Kalman se predice la posición de cada objeto en el siguiente *frame* y determina la probabilidad de que cada detección se asigne a cada *track*.
En cada *frame* algunas detenciones pueden ser asignadas a *track* mientras que otras detecciones y *track* podrían quedar sin asignarse. Los *track* asignados son actualizados usando las correspondientes detecciones mientras que los no asignados son marcados como invisibles. Una detección no asignada se convierte en un nuevo *track* y se hace

visible cuando su edad supera el umbral (en nuestro caso se ha fijado dicho umbral a cuatro).

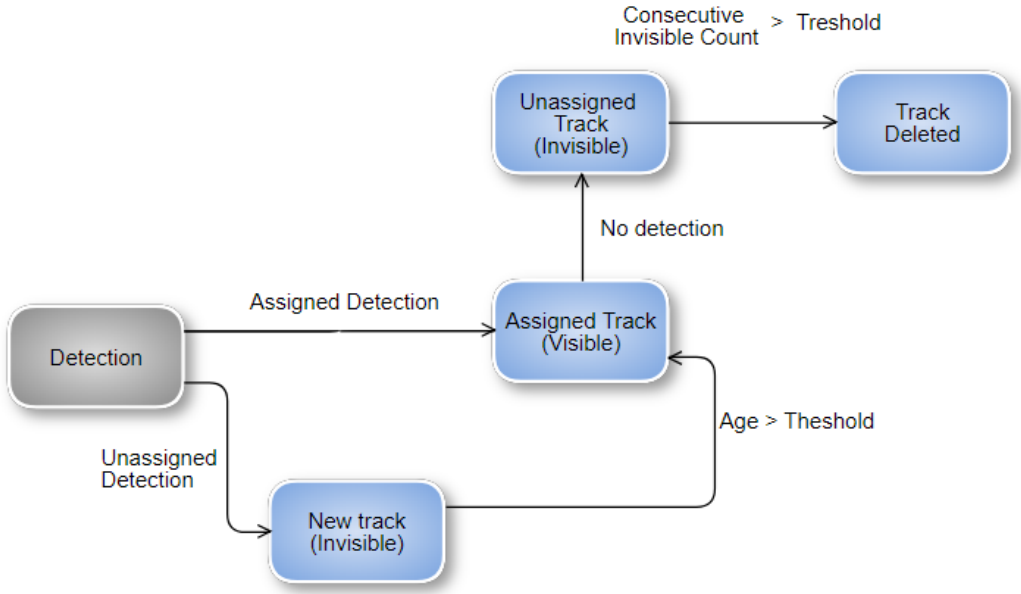


Figura 10: Esquema sobre el funcionamiento del filtro de Kalman

Cada *track* guarda el número de *frames* consecutivos en el que permanece sin asignar. Si este número supera un umbral el algoritmo asume que el objeto ha dejado de ser visible y borra el *track*. En la Figura 10 se ha representado un pequeño esquema sobre el funcionamiento del filtro de Kalman para tratar de hacer más sencilla su comprensión.

Las funciones usadas están implementadas en *Mathworks* y son:

```
predictNewLocationsOfTracks()
detectionToTrackAssignment()
updateAssignedTracks()
updateUnassignedTracks()
deleteLostTracks()
createNewTracks()
```

Aplicando este filtro se impide que, aunque en un *frame* determinado aparezcan resultados extraños por culpa del ruido o por una mala posición de un objeto que impida el correcto reconocimiento, el resultado no se vea afectado.

4. Seguimiento de personas

4.1. Seguimiento de una persona

Para poder ayudar al usuario en el procedimiento de elaboración de una receta es necesario saber qué acciones está realizando la persona en cada momento. Es por eso por lo que en cada *frame* hay que prestar atención al estado de la persona y a los objetos que lleva en sus manos y que le rodean.

Para estimar el estado del usuario que aparece en pantalla se han implementado condiciones.
En un principio se pensó en hacerlo también mediante redes neuronales, pero la base de datos más acorde que se encontró fue *Cornell Activity Datasets: CAD-60 & CAD-120* En esta base de datos las acciones relacionadas con la concina son muy limitadas y además los usuarios aparecen realizando dichas acciones de perfil.
Otra opción habría sido el crear nuestra propia base de datos, pero ello requiere mucho tiempo de grabaciones.
Es por ello por lo que, como se iba a empezar por un número pequeño de acciones a reconocer, se escogió la opción de clasificarlas según condiciones.

El primer paso que realiza la función es reconocer si la persona está en movimiento o quieta. Para ello mira las posiciones actuales de los centroides de las manos en tres dimensiones y los de tres *frames* anteriores. Si hay una distancia mayor a un valor escogido quiere decir que las manos del usuario se han movido y que por tanto está realizando alguna acción.
Una vez detectado el movimiento se mira si la persona lleva algún objeto en las manos.
Si no es así, aparece sobre el usuario una etiqueta en la cual se lea *moving.*
Si por el contrario sí que lleva, pasamos a comprobar si cumple los requisitos que se han especificado de alguna de las acciones que se buscan.
Las acciones que se van a reconocer son: **Cortar con cuchillo** o **tijeras algún alimento**, **Exprimir un limón**, **Batir con varillas**, **Coger una cucharada** y **Llevar uno o dos alimentos/utensilios**.

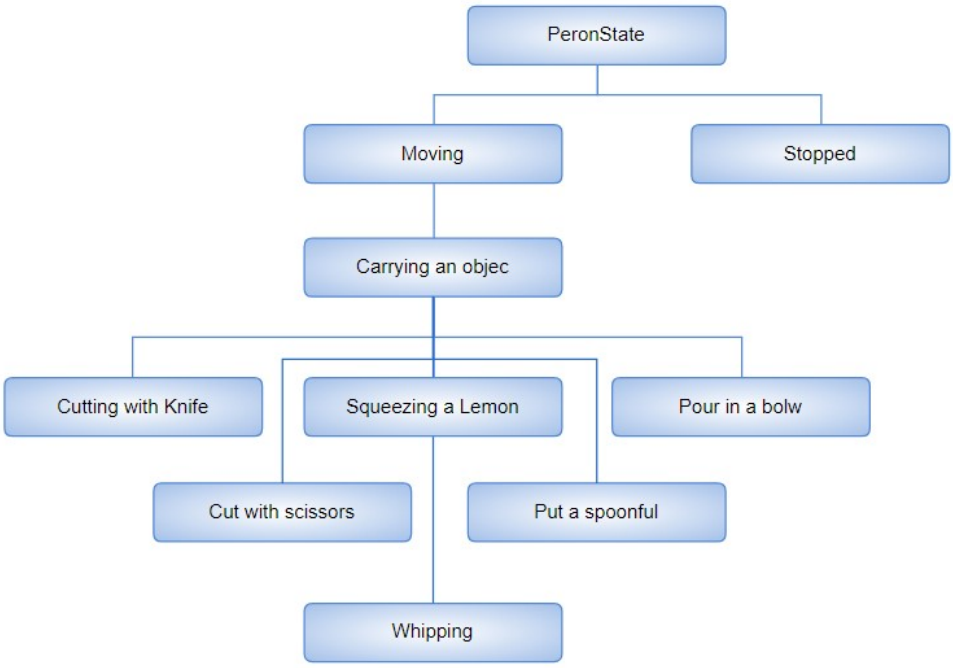


Figura 11: Estructura de la función PersonState

Las condiciones para cada una de las acciones son las siguientes:

- Cortar con cuchillo algún alimento

En una de las manos el usuario tiene que llevar un cuchillo, en la otra mano tiene que llevar un alimento, las manos no pueden estar separadas entre ellas más de una cierta distancia y no pueden estar lejos de la superficie de la mesa.

- Cortar con tijeras algún alimento

Es igual que el anterior solo que en vez de un cuchillo la herramienta requerida son unas tijeras y la condición de cercanía a la mesa es eliminada.

- Exprimir un limón

El usuario tiene que llevar en la mano un exprimidor, en la otra un limón y no pueden estar separadas más de una cierta distancia.

- Batir con varillas

Si en una mano el usuario lleva unas varillas, en la otra lleva un plato y sus manos están bastante juntas la función da como resultado que está batiendo.

- Coger una cucharada

En una mano el usuario tiene que llevar una cuchara y por debajo y cerca del centroide de esa mano tiene que haber un recipiente (vaso, cacerola o plato).

Este caso es un poco más complejo porque no solo entran en juegos los objetos que tiene el usuario en las manos, sino que también hay que prestar atención a los recipientes que se hallen encima de la mesa.

- Llevar uno o dos alimentos/utensilios

Si la persona se está moviendo y lleva algún objeto, pero no cumple ninguna de las acciones anteriores puede deberse a que está realizando una acción no programada o que simplemente está llevando dicho objeto.

Un ejemplo del reconocimiento de acciones lo podemos encontrar en la Figura 12 donde aparece el usuario cumpliendo todas las condiciones que conlleva la acción de llevar una cuchara.

Figura 12: Adding a Spoonful

En la Figura 13 se puede observar la diferencia entre cortar con cuchillo y cortar con tijeras. En la imagen de arriba las manos tienen que estar cerca de la mesa y en la segunda las manos pueden encontrarse en cualquiera parte de la escena.

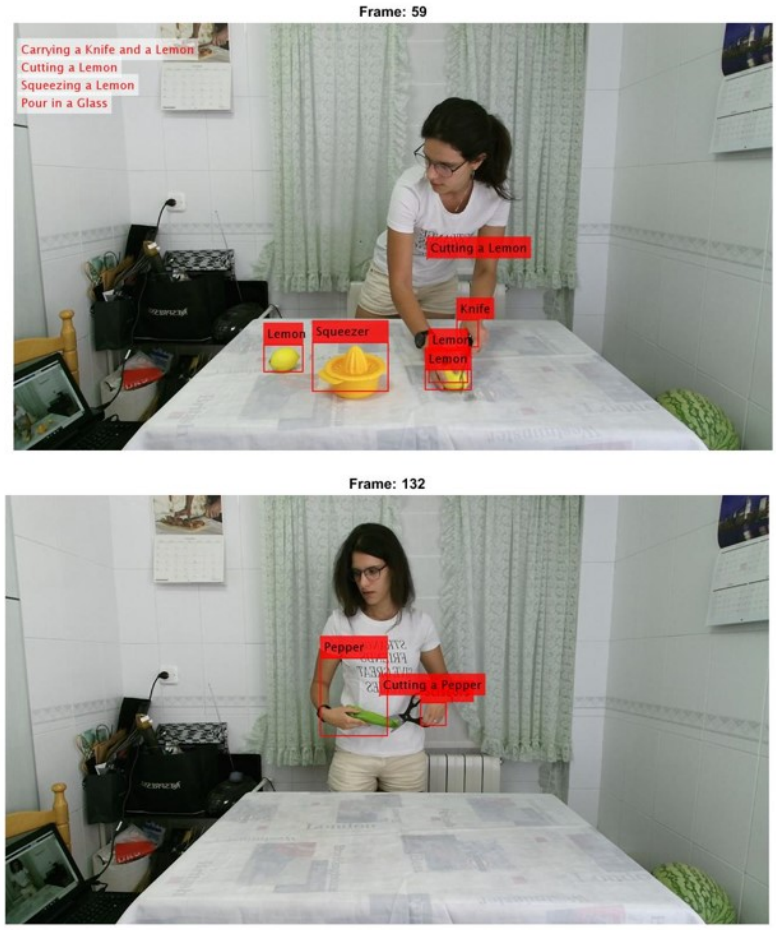


Figura 13: Acción de cortar tanto con cuchillo como con unas tijeras

Una vez detectada la acción que está llevando a cabo el usuario solo queda comparar con los pasos que hay que seguir para la elaboración de la receta.

4.2. Seguimiento de dos personas

No es inusual que en una receta participen dos personas al mismo tiempo. Además, teniendo en cuanta que la Kinect es capaz de seguir hasta dos personas se ha modificado el código para que pueda seguirla simultáneamente.

En este caso ya no solo hay que mirar a las dos manos de un usuario, sino que hay que mirar también a las de la otra persona que aparece en la escena y clasificar los objetos en función de quien los lleve. Esta diferenciación es muy importante ya que se utiliza para definir el estado o acción que está llevando a cabo cada persona.

Las normas seguidas para a clasificación son las mismas que en el apartado 4. Seguimiento de personas

4.1. Seguimiento de una personaEn este caso esta clasificación hay que realizarla dos veces, una para cada individuo. Otra modificación que se ha realizado ha sido en la estructura de los *tracks* ya que, el índice que clasifica en que parte se encuentra el objeto a seguir ahora tiene que diferenciar si está en las manos de la primera persona, en las manos de la segunda o si se encuentra encima de la mesa.

Figura 14: Seguimiento de dos personas en la misma escena.

En la Figura 14 se puede observar como el algoritmo modificado también es capaz de detectar los objetos que manipulan las dos personas y predecir la acción que están realizando.

El único problema es que, al haber mucha más información en la imagen, de haber dos manos en la que mirar si hay objeto y clasificarlo, se pasa a cuatro, el tiempo de procesamiento de la imagen se incrementa bastante.

Por ejemplo, el tiempo tardado en procesar setenta *frames* del video de añadir una cucharada fue de 199.6028 segundos, unos 2.8515 segundos por cada imagen. Por otro lado, procesar setenta *frames* del video en el que aparecen dos personas fue de 394.027 esto supone un incremento de 2.779 segundos por cada *frame*

5. Elaboración supervisada de alimentos

Este proyecto incluye además el seguimiento de una persona a la hora de realizar una receta con indicación de los ingredientes necesarios y los siguientes pasos a seguir. Es por eso por lo que al algoritmo que realiza dicho papel se le ha llamado asistente.

5.1 Creación de la receta

Para la realización de la receta se ha utilizado una función que empieza preguntando al usuario por pantalla el número de ingredientes, que enumere los ingredientes y las cantidades y a continuación le pregunta por el número de pasos y los pasos a seguir.
Toda esta información se guarda en un fichero que puede ser leído a la vez que la red neuronal o que el video con el que se trabaja.

5.2 Seguimiento de la receta

El objetivo de este asistente es guiar al usuario mientras sigue una receta por ello nada más empezar muestra por pantalla los alimentos necesarios y los busca encima de la mesa o en las manos del usuario.
Una vez detectado que están presentes todos los alimentos lo muestra por pantalla y seguidamente escribe la lista de pasos que deberá realizar el usuario.
A partir de ese momento empieza a observar en cada *frame* el estado de la persona y se analiza si es el mismo que el paso que toca realizar. Si son iguales marca que el paso ha sido realizado y comienza a buscar el siguiente. Esto se ve reflejado en la Figura 15.

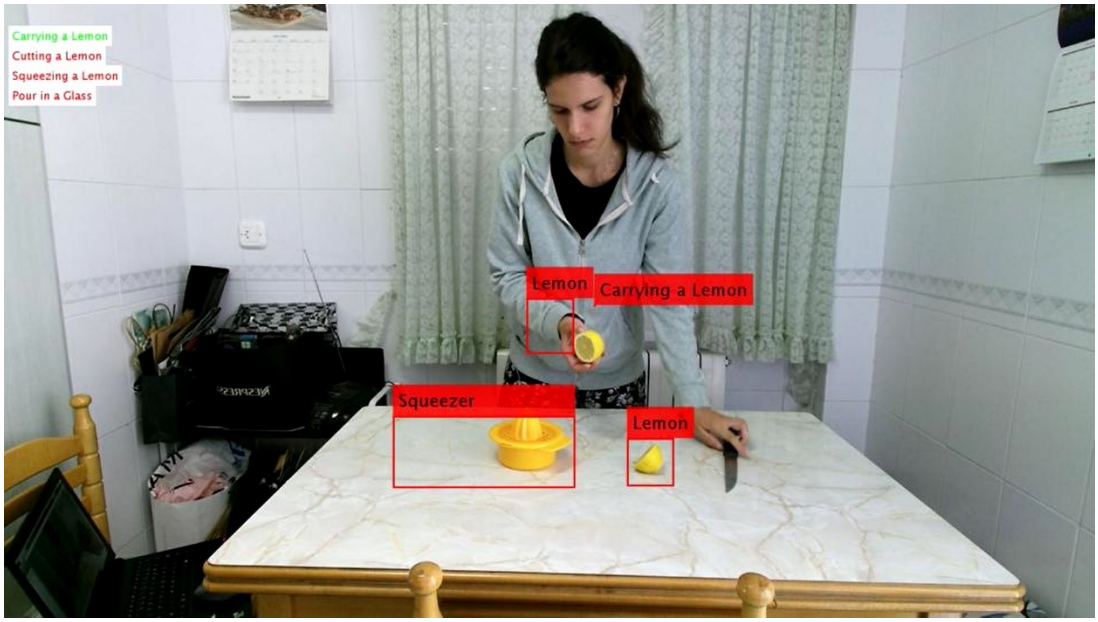


Figura 15: Seguimiento de los pasos de la receta

Puede darse el caso que el asistente no detecte un paso que haya realizado el usuario, por ello pasado un cierto tiempo (30 *frames* en este caso) se pregunta por pantalla si se ha ejecutado ya ese paso, si el usuario responde que si por teclado el asistente marca el paso como completado y pasa al siguiente mientras que si la persona escribe que no se vuelven a esperar otros 20 *frames*.

En el caso de que haya dos personas colaborando en la realización de la receta se presta atención a las dos acciones que se estén realizando y si alguno de los estados de los usuarios coincide con el paso que se está buscando en ese momento se marcará como realizado y se continúa buscando el siguiente.

6. Pruebas y experimentos

Para la realización de este trabajo se han hecho una serie de pruebas mediante la cámara *Kinect v2*.

Estas pruebas se han realizado en cocinas reales con diferentes usuarios con complexiones distintas para asegurar que es capaz de detectar y seguir a personas muy diversas.

Los elementos que se usan en los videos son utensilios de cocina y alimentos reales y las ropas de los individuos también van variando.

La mayoría de los videos se han realizado con un usuario, aunque también se han hecho varias pruebas con dos personas.

Se han hecho pruebas de todas las acciones que puede identificar el asistente, así como de la mayoría de los elementos que es capaz de diferenciar la red neuronal.

También se ha realizado diversas recetas a seguir, aunque la que más se ha tratado ha sido la receta de limonada por su simplicidad. Como ingredientes solo necesita limones y pasos que sigue son

1. Llevar limón
2. Cortar limón
3. Exprimir limón
4. Echar en un vaso

En la imagen superior de la Figura 13 se pueden observar como el algoritmo está buscando el primer paso de dicha receta y aunque el usuario está realizando el segundo paso, este no es marcado como completado.

Un gran problema encontrado al grabar videos con la Kinect era que, si el ordenador se veía sobrecargado, al haber diferentes flujos de datos grabándose simultáneamente, los *frames* de color no coinciden con los *frames* de profundidad y los de seguimiento del cuerpo grabados. Esto se traducía en que, al mandar las imágenes de los objetos en las manos para su etiquetado a la red neuronal, como los centroides de las manos de las imágenes de color eran diferentes de las otras se enviaban a la red cosas sin sentido como trozos de la mesa, del brazo o del fondo.

Para intentar solucionar este problema se trató de hacer un *trigger* manual mediante un bucle *for*. El problema de esta solución fue que, al ir lento el ordenador, el bucle tampoco tiene mucha velocidad y los *frames* tenían una frecuencia de captura muy baja por lo que los videos no parecían reales.

Grabar los videos ha requerido bastante tiempo ya que al problema anterior había que añadirle dos imprevistos más. El primero era que había veces que la cámara no era capaz de conseguir todos los pixeles de profundidad dando errores en la detección de los objetos en la mesa, pues se le pasaba a la red neuronal zonas en las que solo había trozos en blanco y esa los clasificaba de manera aleatoria. En la Figura 16 puede verse como por detrás del vaso y de la cuchara hay una zona de puntos más oscuros que representan que esos puntos no tienen valor numérico en la imagen en profundidad.

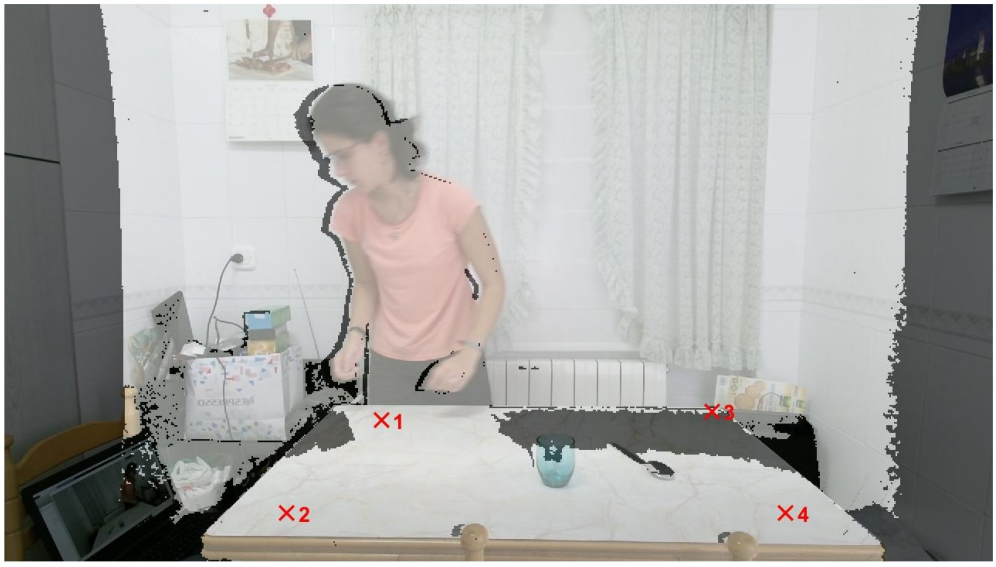


Figura 16: Problemas con la profundidad. Se puede apreciar como hay muchos puntos de la mesa y de los objetos que hay sobre ella que no son captados correctamente.

En esta imagen se puede observar también la "calibración" de la que se habla en el apartado 2.3 y que consiste en seleccionar cuatro puntos pertenecientes al plano de la mesa. Dichos puntos tienen que cumplir la condición de que dentro del cuadrilátero que formen estén los alimentos y utensilios a detectar.

El segundo problema que surge por la cámara es que, al no aparecer las piernas del usuario en la imagen, ya que este se encuentra detrás de la mesa, hay veces que confunde las articulaciones de estas extremidades con las de las manos. Esto se traduce en que piensa que la mano está en el antebrazo y por tanto la imagen que se le envía a la red neuronal no es la que debería.

En la Figura 17 puede apreciarse este efecto ya que se ve como sí que detecta la cuchara, pero al captar la cámara los datos del seguimiento de las articulaciones de la persona mal, se transmite una imagen del antebrazo en vez de la mano a la red neuronal y esta lo clasifica como un cuchillo.

Figura 17: Mala detección de las articulaciones

Otro problema añadido por la falta de procesador del ordenador era que como la Kinect graba varios flujos de información y con bastante calidad cada *frame* tiene bastante tamaño. Por tanto, cuando se cargan los videos completos si estos tienen bastante tamaño el ordenador se cuelga antes de poder procesar ninguna escena.

Es por ello por lo que la duración de los videos es tan pequeña y no se ha podido hacer el seguimiento de recetas con mayor número de pasos.

7. Conclusiones y líneas futuras de trabajo

El objetivo principal de este trabajo era el desarrollo de un asistente de cocina que detectara las acciones del usuario a la hora de realizar una receta y le fuera guiando en su desarrollo.

Para llevar esto acabo ha sido necesario un estudio más a fondo sobre redes neuronales y cómo funcionan, así como modificar una red neuronal pre-entrenada y qué parámetros son interesantes a la hora de ver si funciona correctamente como la *precision*, el *recall* o el índice de Kappa Cohen.

Para la parte de segmentación se ha tenido que inspeccionar el código del que se partía entendiendo todas las operaciones que se realizaban a las imágenes tanto de profundidad como de color con ayuda de la información de las articulaciones de la persona seguida. Además, también se han tenido que modificar varios parámetros para mejorar segmentación y por tanto la detección de los objetos pertenecientes a la escena.

Como futuro trabajo el mayor paso sería cambiar la plataforma que procesa cada *frame* ya que Matlab no es un programa de tiempo real y procesar un video de 150 *frames* puede llegar a costarle unos seis minutos de media (dependiendo de cuantos objetos haya, cuantos individuos aparezcan en la imagen…).

Es por ello por lo que para poder hablar de un asistente de cocina que ayude en el seguimiento de recetas esto habrá que mejorarlo.

Otra mejora que se puede introducir es la interacción por comandos de voz. En vez de que nos muestre los ingredientes necesarios o el siguiente paso por pantalla o que tengamos que escribir si hemos realizado un paso o no, lo ideal sería poder interactuar con el asistente por medio de voz ya que cuando una persona está cocinando suele tener las manos ocupadas.

También sería una buena opción crear una base de datos con grabaciones de varias acciones pertenecientes al entorno de la cocina y entrenar una red neuronal con ellas. De esta manera la detección de acciones se realizaría de forma automática y sin tener que establecer unas normas para cada acción.

Bibliografía

[1] "https://www.bmw.es/es/coches-bmw/serie-7/sedan/2015/funcionalidad-innovadora.html," [Online].

[2] "https://nae.es/tendencias-globales-para-el-mercado-de-los-asistentes-virtuales/," [Online].

[3] N. F. Yukiko Matsushima, "Extensions of Cooking Guidance Function on Android Tablet for Homemade Cooking Assistance System," *Dept. of Electrical and Communication Engineering, Okayama University.*

[4] L. F. Filipe Martins, "STARTING TO COOK A TUTORING DIALOGUE SYSTEM," *Spoken Language Systems Laboratory, L2F – INESC-ID IST / Technical University of Lisbon.*

[5] M. F. J. Veranika Lim, "Design Implications for a Community-based Social Recipe System," *Designed Intelligence Group Department of Industrial Design Eindhoven University of Technology The Netherlands.*

[6] J. Z. Chenxia Wu, "Watch-n-Patch: Unsupervised Learning of Actions and Relations".

[7] "http://www.kinectfordevelopers.com/es/2014/01/28/caracteristicas-kinect-2/," [Online].

[8] "http://image-net.org/index," [Online].

[9] P. d. V. Abraira V, "Generalization of the kappa coefficient for ordinal categorical data, multiple observers and incomplete designs.," *Qüestiió,* 1999.

[10] J. Blasco, "Deteccion y reconocimiento de alimentos mediante el uso de imagenes de profundidad y redes neuronales convolucionales," *Trabajo Fin de Master, Universidad de Zaragoza,* 2018.

[11] A. P. A. S. Grewal, . Kalman Filtering: Theory and Practice with MATLABv.

Tabla de ilustraciones

Figura 1: Sensor Kinect v2 7
Figura 2: Redimensionado de una imagen 10
Figura 3: Matrices de confusión. (a) AlexNet. (b) GoogleNet. (c) ResNet101. (d) Inception-ResNet-v2..... 12
Figura 4 Matriz de Confusión..... 15
Figura 5: Imágenes de la categoría Spoon mal predichas debido a que en ellas aparecen también objetos de la categoría Knife 15
Figura 6: Precision y Recall En el caso de Recall se puede observar cómo es bastante bajo para la categoría Spoon..... 16
Figura 7: Matriz de confusión de la primera red entrenada..... 17
Figura 8: Seguimiento del usuario..... 18
Figura 9: Detección de los objetos encima de la mesa 19
Figura 10: Esquema sobre el funcionamiento del filtro de Kalman 21
Figura 11: Estructura de la función PersonState 23
Figura 12: Adding a Spoonful 24
Figura 13: Acción de cortar tanto con cuchillo como con unas tijeras..... 25
Figura 14: Seguimiento de dos personas en la misma escena. 26
Figura 15: Seguimiento de los pasos de la receta 28
Figura 16: Problemas con la profundidad. Se puede apreciar como hay muchos puntos de la mesa y de los objetos que hay sobre ella que no son captados correctamente. 30
Figura 17: Mala detección de las articulaciones..... 31