Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Sistema de mejora de la extracción de esqueletos de Kinect mediante el algoritmo YOLO

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Héctor Martínez Cebrián

Tutor: Francisco José Abad Cerdá **Cotutor**: Randy Gomez

2017-2018

Resumen

Kinect es el dispositivo más empleado para la extracción de esqueletos a partir de imágenes de personas en movimiento. Sin embargo, la inexactitud de los datos que proporciona limita la calidad de los estudios que se basan en Kinect. Numerosas líneas de investigación tratan de encontrar un método que mejore la exactitud y utilidad de los datos que proporciona el dispositivo. Este trabajo presenta el uso conjunto y personalizado de *DepthYOLO* y *Decision Maker* como método alternativo de extracción de esqueletos basado en *deep learning* y a partir de la información que ofrece Kinect. Los resultados indican una clara mejoría respecto a los resultados brutos que ofrece Kinect si bien existe margen de mejora, sobre todo en el procesamiento que hace Decision Maker.

Palabras clave: extracción de esqueletos, redes neuronales, aprendizaje profundo, Kinect, YOLO

Abstract

Kinect is the most used device for skeleton extraction from images of people in motion. However, it renders inaccurate data that limit the quality of research based on Kinect. Many lines of research try to find a method which improves the accuracy and usability of Kinect data. This work explores the use of DepthYOLO and Decision Maker in a joint and customized way as an alternative method of skeleton extraction based on deep learning applied to the information provided by Kinect. Results show a clear improvement of accuracy in skeleton extraction compared with raw data provided by Kinect, but there is still room for improvement especially regarding data processing made by Decision Maker.

Keywords : Skeleton extraction, neural network, deep learning, Kinect, YOLO

Tabla de contenidos

Índice de ilustraciones

1. Introducción

El dispositivo Kinect, desarrollado por Microsoft inicialmente para detectar la posición y el movimiento del jugador en interacción con la consola Xbox 360, se emplea en gran variedad de proyectos como dispositivo para la extracción de esqueletos debido a su bajo coste. Sin embargo, Kinect tiene problemas a la hora de detectar algunas partes del cuerpo del sujeto y tampoco tiene mecanismos para almacenar datos obtenidos anteriormente, de manera que los datos que ofrece son inexactos. Detectar al sujeto en posiciones no posibles físicamente o inventar movimientos bruscos irreales son los errores más comunes producidos por Kinect. Para mejorar la exactitud de los datos y su utilidad para el estudio del movimiento humano se han propuesto diferentes alternativas con resultados diversos. Una de las líneas más prometedoras es la aplicación de *deep learning*. Dentro de ese campo, los módulos *DepthYOLO* y *Decision Maker*, basados en redes neuronales convolucionales, pueden ser una alternativa que permita obtener esqueletos más ajustados a la realidad, disminuyendo los errores provocados por Kinect. El presente trabajo explora la aplicación de esta tecnología, que se ha personalizado para acomodarla al objeto de la investigación.

En este capítulo inicial se mostrarán las razones que llevan al desarrollo del presente proyecto, además de los objetivos a perseguir y proponer unos pasos para cumplir los objetivos deseados.

Se ha redactado esta memoria de manera que pueda ser leída cómodamente por técnicos y profesionales relacionados con el área de la informática, aún sin tener formación específica en inteligencia artificial, redes neuronales o *deep learning*. En el capítulo 9 se detallan aquellos términos técnicos más específicos o característicos que puedan resultar menos conocidos para el lector.

Indicar también que este proyecto se ha realizado en el entorno de la empresa *Honda Research Institute Japan*. Por tanto, la información expuesta en forma de texto, ilustraciones, datos o resultados se encuentran limitados por un acuerdo de confidencialidad.

1.1. Motivación

Kinect es un dispositivo formado por una cámara de captación de imágenes a color (RGB), un proyector láser infrarrojo conjunto a un sensor de infrarrojos para captar la profundidad del escenario y un conjunto de micrófonos. Este dispositivo salió a la venta en Estados Unidos en noviembre de 2010. Se vendieron 8 millones de dispositivos durante los primeros 60 días, obteniendo el record Guinness como periférico para juegos más rápidamente vendido (Records, 2011) y alcanzó las 24 millones de unidades vendidas a inicios de 2013 (Epstein, 2013). El dispositivo capturaba vídeos RGB con una resolución de 640x480 píxeles mientras que las imágenes de profundidad contaban con una resolución de 320x240 píxeles (Plunkett, 2010).

En mayo del 2013 apareció en el mercado la segunda versión de este dispositivo, llamada *Microsoft Kinect for Windows v2*, junto a la nueva consola Xbox One de Microsoft. Para esta nueva versión, se emplearon cámaras RGB y sensores de profundidad de mayor resolución pasando a 1920x1080 píxeles para imágenes a color y a 512x424 píxeles en el caso de imágenes de profundidad (Lachat, et al., 2015).

Se detuvo la fabricación de dispositivos Kinect en octubre del 2017. En total 35 millones de dispositivos fueron vendidos en la vida total del producto (Reisinger, 2017). A partir de entonces, Microsoft invitó a los desarrolladores e investigadores que emplearan las cámaras de profundidad RealSense de Intel, ya que *Kinect for Windows* no sería actualizado ni se realizaría ningún tipo de servicio técnico por parte de Microsoft (Microsoft, 2018).

Además de las características mostradas anteriormente, Kinect es capaz de extraer información sobre las articulaciones y extremidades del sujeto que se encuentre frente al sensor. A esta información se le denomina comúnmente *esqueleto* mientras que las extremidades y las articulaciones se denominan sencillamente *articulaciones* (véase la ilustración 1).

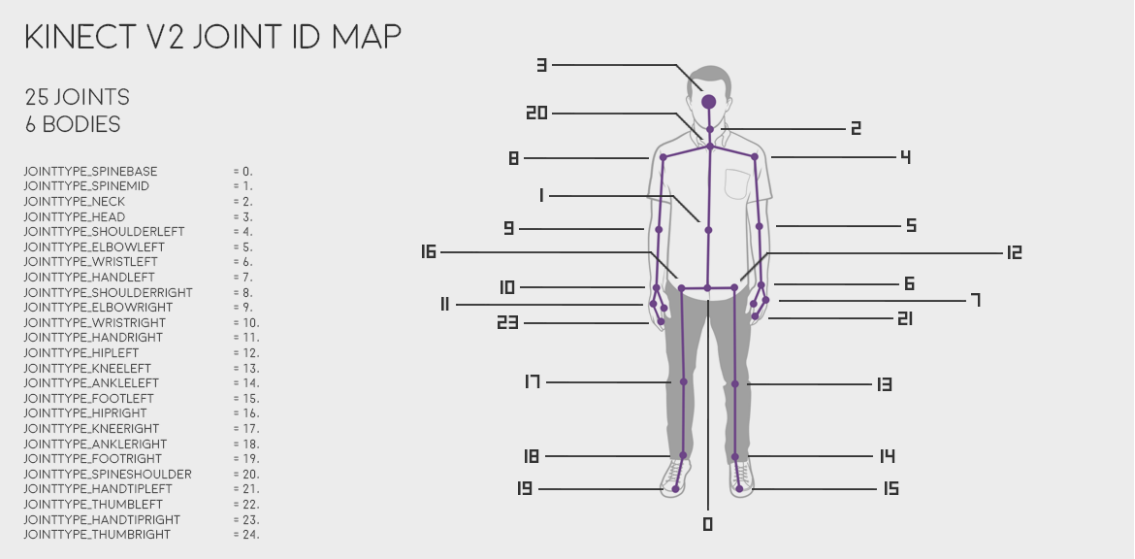

Ilustración 1 – Gráfico con el esqueleto formado por las correspondientes articulaciones (Vulliet, 2017)

Este dispositivo estaba destinado a ser empleado en el mundo de las consolas y videojuegos. Sin embargo, su uso se ha expandido a lo largo del mundo por muchos laboratorios de investigación tanto públicos como privados. Esto se debe a que el dispositivo tiene un precio asequible y además se encuentra fácilmente en el mercado.

Pese a ser un dispositivo tan comercializado, tiene una serie de importantes inconvenientes o problemas. Los que afectan en mayor medida a los investigadores son aquellos relacionados con la incorrecta extracción del esqueleto ya que habitualmente se emplea esta información para el reconocimiento de gestos o para interactuar con los diversos sistemas mediante una interfaz de usuario adecuada. Un esqueleto que difiere

excesivamente de la realidad puede afectar en gran medida a los resultados de las investigaciones, bien porque la entrada no sería correcta o porque no se pueden establecer estos esqueletos como línea base de referencia (*Ground truth*) para su comparación con otros sistemas de extracción.

Se pueden clasificar los fallos de la extracción de esqueletos del sistema Kinect en cuatro puntos:

1. El problema del fantasma. El sistema devuelve información sobre un esqueleto cuando en realidad no se encuentra ningún sujeto en escena. Puede aparecer durante unos pocos *frames* correlativos o bien se mantiene durante largos periodos.

2. Esqueleto espasmódico. Kinect genera información sobre un esqueleto sin tener presente la posición que tenía el esqueleto en el *frame* anterior. Por ello, entre *frames* consecutivos pueden aparecer inconsistencias relativas a la posición de una articulación, restableciéndose habitualmente en el siguiente *frame*.

3. Gestos imposibles. Kinect no dispone de ninguna manera de controlar si la información sobre el esqueleto que retorna es coherente a las limitaciones físicas. En ocasiones, se pueden encontrar partes del esqueleto que atraviesan otras partes del esqueleto, como una mano atravesando el brazo contrario o las rodillas cruzadas con ángulos imposibles.

4. Fallo en el reconocimiento del lateral. Cuando el sujeto se encuentra en movimiento, en algunas situaciones se intercambian las posiciones de algunas articulaciones por la articulación del lateral contrario. Es decir, confundir la mano derecha por la izquierda y viceversa.

Además, se puede señalar un problema adicional del sistema de extracción de esqueletos de Kinect. Cuando el sujeto se sitúa de espaldas al sensor, el sistema no es capaz de reconocer ninguna de las articulaciones y por tanto no ofrece esqueleto alguno como resultado de la extracción. A veces, en un intento de reconocer algunas articulaciones en estas situaciones, Kinect retorna una posición interpolada sobre otras articulaciones del esqueleto, donde se suele cometer el fallo sobre el reconocimiento del lateral (Punto 4 de los problemas presentes en Kinect).

1.2. Objetivos

El objetivo principal del presente Trabajo Fin de Grado es desarrollar un sistema capaz de corregir el esqueleto devuelto por el sistema Kinect o bien emplearlo en la creación de un nuevo esqueleto. Además, debe reducir al mínimo el posible retraso producido por costes computacionales ya que el sistema debe ser capaz de ser ejecutado en tiempo real. Por último, dado el actual interés en la comunidad científica, se empleará algún sistema basado en redes neuronales, los cuales también resultan de interés desde el punto de vista de emplear los conocimientos adquiridos durante el desarrollo de los estudios informáticos.

1.3. Impacto Esperado

Este proyecto se desarrolla con la finalidad de ayudar a los grupos de investigación donde se han invertido recursos en el sistema Kinect para la extracción de esqueletos. Obtener esqueletos más ajustados a la realidad hace más funcional el sistema, reduciendo tanto falsos positivos como falsos negativos en dichas investigaciones. Por tanto, el presente proyecto espera abrir un camino a la mejora de dicha extracción y frenar la necesidad de cambiar los dispositivos de captura.

También se espera que se pueda emplear esta mejora en entornos relacionados con los videojuegos. Con este sistema, se pueden ofrecer interacciones más veraces, mejorando la experiencia del usuario y, por tanto, mejorando la imagen corporativa hacia los jugadores con la posibilidad de futuras compras.

1.4. Metodología

Para poder cumplir los objetivos contemplados, el desarrollo del Trabajo Fin de Grado debe comenzar con un estudio del Estado del Arte. Este estudio tiene como finalidad la búsqueda de información actual y alternativas entre los distintos tipos de redes neuronales y sistemas de extracción de esqueletos. Como resultado de este estudio, y un posterior análisis, se propondrá el sistema corrector. Este sistema se desarrollará de manera que se pueda probar la validez del sistema planteado. Por último se mostraran posibles evoluciones y desarrollos derivados a partir del presente proyecto.

1.5. Estructura

En la sección 2 (Estado del arte) se recorren diversas investigaciones relacionadas con los sistemas de extracción de esqueletos, reconocimiento de imágenes y se presentan las redes neuronales que resultan de mayor interés para el presente proyecto.

A lo largo de la sección 3 (Análisis del problema) se plantean las posibles soluciones a los problemas marcados en los objetivos empleando la información de la sección anterior. A continuación, se propone una solución junto a las razones de esta decisión.

En la sección 4 (Diseño de la solución) se introduce el diseño que debe tener el proyecto empleando las tecnologías mostradas en el estado del arte y las decisiones argumentadas a lo largo de la sección 3.

Le sigue la sección 5 (Desarrollo de la solución propuesta) en el que se detallan aspectos técnicos relativos a la implementación de las diversas secciones que componen el proyecto y las conexiones entre estas.

Una vez alcanzada la sección 6 (Pruebas y resultados) el proyecto se encuentra en condiciones de mostrar los resultados obtenidos para las distintas pruebas. Estos

resultados se acompañan de un análisis para una mayor comprensión del grado de cumplimiento con los objetivos expuestos.

En la sección 7 (Conclusiones) se extraen conclusiones de los resultados obtenidos en la fase de pruebas y se analizan desde el punto de vista de los resultados, si las decisiones tomadas en el planteamiento del sistema fueron correctas. Además, le sigue una argumentación sobre los posibles cambios a realizar en el proyecto al observar si son correctas o no algunas de las decisiones tomadas en el planteamiento del diseño de la solución.

La sección 8 (Trabajos futuros), se presentan trabajos que pueden derivar del presente proyecto y posibles líneas de investigación relacionadas.

Se puede encontrar en la sección 9 (Terminología) un glosario con los términos técnicos que pudieran resultar desconocidos para el lector o términos que son comunes pero se emplean de una manera más específica en este documento.

En la sección 10 (Referencias) se incluye un listado con las referencias a los artículos y otros documentos que se han empleado como base para formar el estado del arte principalmente.

Cerrando el desarrollo del Trabajo Fin de Grado se incluyen unos breves agradecimientos se muestran en la sección 11 (Agradecimientos).

1.6. Convenciones

Las referencias se realizan siguiendo el estándar *Harvard – Anglia*. Este sistema de referencia, cuando referencian a artículos, muestra los siguientes campos y en el orden indicado: Autor o autores, año de publicación, título del artículo, lugar de publicación, tomo, volumen y números de páginas.

Consulte https://libweb.anglia.ac.uk/referencing/harvard.htm para otras referencias o detalles más específicos.

2. Estado del arte

Desde que Kinect apareció en el mercado, surgieron muchos estudios en los que se empleaba el esqueleto de manera relevante en la investigación. Pero los investigadores detectaron que las cámaras Kinect producían muchos errores en los esqueletos extraídos. Esto llevó a realizar pruebas para verificar cuán exactos eran los datos retornados por el sensor.

En el estudio realizado por Obdrzálek (Obdrzálek, et al., 2012) concluye que las articulaciones que conforman el esqueleto suelen tener un error en torno a 10cm respecto a su posición original siempre y cuando el sujeto se encuentre a una distancia superior a dos metros e inferior a 4 metros desde el dispositivo. Otra variante de estudio respecto a la veracidad de los datos la podemos encontrar en el estudio realizado por Kaenchan (Kaenchan, et al., 2013). En este, se muestran evidencias de diferencias comparando los resultados de dos Kinects distintas captando la misma escena desde la misma situación y por tanto, extrayendo el mismo esqueleto. Otro fue el método empleado en el estudio realizado por Aniruddha et al. (Sinha, et al., 2013). No llega a cuantificarse, pero sí que se argumenta sobre los errores introducidos por Kinect en los esqueletos extraídos cuando el sujeto se encuentra en un movimiento cíclico como puede ser el caminar. Por último, el estudio dirigido por Livingston (Livingston, et al., 2012) pone al descubierto las diferencias que surgen cuando hay más de un sujeto en la escena de manera que a mayor cantidad de sujetos, más inexactitud en todos los esqueletos extraídos.

Estos fallos han llevado a los investigadores a estudiar maneras de corregir dichos esqueletos o bien maneras alternativas de extraer esqueletos empleando las imágenes proporcionadas por Kinect. Estas investigaciones se pueden clasificar en tres ramas diferentes donde las dos primeras estarían basadas en la corrección y la última estaría basada en la reextracción:

- Correcciones con filtros temporales
- Correcciones con ajustes físicos
- Nuevas extracciones basadas en imágenes Kinect

Correcciones con filtros temporales

Dentro de la categoría de correcciones mediante filtros temporales se pueden observar dos vertientes. La primera de ellas emplea filtros sobre cada una de las articulaciones empleando como información las posiciones de las articulaciones en *frames* anteriores. La segunda vertiente son aquellos métodos que emplean filtros para cada una de las articulaciones, pero empleando en esta ocasión la posición del resto de articulaciones del mismo esqueleto o mismo *frame*.

En la primera vertiente podemos encontrar el estudio realizado por Shu et Al. (Shu, et al., 2014) expandido por Mochamad et al. (Bachtiar, et al., 2016) donde se

emplean principalmente *filtros de Kalman*[1]. Los resultados obtenidos reducen esqueletos espasmódicos dando como respuesta el suavizado de aplicar un filtrado a lo largo de varios *frames*. Una de las mejoras agregadas a este estudio es la capacidad de realizar una interpolación de las articulaciones no extraídas correctamente empleando las articulaciones disponibles del mismo *frame*. En caso de *frames* consecutivos donde el esqueleto es erróneo, este filtrado no retorna información o no es válida dado que requiere los esqueletos anteriores para poder aplicar el filtrado.

Otra de las modificaciones del estudio anterior, se puede encontrar en el trabajo realizado por Tripathy et al. (Tripathy, et al., 2017) en el que los filtros no usan únicamente información de la articulación a lo largo de varios *frames* sino que además emplean información de articulaciones adyacentes del mismo *frame* por lo que se mantienen de manera consistente las distancias entre las articulaciones que forman el esqueleto.

Correcciones con ajustes físicos

Existen dos estudios principales en este apartado. En el primer estudio, realizado por Handrich et al. (Handrich & Al-Hamadi, 2015), se emplea un modelo 3D del cuerpo humano y se ajusta según el esqueleto extraído por Kinect. De esa manera, las articulaciones mal estimadas por Kinect se aproximan haciendo que encajen con el modelo.

Otro estudio similar en el que se realiza la corrección desde el punto de vista físico del esqueleto, emplea la imagen de profundidad y a su vez la distancia que debería existir entre las distintas articulaciones para corregir el esqueleto acercándolo más al esqueleto que debería haber extraído Kinect (Valcik, et al., 2015).

Nuevas extracciones basadas en imágenes Kinect

La alternativa a emplear filtros se centra en extraer un conjunto de articulaciones del esqueleto empleando para ello la imagen de profundidad, la imagen a color o ambas al mismo tiempo del mismo modo que hace el dispositivo Kinect. El método de extracción de la posición de las articulaciones deseadas se reduce a aplicar como base algún método de reconocimiento de imágenes o reconocimiento de objetos.

En el reconocimiento de imágenes u objetos se pueden encontrar dos vertientes. La primera vertiente es aquella basada en *puntos de interés* o *regiones de interés*. Estos pueden estar basados en reconocimiento de bordes, esquinas, matices o degradados y texturas.

Sobre la primera vertiente, una de las opciones estudiadas es emplear por una parte la imagen RGB para extraer un esqueleto y, por otra parte, se emplea la imagen de profundidad para corregir el esqueleto obtenido anteriormente. (Kar, 2010). En este

[1] Es un algoritmo u operación de corrección empleada para eliminar el ruido entre los datos.

estudio se emplean filtros Haar[2] y con estos resultados se procede a clasificar cada una de las secciones que componen el cuerpo del usuario.

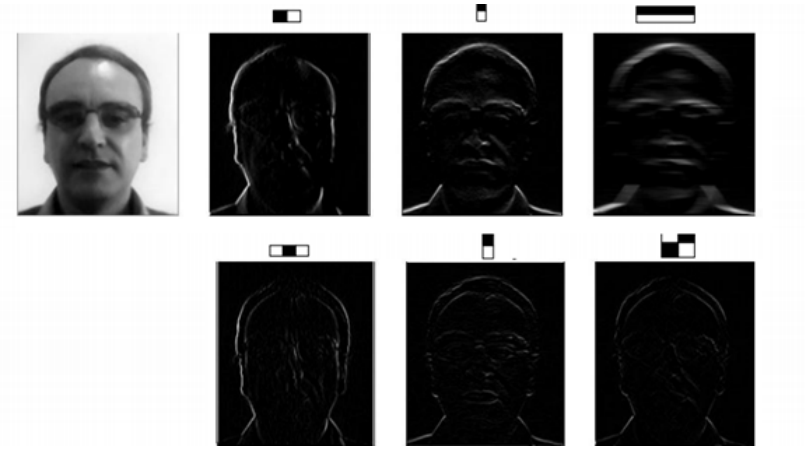

Ilustración 2 – Filtros Haar aplicados a una fotografía. (Davisclick, 2018)

La segunda vertiente, más reciente y con mayor expansión, es la formada por redes neuronales. Las redes neuronales se encuentran dentro del campo de *machine learning*, una de las ramas principales de la inteligencia artificial.

Para comprender la evolución y las diferencias entre diversos estudios sobre redes neuronales, y más tarde ser capaces de razonar posibles soluciones o alternativas al problema inicial, se necesitan explicar algunos conceptos relativos a las redes neuronales dejando de lado aquellos que no sean de especial interés para el presente proyecto.

Aproximándonos al funcionamiento del cerebro, definimos una *red neuronal* como un conjunto de unidades básicas llamadas *neuronas* las cuales se encuentran comunicadas entre sí de manera específica. Estas neuronas aplican un *peso* o ratio a cada entrada, las une en una *función aditiva* y retorna el resultado normalizado mediante una *función de activación* a las neuronas que estén conectadas a la salida (véase la ilustración 3). Este tipo de neurona, en informática, se denomina *perceptrón* (Rosenblatt, 1957).

[2] Algoritmo u operación empleada para detectar diferencias de intensidad entre regiones de la imagen de manera óptima.

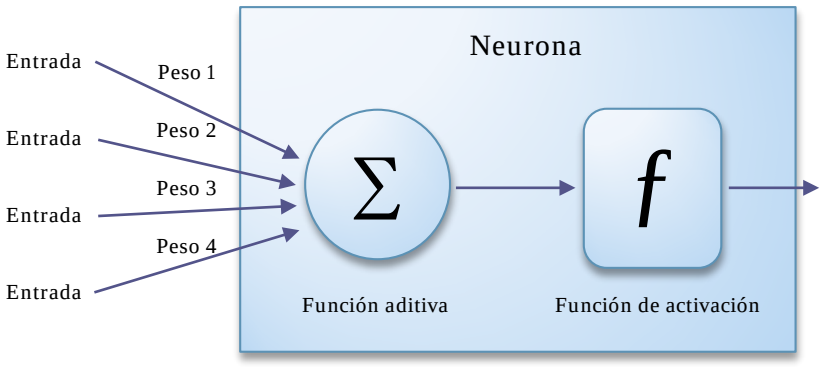

Ilustración 3 – Diagrama de una neurona perceptrón

Las neuronas, empleadas de manera individual, permiten hacer un sistema de discriminación entre dos *clases* siempre y cuando las clases sean linealmente separables. Uno de los primeros problemas era determinar los pesos de cada dato de entrada. Al comienzo, tardaban meses en determinar los pesos ya que realizar un ajuste de estos a mano era una ardua tarea nada trivial conforme aumentaban los datos de entrada. Se desarrolló un algoritmo que ajustaba automáticamente estos pesos. Este algoritmo se denominó *aprendizaje*. El proceso de aprendizaje requiere, al menos, de dos conjuntos de datos llamados *train* y *test*, aunque en sus inicios únicamente se empleaba el de *train*, formado por varios datos de entrada y las clases o etiquetas de estos, es decir, la solución que debe retornar la clasificación. Para una neurona, Rosenblatt diseñó un algoritmo de aprendizaje que ajusta los pesos de entrada según el fallo cometido clasificando el conjunto de test (Rosenblatt, 1961).

Como una neurona solo es capaz de hacer una clasificación binaria, pronto se generó el concepto de capa en redes neuronales. Una *capa* es un conjunto de neuronas no conectadas entre sí, de modo que están todas conectadas con los datos entrantes y cada neurona ofrece como salida un porcentaje de acierto haciendo referencia a si los datos de entrada corresponden a un tipo específico. Estas salidas se unen habitualmente en una función llamada *argmax*, la cual escoge la clase que mayor porcentaje de correspondencia tiene.

Pese a que una capa neuronal es capaz de discriminar entre más de dos clases de datos, las relaciones entre datos de entrada empleadas para realizar la discriminación son básicas. Para solventar este inconveniente, se desarrolló el concepto de *red neuronal multicapa*. Los datos de entrada, *capa de entrada*, se conectan con todas las neuronas de la primera *capa oculta*. Todas las neuronas de esta primera capa se conectan a las de la siguiente capa oculta, siguiendo sucesivamente hasta la última capa, *capa de salida (véase la* ilustración 4*)*. El término capa oculta hace referencia a que los parámetros que ajustan el funcionamiento de dichas neuronas no es visible. El número mínimo de capas ocultas es uno y así, al sumarse a la capa de salida, forman un mínimo de dos capas discriminantes creando relaciones algo más complejas. Las comúnmente denominadas relaciones de segundo nivel, pueden ser imaginadas como aplicar una función G al resultado de sumar las salidas de una función F, donde la función F haría referencia a las neuronas de la capa oculta mientras la función G haría referencia a una neurona de la capa de salida. Este tipo de relaciones entre capas donde

las neuronas solo se comunican con la capa anterior y posterior pero nunca entre neuronas de la misma capa, se denominan capas *fully-connected* (Rosenblatt, 1961).

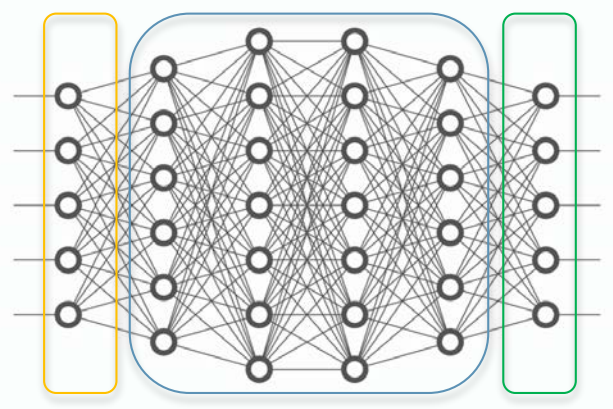

Ilustración 4 – Arquitectura de una red neuronal multicapa. En amarillo la capa de entrada, en azul las capas ocultas y en verde la capa de salida. Imagen modificada (Sena, 2017)

El uso de más de una capa discriminante implica que las características básicas obtenidas en la primera capa oculta pueden relacionarse entre ellas, obteniendo relaciones más complejas. Aun así, estas relaciones más complejas continúan funcionando correctamente solo para problemas de clasificación lineal. Para salvar este problema, se modificaron las funciones de activación a formas no lineales como *sigmoide*, *tangencial hiperbólica* o *ReLU* (*Rectified Linear Unit*) entre otras.

Empleando esta arquitectura, se deben realizar dos distinciones: las *redes perceptrón multicapa* (MLP) y las redes neuronales profundas, más conocidas como Deep Neural Network (DNN). Ambas arquitecturas se encuentran ubicadas en el concepto *Deep Learning* (véase la ilustración 5). Este término referencia todos los algoritmos de aprendizaje que abstraen relaciones o características complejas a partir de los datos de entrada, tratándose muchas veces de relaciones no comprensibles por los humanos o simplemente desconocidas por la dificultad de ser observables.

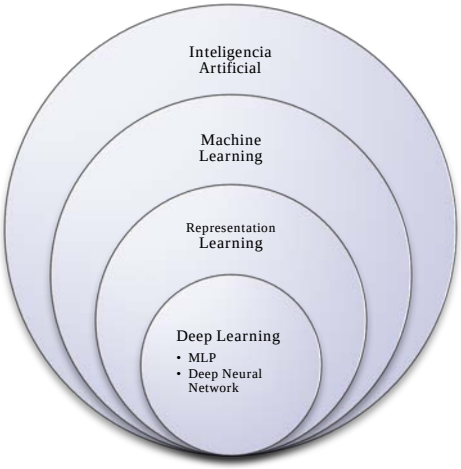

Ilustración 5 – Ubicación de los algoritmos Perceptrón Multicapa y Redes Neuronales Profundas en el campo de la Inteligencia Artificial

Las redes multicapa suelen estar compuestas por entre 2 y hasta 6 capas ocultas, y su funcionamiento es como el descrito anteriormente donde todas las capas se encuentran interconectadas de modo *fully-connected*. El número máximo de capas se encuentra limitado por el número de neuronas en cada capa, la potencia computacional del sistema y la memoria disponible. Un mayor número de neuronas en una capa aumenta el coste computacional exponencialmente. El aumento de número de capas también repercute en un crecimiento exponencial del coste computacional dado que todas las neuronas de una capa están relacionadas con todas las neuronas de la siguiente capa. Además hay que recordar que ambos aumentos computacionales conllevan incremento en la memoria requerida.

A partir de este punto, el método de entrenamiento es similar para todas las redes neuronales ya que todas las redes funcionan como si se tratasen de redes multicapa. Para cada dato incorrectamente clasificado del conjunto de test, el error se divide según los pesos de la última capa de manera que cada neurona de la capa anterior recibe una parte proporcional. Una vez corregida cada neurona de dicha capa, el error se sigue retransmitiendo hacia la capa anterior del mismo modo. El proceso termina cuando se alcanza la capa de entrada. Este proceso se denomina *retropropagación* o *backpropagation* (Bryson, 1961) (véase la ilustración 6).

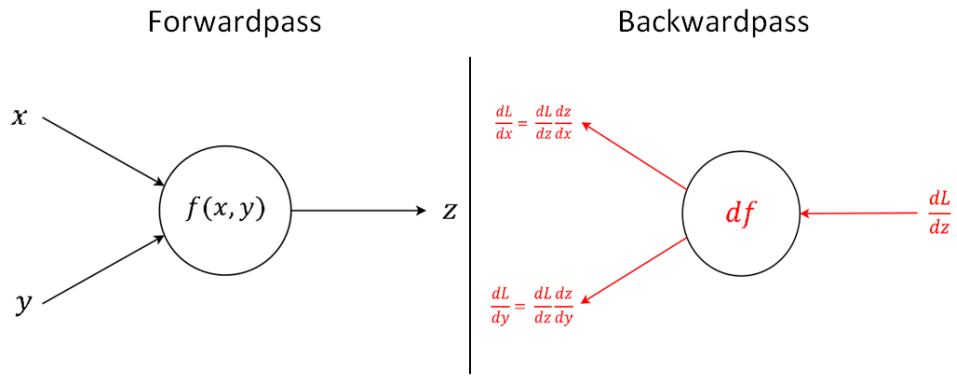

Ilustración 6 – Funcionamiento habitual de una neurona (Izquierda) y su funcionamiento cuando realiza backpropagation. El término L hace referencia al error cometido en la clasificación mientras que las derivadas indican la parte proporcional del error propagado que debe afectar a las neuronas de las capas anteriores. (Jasdeep06, 2017)

Como alternativa a las redes multicapa, aparecieron las redes neuronales profundas. Con la finalidad de reducir los costes computacionales de cada capa, se minimizaron la cantidad de conexiones entre neuronas de manera que cada una de las neuronas de las capas ocultas se encuentran conectadas a unas pocas neuronas de la capa anterior, también llamadas neuronas locales.

Una de las capas que trabajan con neuronas locales más importantes del reconocimiento de imágenes que se pueden encontrar en la actualidad es la denominada *convolucional* ya que su funcionamiento equivale a aplicar una operación convolucional empleando las neuronas locales como neuronas vecinas de una neurona central.

Otra de las medidas empleadas para reducir los costes computacionales fue la adición de capas intermedias entre capas ocultas cuya función es seleccionar qué información es valiosa de un conjunto de neuronas y desechar el resto o bien extraer una media de los datos entre otras posibles operaciones. Algunas de las capas más empleadas y conocidas de este tipo son *max pooling*, *min pooling*, *average pooling* y *dropout*. Esto permite reducir el número de neuronas o conexiones entre cada una de las capas y por ende, aumentar considerablemente la capacidad máxima del número de capas alcanzando características o abstracciones más complejas (véase la ilustración 7).

Por último, al reducir el tamaño de las capas apareció el concepto de *profundidad de capa*. Para un mismo conjunto de neuronas locales, siempre en la misma capa, se ejecutan varias neuronas con varios pesos para las mismas neuronas entrantes, devolviendo más de un resultado para dicho conjunto de neuronas locales. Para comprender mejor esta operación, se puede pensar que para unos mismo datos de entrada se le aplican varias funciones por separado, tantas como la profundidad de capa, y devuelven la solución de cada una de las funciones, sin realizar ninguna combinación de estas soluciones.

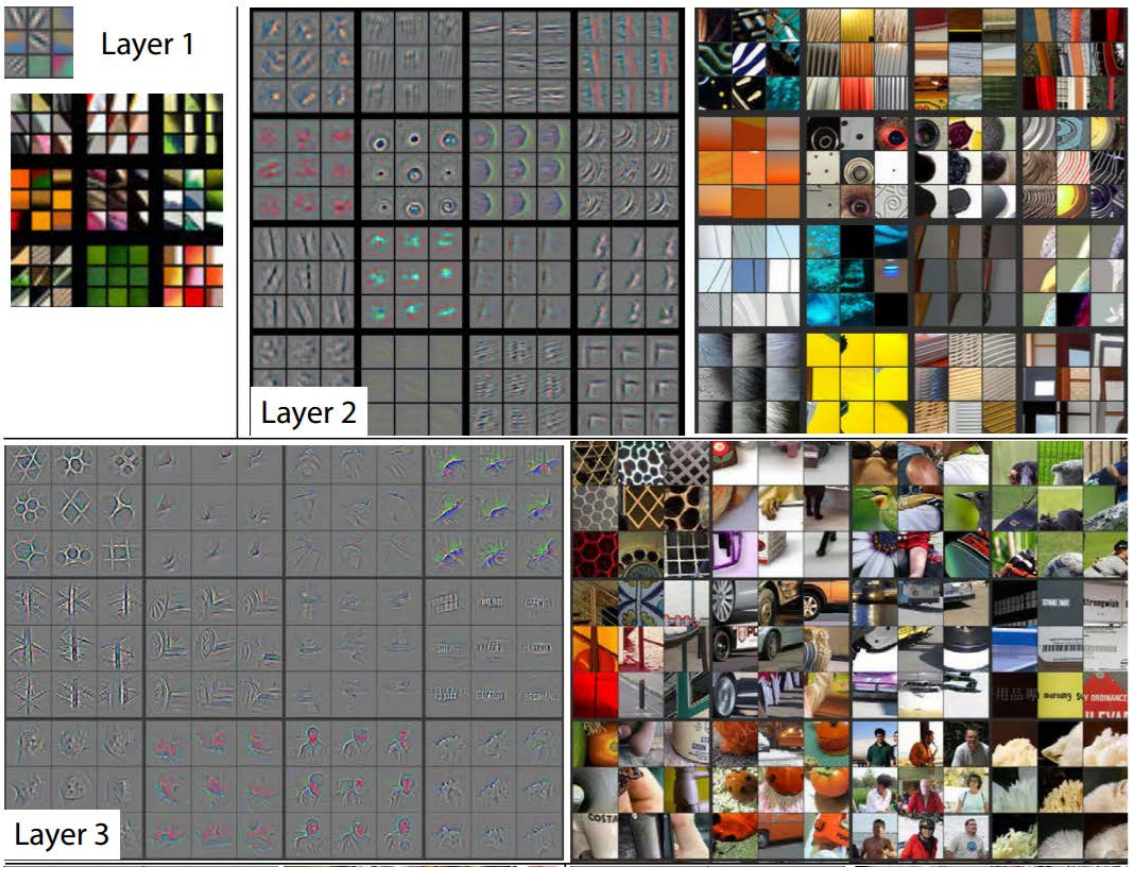

Ilustración 7 – Características extraídas de 3 capas ocultas de una red neuronal convolucional acompañadas de secciones de imágenes que es capaz de clasificar cada neurona (Zeiler & Fergus, 2013)

Dada la gran variación de redes neuronales, específicamente las relativas a redes neuronales profundas, se ha centrado mucho la visión de este campo hacia las redes que pueden resultar de mayor interés. Para un mejor entendimiento de la expansión de las redes neuronales, véase la ilustración 8.

Para el presente estudio, se observan dos divisiones de interés dentro de las redes neuronales profundas. Una de las ramas es la formada por las redes neuronales destinadas a la *clasificación de imágenes* donde solo aparece un objeto, normalmente centrado y puede encontrarse cubierto parcialmente. La otra rama está formada por aquellas redes destinadas al *reconocimiento de objetos* dentro de una imagen.

Las redes orientadas a la clasificación de imágenes íntegras emplean como entrada una imagen donde se muestra principalmente un objeto y retorna una lista de posibles objetos con una probabilidad asignada. A continuación se expondrán las redes neuronales profundas más populares.

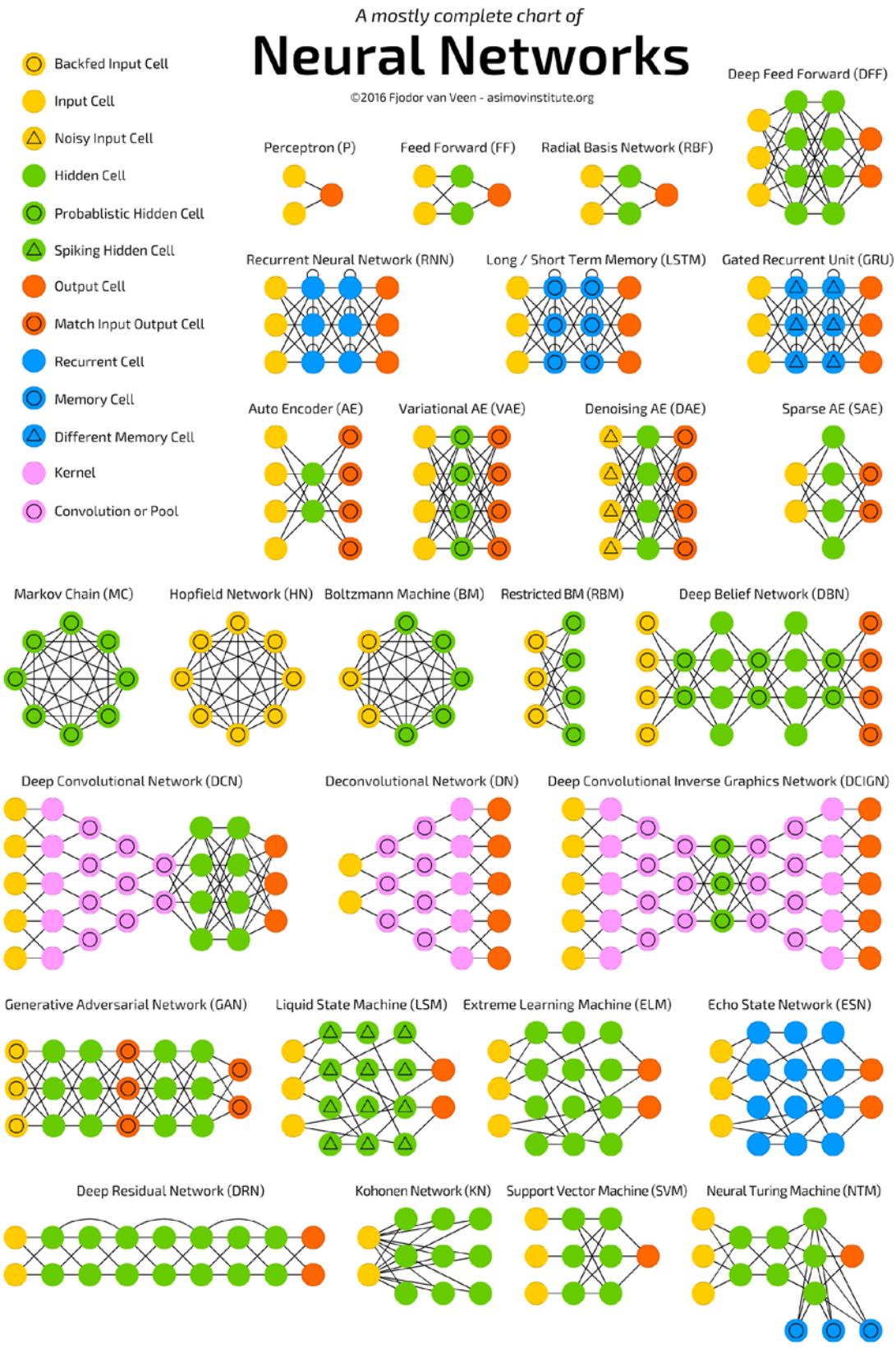

Ilustración 8 – Clasificación de redes neuronales. (Veen, 2016)

Una de las redes pioneras en emplear el deep learning, como se conoce hoy en día, para la clasificación de imágenes fue la VGG16 (Simonyan & Zisserman, 2015). Esta red alterna capas convolucionales con funciones de activación ReLU y capas *max pooling* que reducen la dimensión de la imagen. Las últimas capas son del tipo *fully-connected* para poder extraer conclusiones de las abstracciones complejas que se han obtenido como resultado de las capas anteriores (véase la ilustración 9). El empleo de capas *max pooling* permite centrar el objeto de interés por lo que desecha la información del entorno devolviendo la probabilidad para cada una de las clases del problema. Apareció a su vez la red VGG19 para computadores de altas prestaciones donde se aumenta la profundidad de algunas capas convolucionales, obteniendo significativamente mejores resultados en algunos entornos.

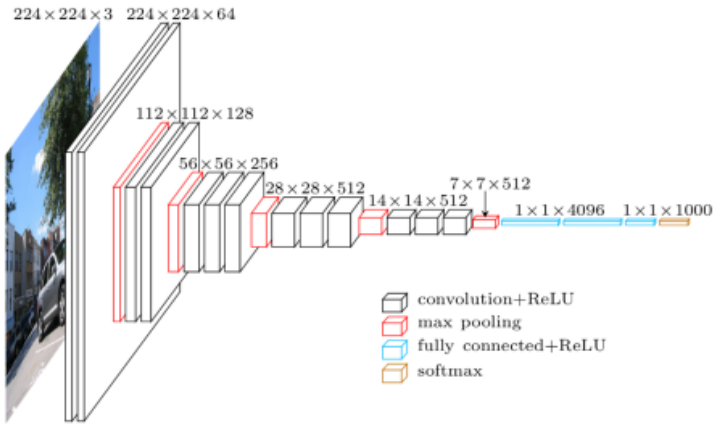

Ilustración 9 – Descripción de la red neuronal VGG16 (Blier, 2016)

La red GoogLeNet (Szegedy, et al., 2015) es una red neuronal que emplea varios módulos dentro de una misma capa. Estos módulos están formados por un *max pooling* y convoluciones de diversos tamaños que se emplean para extraer relaciones del nodo de entrada principal con el entorno de los nodos locales. Los resultados de estos módulos se concatenan como salida de la capa. Esta capa, rebautizado como módulo, recibió el nombre de *Inception*. Existe una variante de este módulo empleado para imágenes de grandes dimensiones que retornan resultados con dimensiones más reducidas, *Inception with dimensionality reduction (véase la* ilustración 10*)*. A diferencia de la red VGG16, en esta ocasión, cada una de las capas tiene en consideración información del entorno por lo que una imagen donde el fondo se detecta como agua ayuda a clasificar que debe tratarse de un animal marino o un animal flotando. Para observar la profundidad de GoogLeNet véase la ilustración 11.

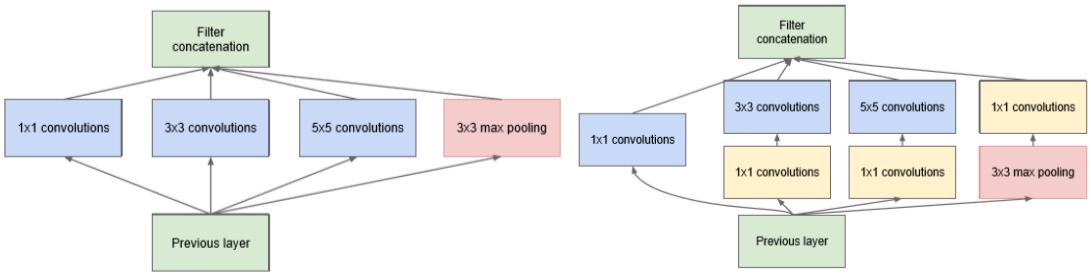

Ilustración 10 - Módulo Inception (izquierda) y módulo Inception with dimensionality reduction (derecha) (Szegedy, et al., 2015)

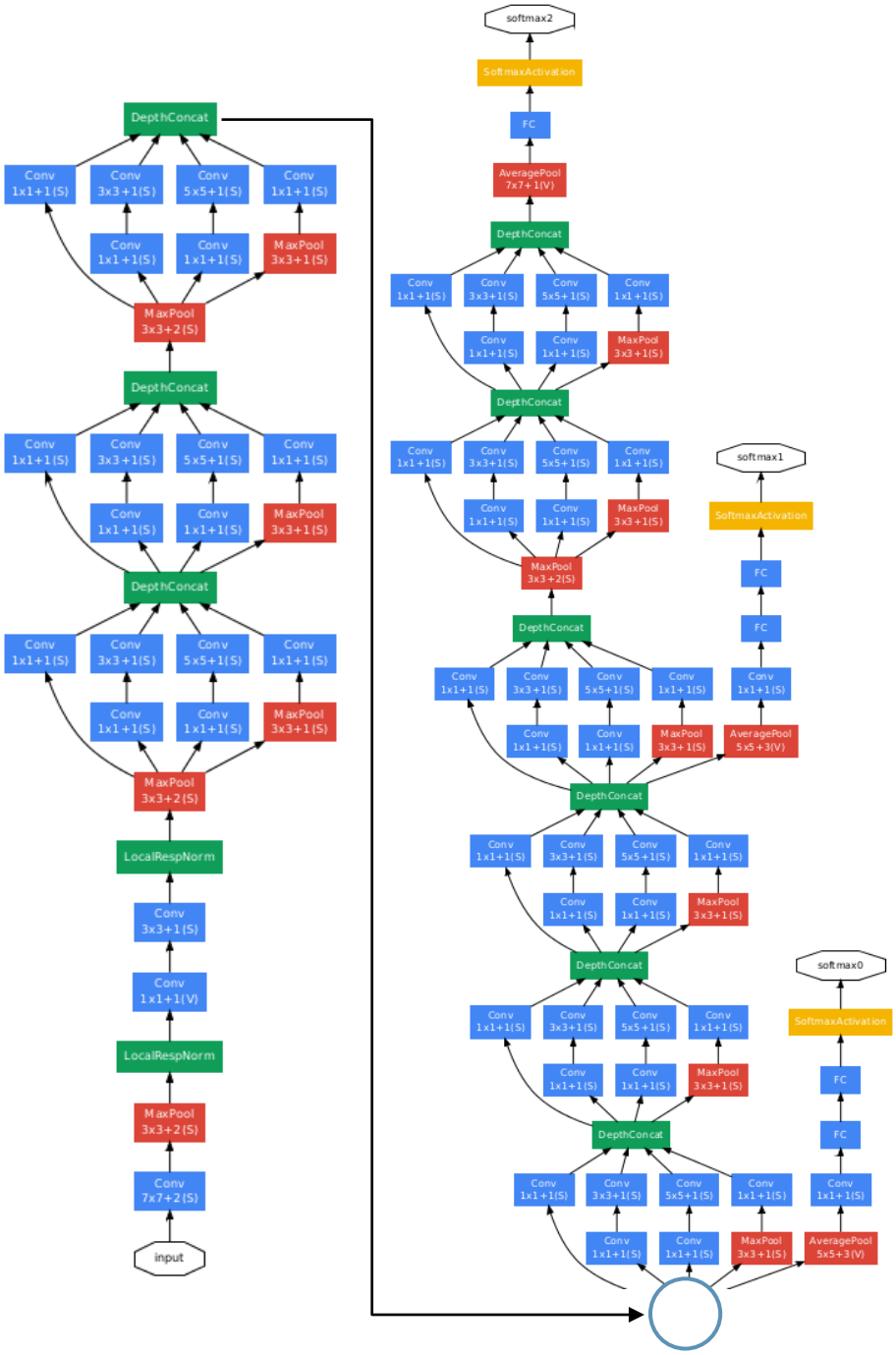

Ilustración 11 – Arquitectura de la red neuronal GoogLeNet (Szegedy, et al., 2015)

Avanzando en la profundidad de las redes neuronales, el estudio realizado por (He, et al., 2016) muestra evidencias que conforme se incrementa el número de capas, la red neuronal desvirtúa los datos de entrada perdiendo referencias a la imagen inicial conforme se extraen características más complejas. Como solución, He propone varias alternativas teniendo siempre como prioridad mantener información residual a lo largo de las capas. Estas alternativas van desde las más sencillas como podría ser agregar la entrada de una capa como salida de otra de las capas posteriores, hasta más complejas como mejorar los resultados obtenidos fusionando la entrada de las capas al resultado de estas (véase la ilustración 12). Las redes que emplean estas modificaciones se denominan *ResNet*.

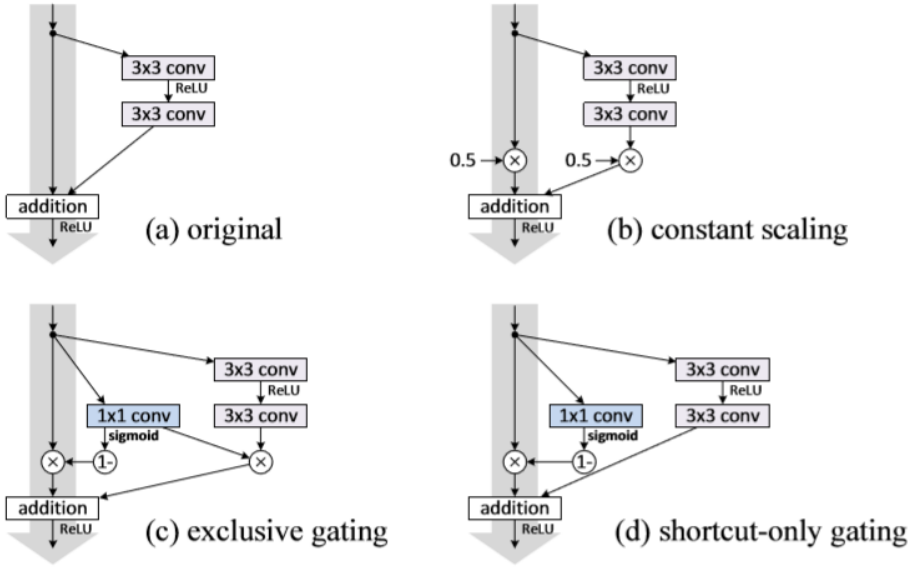

Ilustración 12 – Alternativas empleadas en ResNet (He, et al., 2016)

Como última red neuronal a mencionar dentro de las redes neuronales profundas empleadas en la clasificación de imágenes, podemos encontrar las redes Xception (Chollet, 2017). Empleando como ideas iniciales los módulos *Inception* y las alternativas de ResNet, y con el objetivo de profundizar aún más en la abstracción de características complejas, Chollet crea una red neuronal con profundidad adaptable a las necesidades de cada sistema. Para ello, divide la red neuronal en tres secciones. La primera sección, el flujo de entrada, se encarga de reducir las dimensiones de la imagen de entrada hasta que ocupa alrededor de un 6% con respecto a la imagen original. En la siguiente sección, el flujo intermedio, se calculan relaciones entre características para abstraer relaciones complejas. Esta sección se puede ejecutar recursivamente tanto como se establezca para el sistema. Por último, la sección de flujo final, la red neuronal se encarga de establecer una conexión entre estas características complejas y la clasificación en las distintas clases (véase la ilustración 13). Dada la recursividad del flujo intermedio, esta red neuronal también puede ser incluida en la categoría de redes neuronales recursivas no analizadas en este trabajo dado el poco interés en esta categoría.

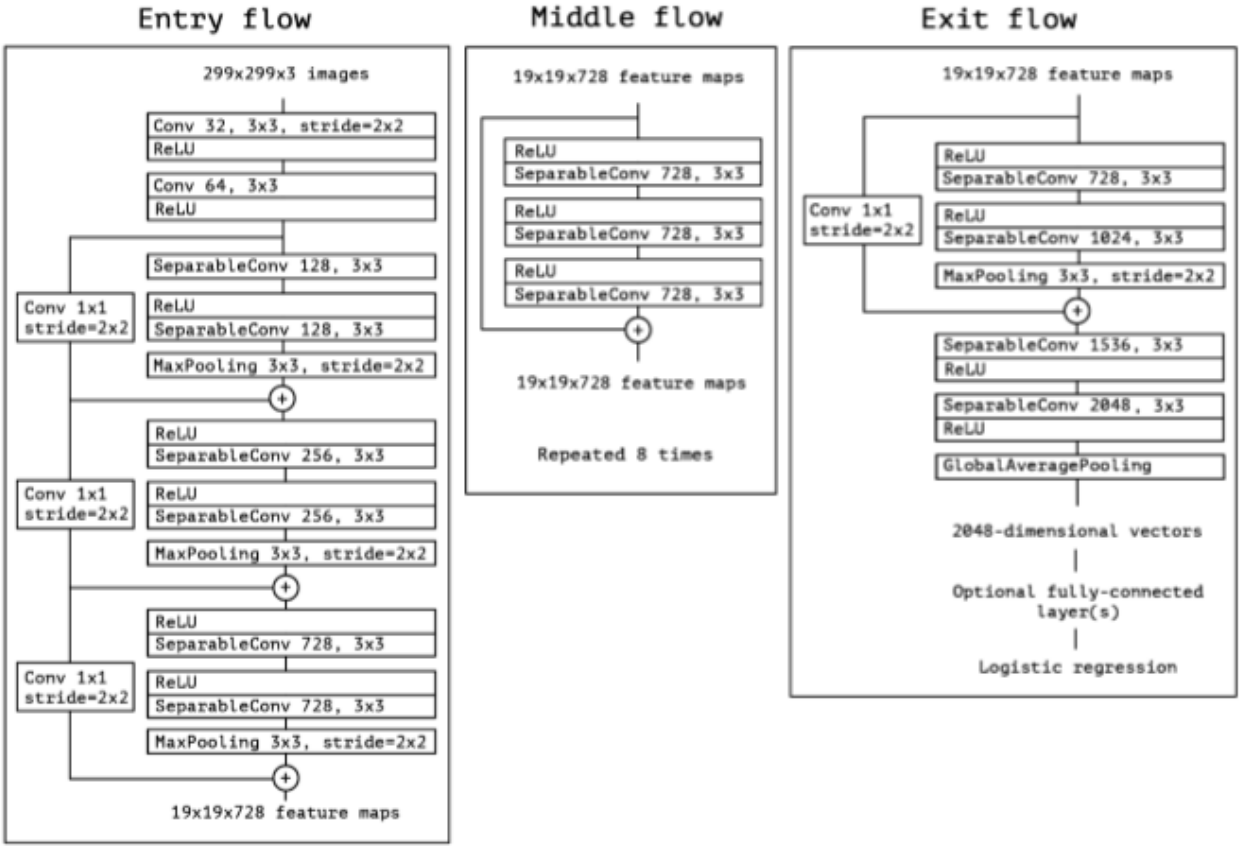

Ilustración 13 – Flujos presentes en la red neuronal Xception (Chollet, 2017)

Las redes neuronales destinadas al reconocimiento de objetos pueden aplicar dos métodos. Las que componen el primer método, están compuestas de dos secciones bien marcadas. La primera sección extrae las posiciones de los posibles objetos, produciendo recortes de la imagen principal. La segunda sección realiza un reconocimiento de imagen como los vistos anteriormente aplicado a cada uno de los recortes.

Para el caso presentado en este proyecto, resulta de interés la segunda clase de métodos. Son aquellos que emplean la imagen completa sin necesidad de realizar recortes de esta. El interés radica en que tanto el recorte como la ejecución repetitiva de partes o toda la red neuronal implica un considerable aumento del coste computacional haciendo que no sean aptos para sistemas ejecutados en tiempo real.

La red neuronal Inside-Outside Net (Bell, et al., 2016) obtiene de la imagen de entrada 2000 secciones y clasifica cada una de estas secciones obteniendo las probabilidades de que un objeto se encuentre presente en esa sección y una *bounding box* del objeto. Esta red neuronal emplea la red neuronal VGG16, levemente modificada, con la imagen completa sin tener en cuenta si se encuentra más de un objeto en la imagen. Una vez ejecutada la red neuronal, se extrae información referente a cada una de las secciones de la imagen, obteniendo dicha información entre los

resultados de las diversas capas. Por último, analizando las probabilidades de cada una de las secciones y se unifican los *bounding boxes* para obtener la segmentación y clasificación de los objetos de la imagen (véase la ilustración 14). Esta red neuronal emplea la información de píxeles adyacentes para una mejor clasificación de las secciones y una obtención más acertada de los *bounding boxes* asociados.

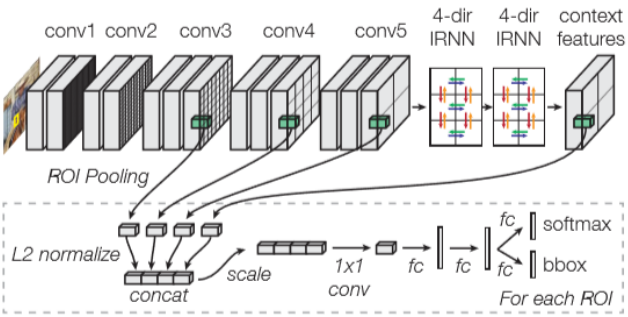

Ilustración 14 – Detalle de la arquitectura de Inside-Outside Net (Bell, et al., 2016)

El algoritmo *You only look once (YOLO)* se presenta como uno de los mejores algoritmos para el reconocimiento de objetos (Redmon, et al., 2016). Su funcionamiento es muy sencillo. Una red neuronal profunda extrae para cada ventana de n x n píxeles, ejecutando una sola vez la red neuronal, las probabilidades de las diversas clases y 5 *bounding boxes* para las 5 clases más probables. Finalmente, estos resultados se fusionan para obtener *bounding boxes* de mayor tamaño siempre que superen una probabilidad mayor al umbral de aceptación o *threshold* (véase la ilustración 15). La red neuronal empleada está inspirada en GoogLeNet. Está compuesta por 24 capas convolucionales y 2 *fully-connected*, acercándose también a la red neuronal descrita por Lin et al. (Lin, et al., 2014) Además, este algoritmo permite modificar la red neuronal por cualquier otra siempre que se ajuste tanto al tamaño de la capa de entrada como la de salida al entorno donde se va a aplicar. Dada la velocidad de ejecución del algoritmo y las pocas operaciones adicionales, este puede ser empleado en aplicaciones en tiempo real. Así lo demuestra Lin en repetidas ocasiones a lo largo de las publicaciones derivadas del artículo inicial, posicionándose como el algoritmo de reconocimiento de objetos más rápido de la comunidad científica.

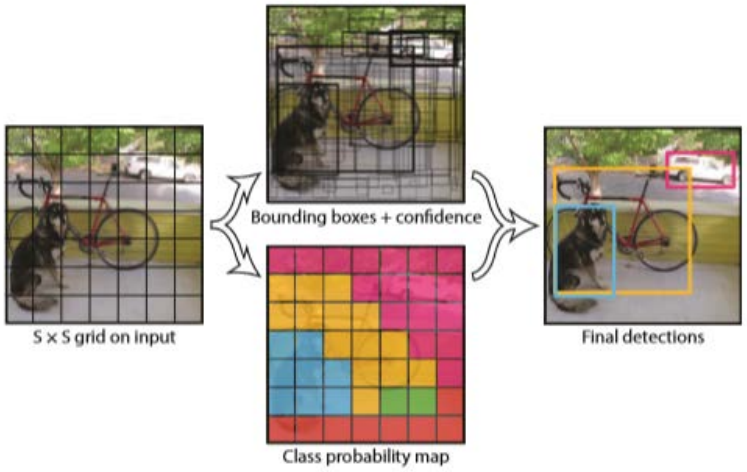

Ilustración 15 – Representación del funcionamiento de YOLO (Redmon, et al., 2016)

Dejando de lado las arquitecturas de las redes neuronales profundas, otra línea de investigación de interés para el presente proyecto es el método de entrenamiento de dichas redes neuronales.

Anteriormente se argumentó que el aumento del tamaño de las capas y del número de neuronas en cada capa, hace que aumente el número de parámetros internos a configurar de manera exponencial. A modo de ejemplo, la red neuronal GoogLeNet contiene aproximadamente 6,8 millones de parámetros (Szegedy, et al., 2015) mientras la red VGG-16 contiene 138 millones (Simonyan & Zisserman, 2015).

Dado que realizar el entrenamiento completo de una red neuronal puede consumir muchos recursos tanto temporales como materiales (computación, memoria y otros) se buscaron alternativas para disminuir dichos costes.

Una de ellas es la llamada *Extreme Learning Machine (ELM)* (Tang, et al., 2016). El funcionamiento de ELM se basa en asignar, una vez diseñada la red neuronal, los parámetros de las neuronas de las capas intermedias de manera aleatoria siguiendo una distribución probabilística, normalmente una distribución gaussiana. De esa manera, solo debemos entrenar la capa final, reduciendo así la cantidad de parámetros a ajustar. Se han realizado estudios al aplicar ELM en las diversas redes neuronales más conocidas mostrando la eficacia de este método para hacer un entrenamiento rápido aunque no definitivo ya que un entrenamiento aplicado a toda la red obtienen ligeramente mejores resultados. También se encuentran estudios referidos a ELM que comprueban cómo deben estar distribuidos probabilísticamente los parámetros ocultos para que converja el aprendizaje en el menor número de iteraciones reduciendo así el tiempo empleado en el aprendizaje.

Otra alternativa para recortar el tiempo de computación empleado para entrenar las redes neuronales es partir de una red neuronal ya entrenada, que sea similar a la red neuronal que deseamos entrenar (Radzi, et al., Enero 2015). A continuación, se añaden las capas deseadas tanto previas como posteriores a la red neuronal original, adaptándola a las necesidades del entorno. Por último, se realiza un

entrenamiento habitual con el método de *backpropagation* limitado a las capas de entrada y salida o bien a la red neuronal completa. Este estudio avala la velocidad de convergencia de este método y el grado de validez en sus resultados.

2.1. Crítica al estado del arte

Como se puede comprobar del análisis al estado del arte, no se han realizado estudios sobre el funcionamiento de redes neuronales destinadas al reconocimiento de objetos empleando como entrada imágenes de profundidad. Por ello, no existe ningún estudio aplicado al entorno del proyecto desarrollado en este trabajo fin de carrera.

Tampoco se encuentran estudios o métodos para verificar si son correctos los esqueletos extraídos de imágenes obtenidas por Kinect a no ser que se basen en mediciones realizadas por otros dispositivos auxiliares.

Además, es habitual no tener en cuenta el tiempo de ejecución en los sistemas de reconocimiento de imágenes por lo que su aplicación a contextos de tiempo real puede verse comprometida si la ejecución del sistema es costosa temporalmente.

2.2. Propuesta

La propuesta del presente estudio, tras analizar el estado del arte, es aunar las ramas de la detección de esqueletos junto a la de detección de imágenes empleando redes neuronales.

Dado que el sistema Kinect realiza la extracción del esqueleto empleando únicamente la imagen de profundidad, para no producir un retraso en nuestros resultados será necesario emplear también únicamente la imagen de profundidad.

Se aconseja el empleo de YOLO dada las necesidades de ejecución en tiempo real. La red neuronal empleada internamente por YOLO, dada la velocidad de ejecución que requiere el sistema, es recomendable que sea una red neuronal GoogLeNet adaptada al problema del presente proyecto.

Además, será necesario emplear la técnica de reentrenamiento ya que el coste temporal y de recursos que necesita un entrenamiento completo sería demasiado extenso para poder abarcarlo en la duración de este proyecto.

3. Análisis del problema

Tal y como se indicó en el apartado 1.1, la extracción de esqueletos del dispositivo Kinect produce una serie de errores que pueden ser clasificados en cuatro grupos: El problema del fantasma, el esqueleto espasmódico, los gestos imposibles y el fallo en el reconocimiento del lateral. A continuación, se analiza cada uno de estos problemas en profundidad.

El problema del fantasma

Ante un escenario, en el que no se encuentra ningún sujeto, el sistema Kinect retorna información sobre un esqueleto. Este esqueleto está formado por articulaciones con posiciones aleatorias (véase la ilustración 16) con tamaños relativos entre las articulaciones muy dispares, como puede ser un antebrazo de pocos centímetros y el antebrazo contrario puede alcanzar el metro de distancia. A simple vista se reconoce este fallo dada la arbitrariedad de las articulaciones o al comprobar que en la escena no hay ningún sujeto.

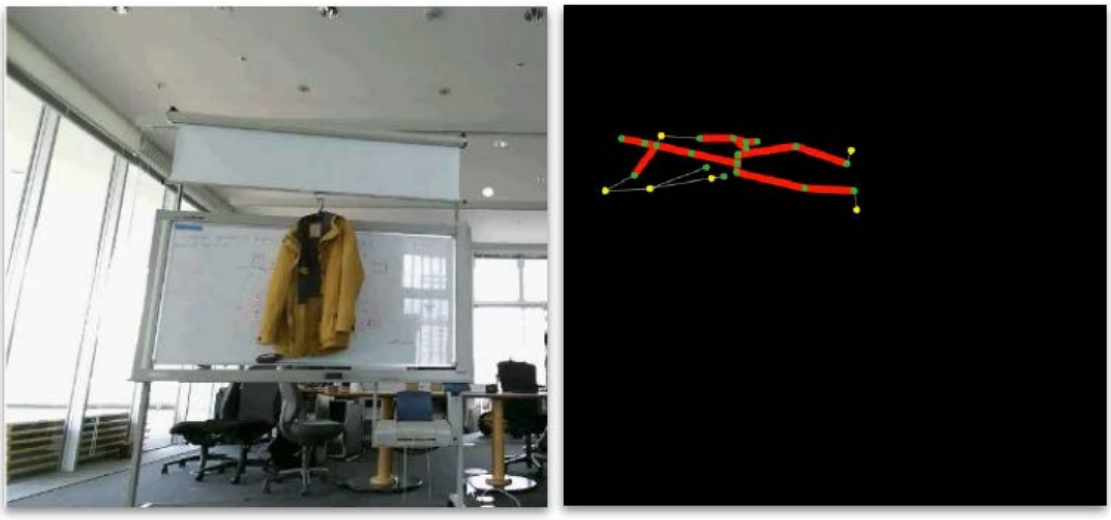

Ilustración 16 - Imagen izquierda, imagen RGB captada por la cámara Kinect. Imagen derecha, esqueleto retornado por la cámara Kinect en el mismo escenario.

Esqueleto espasmódico

Entre *frames* consecutivos, las articulaciones varían radicalmente su posición. En un *frame* posterior vuelve a la posición que le corresponde. Si se visualiza el esqueleto en movimiento, es como si las articulaciones dieran brincos erráticos. Este problema se debe principalmente a que Kinect no tiene en cuenta la posición del esqueleto del último *frame* procesado y por tanto, no dispone de referencias en las que basarse en caso de baja confianza (véase la ilustración 17).

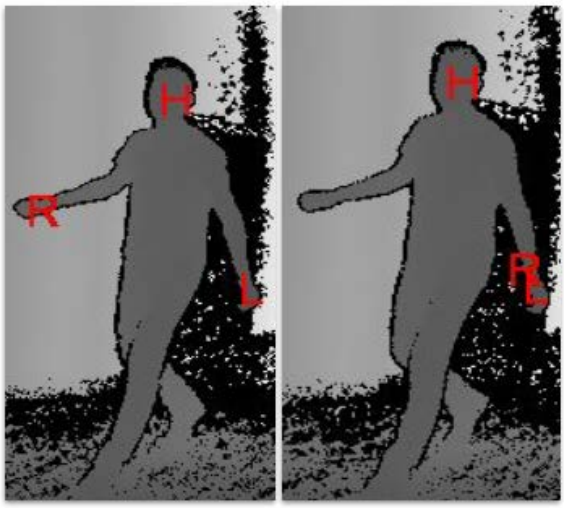

Ilustración 17 - Articulaciones obtenidas en dos frames consecutivos sobre la imagen de profundidad donde se observa como la mano derecha cambia repentinamente de posición. H indica la cabeza, R indica la mano derecha mientras L indica la mano Izquierda.

Gestos imposibles

En ocasiones, Kinect devuelve esqueletos bien formados en cuanto a distancias entre articulaciones pero que físicamente la disposición de las articulaciones no podrían ser reales. Un brazo atravesando el torso o una rodilla doblada hacia el frente puede ser una de las posiciones que Kinect encuentra como válidas. Kinect solo tiene conocimiento de qué articulaciones existen y cómo están conectadas entre sí, pero no comprende que la conexión entre articulaciones tiene un volumen y que, por tanto, pueden estar traspasando otra sección del cuerpo, ni tiene conocimiento sobre los ángulos posibles que puede tomar el esqueleto (véase la ilustración 18). Estos gestos imposibles no se solucionan a lo largo de varios *frames* si el sujeto no se mueve. En otras ocasiones, el movimiento del sujeto tampoco soluciona este problema ya que puede estar producidos por otros objetos que se encuentran en la escena y que reconoce como parte del cuerpo del sujeto.

Ilustración 18 - Articulaciones obtenidas sobre la imagen de profundidad en la que se puede observar cómo la mano derecha del esqueleto se encuentra atravesando el brazo izquierdo. H indica la cabeza, R indica la mano derecha mientras L indica la mano Izquierda.

Fallo en el reconocimiento del lateral

En ocasiones, Kinect identifica alguna de las articulaciones como su opuesta y viceversa. Por ejemplo, el sistema puede devolver un esqueleto en el que la rodilla izquierda esté identificada como rodilla derecha y la rodilla derecha esté señalada como izquierda. Este fallo ocurre habitualmente cuando el sujeto se encuentra en movimiento y, en algunos *frames*, Kinect no es capaz de observar o localizar la articulación en su sensor de profundidad. También puede ocurrir con el sujeto en posición estacionaria (véase la ilustración 19).

Ilustración 19 - Articulaciones obtenidas sobre la imagen de profundidad donde la mano izquierda y derecha se han confundido entre sí. H indica la cabeza, R indica la mano derecha mientras L indica la mano Izquierda.

Tras exponer los problemas que presenta el sistema Kinect se necesita identificar y analizar las posibles soluciones a estos problemas.

3.1. Identificación y análisis de soluciones posibles

Tal y como se ha podido observar en la sección 2, desde el punto de vista de mejorar el esqueleto retornado por el dispositivo Kinect, se puede simplificar la búsqueda a dos grandes grupos. El primer grupo sería aquel formado por soluciones que implique emplear alguna clase de filtro, ya sea filtrando el esqueleto en cada *frame* o bien filtrando cada una de las articulaciones empleando información de *frames* anteriores. El segundo grupo estaría orientado a la extracción de un esqueleto empleando únicamente las imágenes obtenidas por Kinect, o bien generar un esqueleto nuevo realizando una combinación entre el esqueleto extraído por Kinect y otro extraído por otros medios.

Expandiendo este segundo grupo, se debe tener en cuenta los distintos procesos de extracción de esqueletos a partir de imágenes. Estos procesos pueden emplear únicamente imágenes a color, imágenes de profundidad o bien, emplear ambos tipos de

imágenes. Como Kinect emplea únicamente imágenes de profundidad para extraer el esqueleto y el presente trabajo busca poder ejecutarse en tiempo real, sin retardos, la búsqueda se centrará en el grupo que solo emplea imágenes de profundidad.

Para la extracción de un esqueleto empleando imágenes obtenidas por Kinect se pueden emplear los algoritmos explicados en el estado del arte aunque este proyecto busca plantear variantes de estos sistemas.

Uno de estos sistemas posibles sería el relativo a extraer el fondo de la imagen, encontrar los puntos más distantes dentro del conjunto de puntos que forman el cuerpo, y empleando reconocimiento de formas, clasificar cada uno de esos puntos con sus articulaciones correspondientes. El resto de las articulaciones del esqueleto se pueden interpolar con las articulaciones localizadas anteriormente, encajando un modelo 3D del cuerpo humano o bien empleando información de esqueletos extraídos en los *frames* anteriores.

Otro posible sistema de extracción de esqueletos que puede modificarse para emplear imágenes proporcionadas por Kinect puede ser el presentado por Wang et al. (Wang, et al., 2016). En este estudio se transforma la superficie creada por el cuerpo en un plano geodésico de manera que todos los puntos que conforman la superficie corporal ahora están situados en un plano 2D y cada punto tiene asociada una altura. Se calculan los puntos más equidistantes geodésicamente y con ello se obtienen varios puntos, similar a los puntos de interés en una imagen 2D. Estos puntos pueden ser utilizados para relacionar dos superficies corporales de dos frames distintos. La relación permite ajustar los esqueletos que no han podido ser extraídos correctamente.

Un método de extracción del esqueleto un poco más complejo puede emplear directamente algún algoritmo de reconocimiento de imágenes. Reconocimientos dónde se utilizan cascadas de filtros Haar (Nishimura & Kuroda, 2010) se han empleado para extraer puntos característicos y con ellos, realizando interpolaciones, se puede hallar el resto de articulaciones del esqueleto.

Por último, se puede realizar una modificación de algún algoritmo de reconocimiento de objetos para extraer la posición de ciertas articulaciones clave y con ello, empleando los datos extraídos por el dispositivo Kinect, formar un esqueleto final.

3.2. Solución propuesta

La proposición para este proyecto es emplear el algoritmo YOLO, realizando las modificaciones pertinentes, para extraer la posición de las articulaciones clave. En un paso posterior, empleando la información extraída por YOLO, el esqueleto extraído por Kinect y el resultado o resultados anteriores, determinar la posición de cada una de las articulaciones.

3.3. Plan de Trabajo

El trabajo se realiza durante una investigación en *Honda Research Institute Japan*. El tiempo total de investigación es de ocho meses. Como se han invertido tres

meses en recopilar, leer, y organizar la información necesaria para comprender el estado del arte, el tiempo restante para desarrollar el proyecto, realizar las pruebas y finalizar el presente documento es de cinco meses. Por tanto, se estima una duración de un mes para obtener los conjuntos de imágenes empleadas tanto para el entrenamiento de la red neuronal como en las pruebas finales y etiquetarlas. Se estiman dos meses para el desarrollo del proyecto y el entrenamiento de la red neuronal. Medio mes para la ejecución y análisis de las pruebas. Durante el mes y medio restante se debe finalizar el actual documento además de realizar y corregir la presentación solicitada por *Honda*.

4. Diseño de la solución

4.1. Arquitectura del Sistema

El sistema se encuentra formado por dos módulos a implementar que no se ejecutan paralelamente ya que como se verá más adelante, el segundo módulo requiere el resultado del primero.

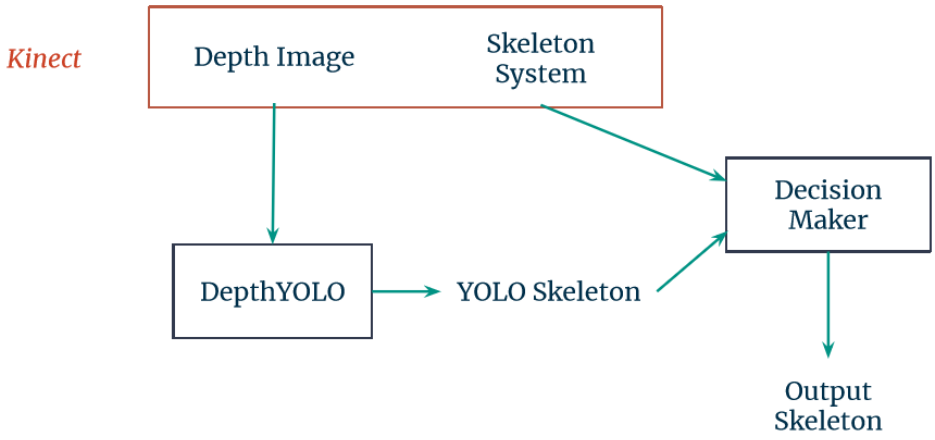

Ilustración 20 – Arquitectura de la solución. Se puede observar el flujo entre la cámara Kinect, el módulo DepthYOLO y el módulo Decision Maker

El primer módulo es el encargado de extraer, partiendo de la imagen de profundidad obtenida por el dispositivo Kinect, la posición de diversas articulaciones formando un pseudoesqueleto. Este módulo recibe el nombre *DepthYOLO*. Como resultado se produce un esqueleto al que llamaremos *YOLO Skeleton*.

El segundo módulo, llamado *Decision Maker*, es el encargado de tomar el esqueleto extraído por Kinect y fusionarlo con el *YOLO Skeleton*. Como resultado, elabora el esqueleto final del sistema.

4.2. Diseño Detallado

En este apartado se mostrará la implementación de cada uno de los dos módulos en mayor profundidad.

Módulo DepthYOLO

Partiendo de la implementación DarkNet del algoritmo YOLO se realizan las modificaciones necesarias para adaptarlas al nuevo entorno (Redmon, 2016).

La primera modificación a realizar es la interfaz de entrada del algoritmo YOLO. Esta implementación de YOLO está preparada para admitir imágenes con tres capas, una capa para cada color, y con valores comprendidos entre 0 y 255. En el presente caso, se necesita que acepte imágenes con una sola capa, el valor de la profundidad, con un rango de valores desde 0 hasta 8000. Además, el tamaño de las imágenes de entrada pasa de 416 x 416 píxeles a 512 x 424 píxeles.

La segunda modificación hace referencia a modificar y reentrenar la red neuronal que forma parte del núcleo de YOLO, más exactamente, la red neuronal *Darknet-19* que está basada en la red *GoogLeNet*. Esta modificación, al igual que el resto de modificaciones, se explica en mayor detalle en el apartado 5.

En cuanto a la última modificación, se necesita ajustar el umbral de aceptación que emplea YOLO en su última fase de manera que elimine las *bounding boxes* cuya probabilidad sea menor al *thresholding* o, en caso de haber más de un *bounding box* para un mismo objeto, elimine los que menor probabilidad tengan ya que en el esqueleto solo puede aparecer una vez cada parte del cuerpo (véase la ilustración 21).

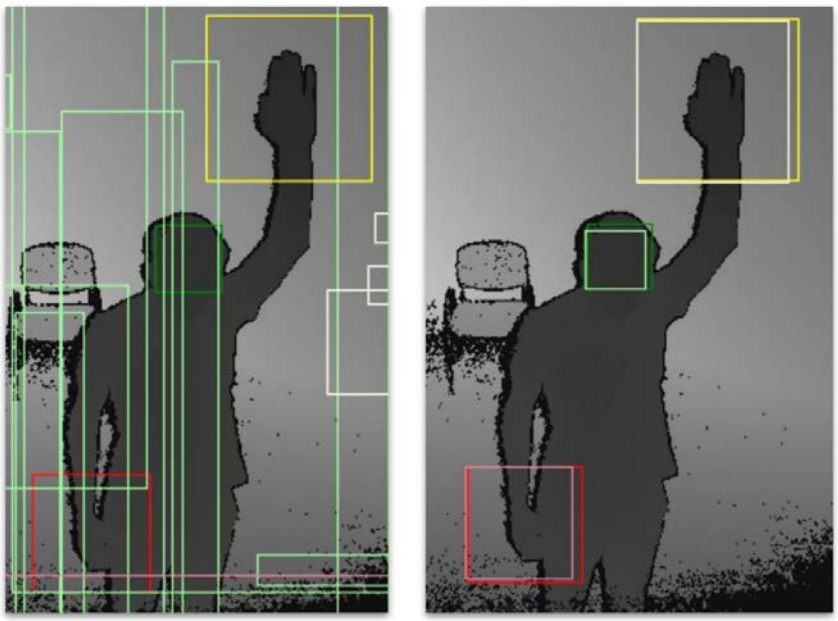

Ilustración 21 – Umbral antes de reajustar (izquierda) y después del ajuste (derecha)

Este módulo conforma el esqueleto al que se ha denominado *YOLO Skeleton*. Este esqueleto, junto al esqueleto obtenido por Kinect, formarán los datos de entrada al siguiente módulo, *Decision Maker*.

Módulo Decision Maker

Este módulo será el encargado de generar el esqueleto mejorado, es decir, la salida final del sistema. Para ello, analiza las posiciones actuales de las articulaciones

devueltas por Kinect y YOLO, y las posiciones anteriores de las articulaciones de los esqueletos que ha ido retornado este módulo en *frames* anteriores. Apoyándose en todos estos datos, es el encargado de corregir los problemas presentados por la extracción de esqueletos por Kinect o bien de desechar el esqueleto.

4.3. Tecnología Utilizada

Referente al hardware, se ha empleado un único ordenador equipado con un procesador Intel® Core™ i7-8700, dos tarjetas gráficas NVIDIA TITAN X y 8GB de memoria RAM. Además, dispone también de un dispositivo Kinect V2. Para la captura de los conjuntos de entrenamiento y pruebas, se empleó una sala equipada por 6 dispositivos Kinect para que se permitiera obtener el mejor conjunto de datos de cada una de las grabaciones. Todo, cortesía de *Honda Research Institute Japan*.

Por otra parte, en cuanto al software, el sistema incorpora el sistema operativo Windows 10. Se emplea C# para poder utilizar las librerías oficiales de Kinect distribuidas por Microsoft. Como la implementación de YOLO está basada en C++, tanto para el desarrollo del módulo *DepthYOLO* como *Decision Maker* se emplea igualmente este lenguaje.

5. Desarrollo de la solución propuesta

En el desarrollo de este proyecto, el único código externo que se emplea como base para las modificaciones es la implementación de YOLO, en concreto la versión 2, realizada por Joseph Redmon bajo el nombre de DarkNet (Redmon, 2016). Este sistema, para implementar la red neuronal, emplea la librería nVIDIA CUDA 9.1 Toolkit para C++ (nVIDIA, 2018). Además, para la adquisición de imágenes se emplea la librería Kinect for Windows SDK 2.0 desarrollada por Microsoft (Microsoft, 2018).

Dadas las limitaciones temporales de este proyecto, se ha reducido el problema al ajuste de las siguientes articulaciones del esqueleto: cabeza, mano izquierda y mano derecha.

Entorno laboral

El proyecto se ha desarrollado íntegramente en los laboratorios de *Honda Research Institute – Japan (HRI-JP)* (Honda, 2018). Este instituto, creado por la empresa de automóviles Honda, se encuentra en el campus ubicado en Wakoshi, provincia de Saitama, Japón. Según la clasificación hecha por Honda, el instituto se centra en estudios sobre ciencias avanzadas, aunando diversos campos como puede ser la inteligencia artificial, la psicología o incluso la biología.

El grupo dirigido por el Dr. Randy Gomez cambiaba habitualmente de tamaño a causa de los distintos contratos. Estaba compuesto de hasta 10 integrantes de diversas nacionalidades. Cada integrante tenía asignado un tema único y concreto. Aunque la información no era expuesta a otros trabajadores del instituto, sí se hacían reuniones entre los integrantes del laboratorio donde se compartían conocimientos y puntos de vista. Por tanto, el presente proyecto fue realizado en solitario.

Adquisición de datos por Kinect

Antes de describir el desarrollo de los módulos *DepthYOLO* y *Decision Maker*, se necesita describir de qué manera se adquieren los datos por Kinect.

El API de Microsoft para Kinect se encuentra compilado en C# mientras que los módulos *DepthYOLO* y *Decision Maker* están implementados en C++ dado que la implementación de la que deriva *DepthYOLO* se encuentra en este lenguaje.

Para realizar la comunicación, fue necesario desarrollar un programa en C# que por una parte gestionase la entrada de datos por parte de Kinect y por el extremo opuesto, se comunicara con el módulo *DepthYOLO*. Esta conexión se efectúa mediante *sockets* que envían los píxeles de las imágenes ordenados en un array de tipo *unsigned short* además de una señal temporal para identificar la imagen con el esqueleto obtenido por Kinect. El esqueleto también se envía ordenado en un array de tipo *float* donde el orden en el que se envían las articulaciones está prestablecido. Un esqueleto tendrá la misma marca temporal que la imagen de profundidad obtenida anteriormente

ya que Kinect primero devuelve una imagen de profundidad y a continuación retorna el esqueleto extraído de dicha imagen.

Para los conjuntos de imágenes de entrenamiento y de pruebas, se realizaron las grabaciones en una sala con 6 Kinects orientadas al centro de la misma. Tras recoger los vídeos, se seleccionó el video con mayor información sobre el esqueleto para cada una de las grabaciones.

Módulo DepthYOLO

El módulo de *DepthYOLO* es, como se describía en el apartado anterior, una modificación de la implementación que *DarkNet* realizó de YOLO. Esta implementación está realizada en C++.

También como se ha descrito anteriormente este módulo recibe tanto las imágenes de profundidad como los esqueletos extraídos por Kinect. Los datos sobre el esqueleto se trasladan directamente al módulo *Decision Maker*, por eso en la ilustración 20 el esqueleto extraído por Kinect se muestra enlazado al módulo *Decision Maker* y no al módulo *DepthYOLO*. En cuanto a la imagen de profundidad, los datos se normalizan y la imagen se escala manteniendo la proporción de la imagen, tal y como sugiere el investigador que desarrolló YOLO. Por lo tanto la imagen pasa de tener un tamaño de 512 x 424 píxeles a 416 x 345 píxeles. Como la imagen de entrada al algoritmo debe tener una relación de aspecto 1:1 y la imagen devuelta por Kinect tiene una relación de aspecto de 64:53 se agregan datos al valor máximo en una franja superior y otra inferior haciendo encajar la imagen de Kinect con la entrada requerida por el algoritmo.

Una vez alcanzado este punto, se necesita realizar los ajustes en YOLO tal y como se argumentó en el anterior apartado. El primer paso es modificar la arquitectura de la red neuronal, adaptándola a las necesidades del presente proyecto. Esto implica agregar una nueva capa al inicio de la red neuronal y remplazar la capa del final de esta.

Se agrega la nueva capa de entrada de tamaño 416 x 416 píxeles de una capa de profundidad y se relaciona de manera *convolucional* 1x1x3 con la que era la capa de entrada, que se mantiene de tamaño 416 x 416 píxeles de tres capas de profundidad. Esta pasa a ser la primera capa oculta.

En cuanto a la adaptación de la salida, la red neuronal original termina con una capa *softmax* con 1000 nodos. Esta capa se cambia por dos capas. La primera de ellas, la nueva y última capa oculta, está formada por una capa *fully-connected* de 3 nodos. Por último, la nueva capa de salida consiste en una capa *softmax de* 3 nodos. En el caso de desear aplicarlo a todo el conjunto del esqueleto, en vez de ser 3 la cantidad de nodos en las últimas capas, sería igual al número de articulaciones que deseamos.

Tras adaptar la arquitectura de la red neuronal, se necesita reentrenarla para su nuevo propósito.

Para reentrenar la red neuronal, se necesitaron dos conjuntos de datos, *train* y *test*. Estos dos conjuntos deben estar correctamente etiquetados por lo que se trabajó en una preparación de estos. Se realizaron 10 grabaciones y se extrajeron mediante

Kinect los esqueletos asociados a las grabaciones. Tras verificar que Kinect había etiquetado correctamente los esqueletos, y desechando aquellos esqueletos que no eran correctos, se obtuvieron aproximadamente 300 *frames* etiquetados de cada vídeo.

En el proceso de reentrenamiento, se suministraba cada uno de los *frames* del conjunto de *train* a la red neuronal. Los datos de salida eran comparados con los datos de los esqueletos extraídos por Kinect que habían sido verificados como correctos. Con ello se calcula el error cometido por la red neuronal. Dicho error se devuelve a la red neuronal para que realice el *backpropagation* y así ajuste los parámetros internos (véase la ilustración 22).

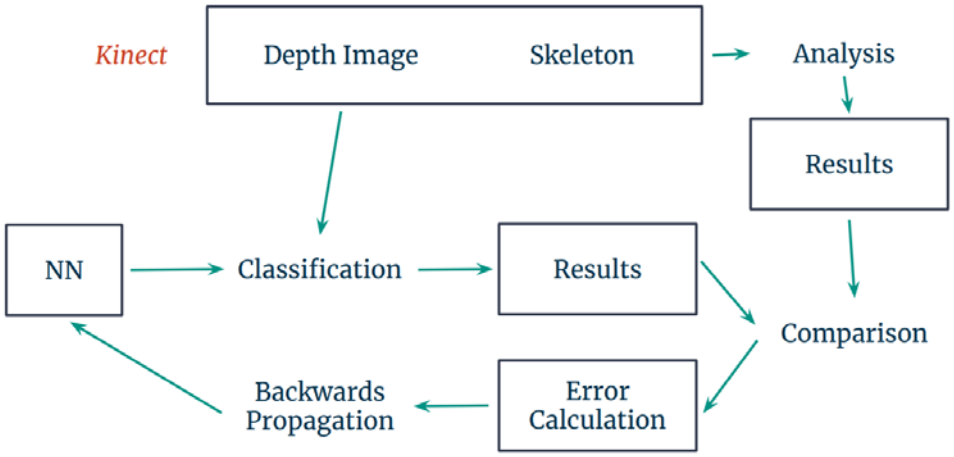

Ilustración 22 – Proceso de reentrenamiento de la red neuronal modificada DepthYOLO

Cada vez que se ha entrenado con todas las imágenes del conjunto *train*, se salvan los parámetros internos de la red neuronal y se evalúa la red neuronal con el conjunto de datos de *test*. El entrenamiento se detiene cuando el resultado con el conjunto de *train* sigue mejorando pero el conjunto de *test* comienza a empeorar. Esto significa que se ha alcanzado *overfitting* o sobreajuste, es decir, la red neuronal ha pasado de reconocer características generales a reconocer exclusivamente dichas imágenes, de manera que ya no es capaz de reconocer las mismas características en imágenes diferentes de la misma etiqueta. Por ejemplo, si entrenamos con imágenes de gatos y el sistema llega al *overfitting*, la red neuronal solo reconocerá las imágenes de gatos si son las mismas que tuvo como entrenamiento. En caso de mostrar otro gato no sería reconocido correctamente. En caso contrario, si la red neuronal no llega al *overfitting*, la red neurona ha aprendido a reconocer aspectos o características que tienen las imágenes de gatos como puede ser la disposición de los ojos o la existencia de bigotes. Como para la detención del entrenamiento debemos llegar al *overfitting*, necesitaremos volver a los parámetros que tenía la red neuronal antes de llegar a este estado. Esa es la razón por la que se han ido almacenando los parámetros de la red neuronal y gracias a esto, se puede retroceder al estado anterior a llegar a *overfitting*.

Una vez realizado el reentrenamiento solo queda ajustar el *threshold* o umbral de aceptación. Se bloquean los parámetros internos de la red neuronal y aplicando el mismo conjunto de *train* y *test*, se realiza un aprendizaje del *threshold* que se ajuste al entorno del problema. Una vez más, se procede a realizar el ajuste mientras tanto el conjunto de *test* como el de *train* disminuyen en error. Cuando el error del conjunto de *test* comienza a incrementar mientras el de *train* se mantiene o sigue reduciéndose, podemos deducir que está generando *overfitting* por lo que el proceso de aprendizaje se detiene.

Módulo Decision Maker

El módulo decidirá las posiciones de cada articulación del esqueleto valiéndose de los esqueletos de Kinect y *DepthYOLO*, y el buffer de resultados anteriores.

Antes de describir el desarrollo del módulo, se necesita introducir la *función de validación* empleada en el algoritmo de decisión.

La función de validación se basa en comprobar si la nueva posición se encuentra en una determinada zona de validación. Esta zona está formada por dos semiesferas de distinto tamaño y una sección cónica, formando la figura que se puede observar en la ilustración 23. El vector distancia entre las posiciones de la articulación en los dos últimos esqueletos retornados determina el tamaño y la dirección del eje central de la zona de validación. En cuanto a la posición de esta zona, la última posición de la articulación se toma como origen o centro de la semiesfera de menor tamaño. El radio de la semiesfera mayor está determinado por el valor del radio de la semiesfera menor multiplicado por un ratio mayor a la unidad que será aprendido de manera automática.

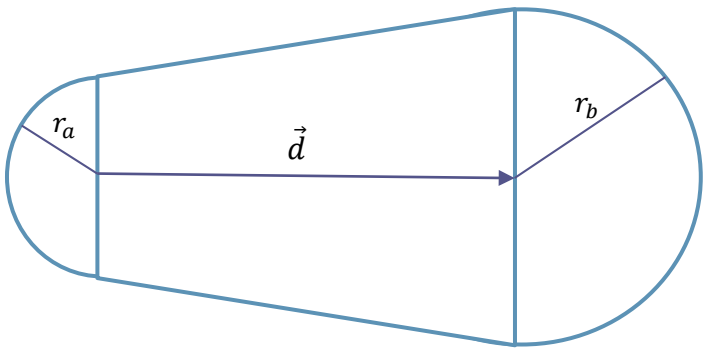

Ilustración 23 – Zona de validación del módulo Decision Maker. Vector distancia ($\vec{d}$), radio de la semiesfera menor (r_a) y radio de la semiesfera mayor (r_b)

Una vez determinada la que será la función de validación, se procede a detallar el algoritmo empleado en la decisión (véase el pseudocódigo 1). Se distinguen tres casos principales: Ni *DepthYOLO* ni Kinect extraen la articulación, Kinect sí que extrae la

articulación pero no *DepthYOLO* y *DepthYOLO* devuelve la articulación (independientemente de Kinect).

El caso de que ninguno de los dos sistemas devuelva la posición de la articulación, el resultado es no dar ninguna posición a la articulación.

Para el caso en el que solo retorne la articulación el sistema Kinect, aplicamos la función de validación a esta posición. Si cumple el criterio y el sistema tiene resultados anteriores, se retorna como posición final la posición que retornaba Kinect. En caso contrario, no se retorna ninguna posición como resultado.

```
SI ¬depthYOLO ENTONCES
        SI ¬kinect ENTONCES
                RETORNA nulo
        SINO SI valido(kinect) ENTONCES
                RETORNA kinect
        SINO RETORNA nulo
        FIN SI
SINO
        SI valido(depthYOLO) ENTONCES
                RETORNA depthYOLO
        SINO SI kinect ENTONCES
                SI valido(kinect) ENTONCES
                        RETORNA kinect
                SINO
                        medio=depthYOLO + (kinect−depthYOLO)/2
                        SI valido(medio) ENTONCES
                                RETORNA medio
                        FIN SI
                FIN SI
        SINO
                RETORNA depthYOLO
        FIN SI
FIN SI
```

Pseudocódigo 1 – Módulo Decision Maker

Por último, el último caso y más complejo, es cuando *DepthYOLO* sí retorna una posición para la articulación. El primer paso consiste en aplicar la función de validación a la posición retornada por *DepthYOLO*. En caso de cumplir la función, se retorna como posición final la posición obtenida por *DepthYOLO*. En caso de que no cumpla la función de validación, se aplica dicha función a los datos retornados por Kinect, en caso de que Kinect haya retornado alguna posición. Si cumple la función, el sistema retorna como posición final la misma posición obtenida por Kinect. En caso negativo, se comprueba si ambos sistemas han devuelto posición para la articulación. Si es así, se calcula la posición media entre las dos posiciones retornadas por los sistemas y se

aplica a este punto medio la función de validación. En caso afirmativo retorna el punto medio como solución. Sin embargo, si no cumple dicha función, se retorna la posición obtenida por *DepthYOLO*. En cuanto a la única posibilidad restante, en caso de que *DepthYOLO* retorne un valor que no cumple la función de validación y además no hallan datos por parte de Kinect, el sistema retorna directamente la posición de *DepthYOLO*.

Una vez implementado el algoritmo, se realiza un aprendizaje de los parámetros de la función de validación. Se busca maximizar el ratio de aciertos, mediante aprendizaje automático y empleando los datos utilizados en los aprendizajes del módulo *DepthYOLO*.

6. Pruebas y Resultados

Por motivos de confidencialidad con Honda, como anteriormente se ha comunicado, no se pueden publicar abiertamente las grabaciones empleadas en el conjunto de pruebas ni algunos resultados.

El conjunto de pruebas se compone de 16 vídeos, de una duración comprendida entre los 10 y los 30 segundos. En cada video aparece siempre en pantalla una persona de manera que todos los *frames* del video deben contener únicamente un esqueleto a excepción de dos vídeos donde aparece el problema del fantasma comentado en el capítulo 3. Los videos están compuestos por dos sujetos varones de 28 y 30 años de edad con una altura de 177 y 165 centímetros respectivamente. Ambos varones realizan diversos movimientos como andar, sentarse o acostarse.

Dado que se desconoce realmente la posición de las articulaciones, no se encuentra disponible ningún *ground truth* con el que comparar los resultados. Por ello, la mayoría de mediciones que se realizan sobre el conjunto de prueba están basadas en datos relativos a los esqueletos retornados por Kinect. Las mediciones realizadas son las siguientes:

- Tasa de cobertura: porcentaje de *frames* que contienen información sobre un esqueleto, sean correctos o incorrectos.
- Tasa de corrección del problema fantasma: porcentaje de *frames* solucionados donde Kinect devuelve un esqueleto sin aparecer ningún sujeto en la imagen.
- Tasa de corrección de esqueleto espasmódico: porcentaje de *frames* corregidos donde Kinect comete el fallo del esqueleto espasmódico, movimiento incoherente de las articulaciones entre *frames* consecutivos.
- Tasa de corrección de gestos imposibles: porcentaje de *frames* corregidos donde Kinect retorna un esqueleto que en caso de estar formado por huesos dichos huesos interseccionarían.
- Tasa de corrección del reconocimiento lateral: porcentaje de *frames* corregidos en los cuales Kinect tenía confusión entre el lateral izquierdo y el derecho del esqueleto, cometiendo el error de etiquetarlos incorrectamente.

En la tabla 1 se puede observar los resultados obtenidos. El tiempo de ejecución del sistema se ha calculado aunque los resultados no pueden ser expuestos por la confidencialidad argumentada al comienzo de la sección. Pese a ello, ha quedado verificado en laboratorio que el sistema funciona en tiempo real.

Tasas según método	Kinect	DepthYOLO	Decision Maker
Cobertura	61,73%	72,36%	80,60%
Problema fantasma	-	100,00%	100,00%
Esqueleto espasmódico	-	83,98%	69,89%
Gestos imposibles	-	71,36%	70,57%
Reconocimiento lateral	-	92,59%	82,47%

Tabla 1 – Resultados de las pruebas realizadas

En los resultados se puede contemplar la baja cobertura ofrecida por Kinect. Esto se debe a que el sistema no es capaz de extraer un esqueleto si el sujeto se encuentra de espaldas o no se encuentra el cuerpo completo dentro de la escena. Sin embargo *DepthYOLO* muestra mejor resultado dado que no necesita del cuerpo entero para poder extraer la posición de las articulaciones. Esta tasa de cobertura puede ser aún mayor si se dispone de un conjunto de *train* donde el sujeto no se encuentre mirando a cámara dado que el empleado en este proyecto dispone pocas de estas imágenes. En cuanto a *Decision Maker* hace aumentar la cantidad de esqueletos extraídos en aproximadamente un 20% por lo que parece cumplir su cometido en cuanto a aportar más esqueletos que los que Kinect devuelve.

Desde el punto de vista de la tasa de corrección del problema del fantasma, *DepthYOLO* no detecta esqueletos en aquellas grabaciones donde Kinect sí devolvía esqueletos pese a no aparecer ningún sujeto en imagen. Por ello, en el conjunto de prueba *DepthYOLO* obtiene un 100% como tasa de corrección. *Decision Maker* también realiza de manera efectiva su tarea retornando los resultados ofrecidos por *DepthYOLO* (véase la ilustración 24).

Ilustración 24 – Esqueletos devueltos en una imagen con problema del fantasma. Kinect (izquierda), DepthYOLO (centro), Decision Maker (derecha)

Observando la tasa de corrección del esqueleto espasmódico, *DepthYOLO* comete un menor número de fallos que Kinect. La tasa de corrección tan alta, 83,98%, muestra que aunque *DepthYOLO* no hace uso de los esqueletos retornados anteriormente, es capaz de realizar un reconocimiento de imagen más preciso que el

realizado por Kinect. Por otro lado, la disminución de esta tasa de corrección por parte de *Decision Maker*, situada en 69,89%, frente a la de únicamente *DepthYOLO* indica que *Decision* Maker reúsa esqueletos que *DepthYOLO* localizó correctamente. Pese a este defecto, mejora en gran cantidad los fallos cometidos por Kinect (véase la ilustración 25).

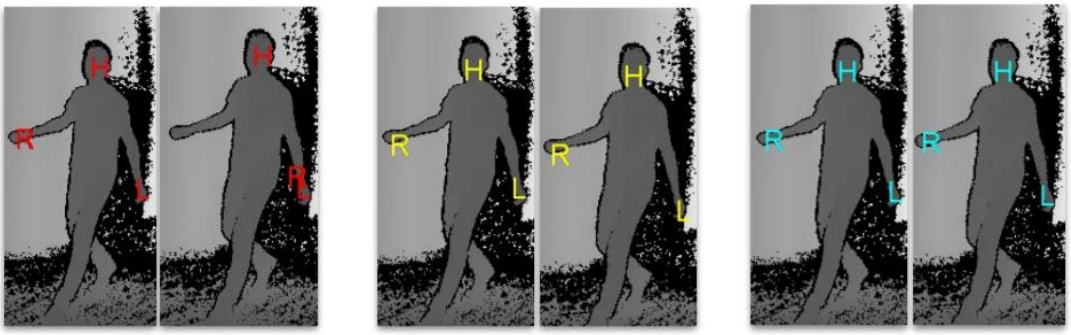

Ilustración 25 – Posición de las articulaciones principales en dos frames consecutivos con problema de esqueleto espasmódico. Kinect (izquierda), DepthYOLO (centro), Decision Maker (derecha)

Si nos centramos en la tasa de corrección de gestos imposibles, se puede observar cómo tanto *DepthYOLO* como *Decision Maker* obtienen un buen resultado. *DepthYOLO* mejora estos esqueletos ya que si no es capaz de localizar una parte de la articulación, como por ejemplo cuando una parte del sujeto se encuentra oculta, no retorna dicha articulación. A su vez, Decision Maker rechaza la articulación extraída por Kinect ya que no se ajusta a las condiciones internas (véase la ilustración 26).

Ilustración 26 – Posición de las distintas articulaciones en un frame con problemas de gestos imposibles. Kinect (izquierda), DepthYOLO (centro), Decision Maker (derecha)

Por último, observando la tasa de corrección del reconocimiento lateral, ambos módulos obtienen resultados interesantes. El módulo *DepthYOLO* se ajusta mejor al reconocimiento lateral ya que el algoritmo no interpola posiciones por lo que, como se ha comentado anteriormente, si no reconoce una sección del sujeto no lo interpola del resto de datos. En este caso, *Decision Maker*, pese a retornar una buena tasa de corrección no alcanza correctamente su objetivo ya que desecha en varias ocasiones los

resultados correctos de *DepthYOLO* decantándose por los resultados de Kinect (véase la ilustración 27).

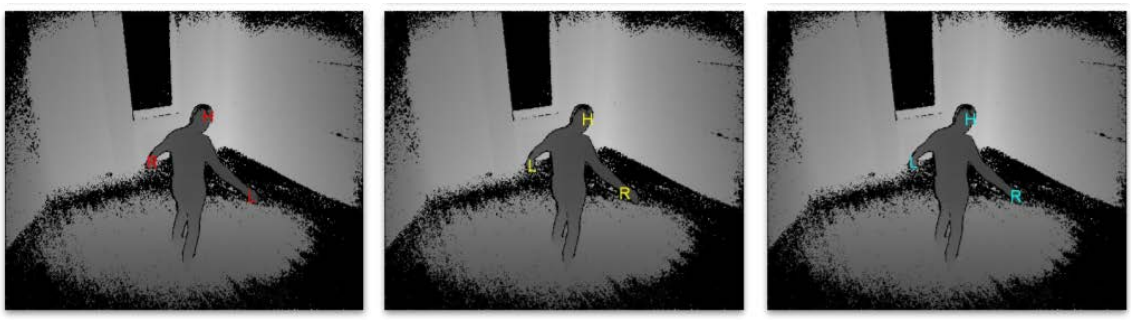

Ilustración 27 – Posición de las articulaciones en un frame con problemas de reconocimiento del lateral. Kinect (izquierda), DepthYOLO (centro), Decision Maker (derecha)

7. Conclusiones

Tras analizar los resultados obtenidos por el modelo planteado para este proyecto, se pueden extraer las siguientes conclusiones.

Por una parte, el sistema formado por una red neuronal convolucional y un módulo que integra los resultados de Kinect con los de la red neuronal solventa en gran medida los fallos cometidos por Kinect en todos los problemas planteados: problema del fantasma, esqueleto espasmódico, gestos imposibles y reconocimiento del lateral.

Por otra parte, al no poder realizar una comparación con otros sistemas y a la vista de los resultados, no se puede confirmar si la red neuronal convolucional empleada es la red neuronal más efectiva. Puede ser que emplear otro tipo de red neuronal convolucional o red neuronal profunda resulte más efectivo para cumplir los objetivos del presente proyecto.

Por último, en cuanto a las conclusiones sobre los resultados obtenidos, la ejecución del sistema confirma que puede ser empleado para entornos de trabajo en tiempo real.

Desde un punto de vista más subjetivo, se pueden extraer las siguientes conclusiones:

Por un lado, emplear la inteligencia artificial para mejorar los resultados que Kinect retorna, ha demostrado unos resultados prometedores. Específicamente, dado el estado del arte actual, se recomienda emplear redes neuronales donde se tenga en cuenta la disposición espacial de los datos, como ocurre en las redes neuronales convolucionales. *DepthYOLO* retorna buenos resultados para las articulaciones probadas en este proyecto. Teniendo en cuenta que el campo de las redes neuronales convolucionales y los sistemas de reconocimiento de objetos son campos en gran expansión, con el paso de pocos años aparecerán con toda seguridad redes neuronales más óptimas para esta tarea, tanto en resultados más robustos como en menor consumo de recursos y una disminución en el tiempo de ejecución.

Por otro lado, el módulo *Decision Maker* puede ser mejorado empleando otras alternativas como un conjunto de reglas, o bien una red neuronal, con varios parámetros que se puedan ajustar al entorno del problema.

Quedaría como alternativa emplear otros sistemas que no empleen el sensor Kinect pero, como se argumentaba al comienzo de este estudio, se intenta no alcanzar este cambio extremo dada la gran inversión realizada por todos los organismos y laboratorios que decidieron invertir en sistemas Kinect.

7.1. Relación del trabajo desarrollado con los estudios cursados

La mayoría de trabajos relacionados con el campo de la informática tienen un gran componente matemático por lo que aquellas asignaturas como análisis matemático o matemática discreta resultan de gran necesidad. Otro aspecto básico en trabajos referidos a este campo es la necesidad de habilidades y conocimientos respecto a fundamentos y diversidad de lenguajes de programación. Por ello, desde las bases estudiadas respecto a la programación pasando por estructuras de datos y algoritmos se vuelven fundamentales. Además, el presente proyecto también se apoya en estudios sobre redes para poder conectar los distintos módulos.

De manera más específica respecto al presente proyecto, resultan de interés varios campos. Por un lado, las asignaturas relacionadas con la imagen como puede ser programación gráfica e interfaces gráficas, donde se fundamentan las bases para el tratamiento de imágenes, tanto en cuanto a métodos de almacenamiento como a análisis y modificación de estas.

Por otro lado, la inteligencia artificial requiere muchos conceptos de diversas asignaturas. A parte de las relacionadas con la programación y el diseño de algoritmos eficientes, se necesitan buenas bases en estadística para, más tarde, poder comprender y aplicar la diversidad de conceptos que se pueden encontrar en la inteligencia artificial. Gracias al master sobre inteligencia artificial que realicé en la UPV, mi conocimiento en este campo se expandió sobremanera, permitiendo comprender las interacciones intrínsecas entre diversas propiedades de los distintos algoritmos empleados en el *machine learning*. Resultaron de gran interés para este estudio las asignaturas de reconocimiento de formas; análisis estadístico de formas; avances en informática gráfica; herramientas y aplicaciones de la inteligencia artificial; lingüística computacional; métodos estadísticos en tecnologías del lenguaje; programación gráfica; reconocimiento de imágenes y tratamiento de la imagen digital.

8. Trabajos Futuros

En los últimos meses han aparecido nuevos sistemas de reconocimiento de formas que podrían ser adaptados a imágenes de profundidad. Una línea de investigación se puede orientar a comprobar qué resultados se logran obtener con estos nuevos sistemas.

Por otra parte, otra de las líneas que pueden derivar de este trabajo, son aquellas que busquen métodos alternativos al módulo de toma de decisiones. Estas se pueden centrar en sistemas de razonamiento o en redes neuronales recursivas que fusionen los esqueletos según el grado de confianza de los métodos de extracción e interpolen las articulaciones que no se localicen correctamente.

9. Terminología

Average pooling: Capa neuronal que divide la capa anterior en cuadrantes de tamaño mayor a 1x1 píxeles y retorna el valor medio dentro de dicho cuadrante.

Backpropagation: Término inglés referido a la propagación hacia atrás de errores o retropropagación. Hace referencia a la dirección en que se propaga o avanza el error cometido por cada neurona, necesario para realizar el ajuste durante el aprendizaje de una red neuronal.

Bounding Box: Recorte rectangular que limita una sección en una imagen.

Capa fully-connected: Capa neuronal donde las neuronas se encuentran conectadas a las neuronas de la capa anterior en una relación todas a todas.

Capa softmax: Esta capa realiza una función similar a *argmax* tomando como entrada la probabilidad de las distintas clasificaciones posibles y devolviendo como salida únicamente la clasificación con mayor probabilidad.

Clasificación binaria: Función con solo dos posibles salidas de manera que solo puede determinar si un dato es o no es de una clase.

Confianza: Ratio o porcentaje que mide el nivel de seguridad de un etiquetado o clasificación.

Dropout: Capa neuronal que bloquea los resultados de las neuronas de la capa anterior si son inferiores a un valor umbral.

Ejecución en tiempo real: Sistemas interactivos de manera continua, cumpliendo las restricciones temporales presentes.

Entorno: Conjunto de circunstancias y características que rodean el problema o sistema.

Escena: Área física captada por el dispositivo Kinect.

Esqueleto: Conjunto de articulaciones que forman la información de cada sujeto.

Etiqueta: Clase o nombre distintivo con el que se clasifica un dato, comparándolo con otras clases presentes en el entorno del problema.

Extracción del esqueleto: Obtención del esqueleto de un sujeto como resultado de la ejecución de un algoritmo.

Filtros de Kalman: Algoritmo u operación de corrección empleada para eliminar el ruido entre los datos.

Filtros Haar: Algoritmo u operación empleada para detectar diferencias de intensidad entre regiones de la imagen de manera óptima.

Frame: Cada una de las imágenes que forman una grabación o vídeo.

Función aditiva: Función sumatoria de todos los datos de entrada

Función de activación: Función matemática que devuelve el valor que debe enviar a otras neuronas empleando como entrada el valor de la función aditiva.

Función ReLU (Rectified Linear Unit): Función empleada como función de activación que devuelve el valor de entrada en caso de ser positivo o 0 en caso contrario.

Función sigmoide: Función empleada como función de activación cuyos valores de salida están comprendidos entre 0 y 1.

Función tangencial hiperbólica: Función empleada como función de activación cuyos valores de salida están comprendidos entre -1 y 1.

Información residual: Información que se mantiene invariable a lo largo de más de una capa neuronal.

Interpolar: Cálculo del valor del dato desconocido basado en datos próximos que sí se conocen.

Articulación: Posición de una extremidad o articulación del sujeto.

Linealmente separable: Se dice que dos conjuntos son linealmente separables si y solo si se puede trazar un hiperplano que divida el espacio de características de manera que cada conjunto quede a un lado distinto de dicho hiperplano.

Machine Learning: Es una rama de la inteligencia artificial que comprende aquellos algoritmos capaces de configurarse automáticamente mediante un algoritmo de aprendizaje.

Max pooling: Capa neuronal que divide la capa anterior en cuadrantes de tamaño mayor a 1x1 píxeles y retorna el valor máximo dentro de dicho cuadrante.

Min pooling: Capa neuronal que divide la capa anterior en cuadrantes de tamaño mayor a 1x1 píxeles y retorna el valor mínimo dentro de dicho cuadrante.

Neuronas locales: Conjunto de neuronas que forman la vecindad de una neurona central.

Operación convolucional: Operación matemática que unifica dos funciones bajo una tercera función. Comúnmente llamado distorsión cuando se refiere a imágenes. Es una operación típica en las medias móviles.

Puntos de interés: Puntos de una imagen que resulta de especial interés ya que según la distribución de varios de estos se puede determinar la posible etiqueta de los datos.

Regiones de interés: Sección o región de una imagen que resulta de especial interés por ser distinto al resto de regiones de la imagen y que ayuda en gran medida a determinar la posible etiqueta de los datos.

Ruido: Pequeñas alteraciones que interfieren con los datos.

Sujeto: Usuario que se muestra frente al sensor Kinect.

Threshold: Palabra inglesa con mismo significado que umbral de validación.

Umbral de validación: Valor a partir del cual se toma como válido un dato, rechazado en caso de no llegar a dicho valor.

10. Referencias

Bachtiar, M. M., Nuzula, F. F. & Wasista, S., 2016. Gait with combination of swing arm feature extraction for gender identification using kinecct skeleton. *International seminar on Intelligent Technology and its application,* pp. 79-82.

Bell, S., Zitnick, C. L., Bala, K. & Girshick, R., 2016. Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks. *Conference: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Blier, L., 2016. *A brief report of the Heuritech Deep Learning Meetup #5 - Heuritech.* [En línea]
Dirección: https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/
[Último acceso: julio 2018].

Bryson, A. E., 1961. A gradient method for optimizing multi-stage allocation processes. *Proceedings of the Harvard Univ. Symposium on digital computers and their applications.*

Chollet, F., 2017. Xception: Deep Learning with Depthwise Separable Convolutions. *CVRP 2017.*

Davisclick, 2018. *Usuario:Davisclick/Taller - Wikipedia.* [En línea]
Dirección: https://es.wikipedia.org/wiki/Usuario:Davisclick/Taller
[Último acceso: julio 2018].

Epstein, Z., 2013. *Microsoft says Xbox 360 sales have surpassed 76 million units, Kinect sales top 24 million - BGR.* [En línea]
Dirección: https://bgr.com/2013/02/12/microsoft-xbox-360-sales-2013-325481/
[Último acceso: agosto 2018].

Handrich, S. & Al-Hamadi, A., 2015. Full-Body Human Pose Estimation. *ACIVS 2015, LNCS 9386,* pp. 287-298.

He, K., Zhang, X., Ren, S. & Sun, J., 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

Honda, 2018. *About HRI.* [En línea]
Dirección: http://www.jp.honda-ri.com/en/about/index.html
[Último acceso: agosto 2018].

Jasdeep06, 2017. *Into-Backpropagation.* [En línea]
Dirección: https://jasdeep06.github.io/
[Último acceso: agosto 2018].

Kaenchan, S., Mongkolnam, P., Watanapa, B. & Sathienpong, S., 2013. Automatic Multiple Kinect Cameras Setting for Simple Walking Posture Analysis. *International Computer Science and Engineering Conference*, pp. 245-249.

Kar, A., 2010. Skeletal Tracking using Microsoft Kinect. *Methodology, vol. 1, no 1,* , p. 11.

Lachat, E., Macher, H., Landes, T. & Grussenmeyer, P., 2015. Assessment and Calibration of a RGB-D Camera (Kinect v2 Sensor) Towards a Potential Use for Close-Range 3D Modeling. *Remote Sensing*, Issue 7, pp. 13070 - 13097.

Lin, M., Chen, Q. & Yan, S., 2014. Network in network. *arXiv,* Issue 1312.4400.

Livingston, M. A., Sebastian, J., Ai, Z. & Decker, J. W., 2012. Performance Measurements for the Microsoft Kinect Skeleton. *IEEE Virtual Reality 2012,* pp. 119-120.

Ma, J. & Choi, S., 2014. Kinematic skeleton extraction from 3D articulated models. *Computer-Aided Design 46,* pp. 221 - 226.

Microsoft, 2018. *Kinect for Windows.* [En línea]
Dirección: https://developer.microsoft.com/en-us/windows/kinect
[Último acceso: agosto 2018].

Microsoft, 2018. *Kinect for Windows SDK 2.0.* [En línea]
Dirección: https://www.microsoft.com/en-us/download/details.aspx?id=44561
[Último acceso: julio 2018].

Nishimura, J. & Kuroda, T., 2010. Versatile Recognition Using Haar-Like Feature and Cascaded Classifier. *IEEE SENSORS JOURNAL, VOL. 10, NO. 5*, pp. 942 - 951.

nVIDIA, 2018. *CUDA Toolkit.* [En línea]
Dirección: https://developer.nvidia.com/cuda-toolkit
[Último acceso: agosto 2018].

Obdrzálek, S. y otros, 2012. Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population. *34th Annual International Conference of the IEEE EMBS,* pp. 1188-1193.

Plunkett, L., 2010. *Report: Here Are Kinect's Technical Specs - Kotaku.* [En línea]
Dirección: https://kotaku.com/5576002/here-are-kinects-technical-specs
[Último acceso: agosto 2018].

Radzi, F. y otros, Enero 2015. An Improved Retraining Scheme for Convolutional Neural Network. *Journal of Telecomunication, Electronic and Computer Engineering,* 7(1), pp. 5 - 9.

Records, G. W., 2011. *Fastest-selling gaming peripheral - Guinness World Records.* [En línea]

Dirección: http://guinnessworldrecords.com/world-records/fastest-selling-gaming-peripheral
[Último acceso: agosto 2018].

Redmon, J., 2016. *YOLO Version 2*. [En línea]
Dirección: https://pjreddie.com/darknet/yolov2/
[Último acceso: agosto 2018].

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. *CVPR 2016*.

Reisinger, D., 2017. *Microsoft Has Finally Killed the Kinect Xbox Sensor - Fortune*. [En línea]
Dirección: http://fortune.com/2017/10/25/microsoft-kinect-xbox-sensor/
[Último acceso: agosto 2018].

Rosenblatt, F., 1957. The perceptron - a perceiving and recognizing automaton. *Cornell Aeronautical Laboratory, INC.*, pp. Report N: 85-460-1.

Rosenblatt, F., 1961. Principles of neurodynamics - Perceptrons and the teory of the brain mechanisms. *Cornell Aeronautical Laboratory, INC.*.

Sena, S., 2017. *Pengenalan Deep Learning Part 2 : Multilayer Perceptron*. [En línea]
Dirección: https://medium.com/@samuelsena/pengenalan-deep-learning-part-2-multilayer-perceptron-e8f98d625b09
[Último acceso: agosto 2018].

Shu, J., Hamano, F. & Angus, J., 2014. Application of extended Kalman filter for improving the accuracy and smoothness of Kinect skeleton-joint estimates. *Journal of Engineering Mathematics. October 2014, Volume 88, Issue 1*, pp. 161 - 175.

Simonyan, K. & Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. *Conference paper at ICLR 2015*.

Sinha, A., Chakravarty, K. & Bhowmick, B., 2013. Person Identification using Skeleton Information from Kinect. *The Sixth International Conference on Advances in Computer-Human Interactions (ACHI 2013)*, pp. 101-108.

Szegedy, C. y otros, 2015. Going deeper with convolutions. *Computer Vision and Pattern Recognition (CVPR)*.

Tang, J., Deng, C. & Huang, G.-B., 2016. Extreme Learning Machine for Multilayer Perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4), pp. 809-821.

Tripathy, S. R., Chakravarty, K., Sinha, A. & Debatri Chatterjee, S. K. S., 2017. Constrained Kalman Filter For Improving Kinect. *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*.

Valcik, J., Sedmidubsky, J. & Zezula, P., 2015. Improving Kinect-Skeleton Estimation. *ACIVS 2015, LNCS 9386,* pp. 575 - 587.

Veen, F. v., 2016. *The Neural Network Zoo - The Asmiov Institute.* [En línea] Dirección: https://www.asimovinstitute.org/author/fjodorvanveen/ [Último acceso: agosto 2018].

Vulliet, J., 2017. *Kinect - vvvv: a multipurpose toolkit.* [En línea] Dirección: https://vvvv.org/documentation/kinect [Último acceso: agosto 2018].

Wang, K. y otros, 2016. Novel correspondence-based approach for consistent human skeleton extraction. *Multimed Tools Appl,* p. 11741–11762.

Yu, L., Han, Q. & Niu, X., 2014. An improved contraction-based method for mesh skeleton extraction. *Multimed Tools Appl,* p. 1709–1722.

Zeiler, M. D. & Fergus, R., 2013. Visualizing and Understanding Convolutional Networks. *arXiv:1311.2901.*

Zhanga, F. y otros, 2015. Skeleton extraction based on anisotropic partial differential equation. *Optik 126,* p. 3692–3697.

11. Agradecimientos

Me gustaría agradecer a Francisco José Abad Cerdá, director del proyecto, las gestiones y el esfuerzo realizado para la buena ejecución de este Trabajo Fin de Grado. Además, agradezco toda la ayuda ofrecida por la profesora M. Carmen Juan Lizandra.

También me gustaría agradecer a la empresa *Honda Research Institute Japan* la oportunidad de realizar este estudio en sus instalaciones japonesas y el material aportado por ellos. Especial mención a los compañeros: Adam Adair, Borja Fourquet, Gonzalo Ferrer, Graeme Deering, Julien Collart, Kabir Chattopadhyay, Kristof Boucher y Léo Ghafari.

Por último, agradecer a familiares y amigos todo el apoyo moral para los momentos de desaliento y desesperación que han ido surgiendo a lo largo del proyecto.