Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

Itinerario de Ingeniería de Sistemas Informáticos

Desarrollo y evaluación de técnicas de visión artificial aplicadas a un problema de reconocimiento gestual

Trabajo Fin de Máster (31105151)

Alumno: Juan Luis Álvarez Herradón

Director del trabajo: Carlos Cerrada Somolinos

Curso 2017/2018 - Convocatoria de Junio

Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

Itinerario de Ingeniería de Sistemas Informáticos

DESARROLLO Y EVALUACIÓN DE TÉCNICAS DE VISIÓN ARTIFICIAL APLICADAS A UN PROBLEMA DE RECONOCIMIENTO GESTUAL

Trabajo Fin de Máster (31105151)

Alumno: Juan Luis Álvarez Herradón

Director del trabajo: Carlos Cerrada Somolinos

Curso 2017/2018 - Convocatoria de Junio

Declaración jurada de autoría del trabajo científico, para la defensa del trabajo fin de master

Fecha: 06/06/2018

Quién suscribe:

Autor(a): Juan Luis Álvarez Herradón D.N.I/N.I.E/Pasaporte.: 51477063G

Hace constar que es la autor(a) del trabajo:

Desarrollo y evaluación de técnicas de visión artificial aplicadas a un problema de reconocimiento gestual

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores, se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente por otros autores, en soporte papel ni en formato digital.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo.: Juan Luis Álvarez Herradón

Autorización

Autorizo a la **Universidad Nacional de Educación a Distancia** a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Juan Luis Álvarez Herradón

Índice

1. Introducción

El presente proyecto se enmarca en la asignatura Trabajo Fin de Máster del correspondiente Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos, cursado en el curso académico 2017/2018. Dicho proyecto surgió de la propuesta del director del mismo, Don Carlos Cerrada Somolinos, de llevar a cabo un trabajo sobre técnicas de visión artificial aplicadas a un problema de reconocimiento facial.

En este ámbito, nos encontramos en una época donde son múltiples las tecnologías que adoptan esta funcionalidad por las ventajas que aporta, y se puede acceder a una amplia cantidad de proyectos de software libre que resuelven de múltiples maneras el reto de identificar un rostro humano. Tras una puesta en común de las diferentes posibilidades e inquietudes y, dado el avanzado estado del arte de estas técnicas de reconocimiento facial, finalmente se orientó hacia un estudio sobre el reconocimiento gestual, encarando los diferentes desafíos que promueve para un sistema inteligente el analizar características propias del cuerpo humano diferentes al rostro; en este caso, la mano.

De esta forma, este trabajo se centra en analizar las complejidades que se presentan dentro del análisis de imagen para localizar una mano y reconocer diferentes gestos interpretables como comandos de entrada a un sistema.

Para ello, se ha llevado a cabo un desarrollo funcionalmente sencillo, basado en Java y OpenCV, que permitiera realizar operaciones en tiempo real sobre una fuente de captura de imágenes como una cámara web.

1.1. Motivación

En los tiempos actuales, donde la tecnología evoluciona cada vez mejor y más deprisa, son múltiples las ramas donde se acude a técnicas de inteligencia artificial de forma que se amplíen las capacidades de interacción con diferentes sistemas. En el caso concreto del aeromodelismo, la prestigiosa marca DJI, dedicada principalmente al desarrollo de drones, presentó en mayo de 2017 un sistema de control mediante gestos incorporado a su modelo *Spark*, que posteriormente se fue introduciendo en otros modelos.

Figura 1: DJI Spark (2017)

Este sistema de control por gestos reconoce la palma de la mano del usuario de forma que hace prescindible el mando de control remoto para el manejo más básico del dron. Una vez que el dron es capaz de localizar la mano, puede llevar a cabo diferentes funciones de posicionamiento y captura de imágenes con sencillos gestos, cambiando el paradigma de interacción con drones por completo.

Que esta tecnología se incorporase a un campo relativamente complejo como es el manejo de una aeronave no tripulada amplía el espectro potencial de usuarios, dado que permite que personas con poca habilidad a los mandos de un control remoto puedan llevar a cabo funciones simples en este tipo de dispositivos.

Figura 2: DJI Spark controlado por gestos

El reconocimiento facial es una función que incorporan una amplia variedad tecnologías en un extenso rango de usos a día de hoy: ordenadores portátiles, cámaras de fotos, smartphones, sistemas de vigilancia, etc. Sin embargo, resulta relativamente novedoso el uso de gestos como posible único componente de control en un dispositivo. Por ello, el presente TFM surge en esta motivación: estudiar qué retos supone enfrentarse a este paradigma, y en particular si es alcanzable por cualquiera o requiere de un análisis y procesado tan complejo que solo grandes productoras de hardware o software son capaces de afrontarlo y llevarlo al público general.

1.2. Estado del arte

El interés sobre el reconocimiento de objetos mediante técnicas de visión artificial muestra un crecimiento imparable, y por ello el reconocimiento de la mano no se ha quedado atrás en este aspecto. Existen diversas publicaciones al respecto, en donde se realizan estudios de carácter matemático y teórico.

Una de las vías utilizadas para llevar a cabo el reconocimiento de gestos es el uso de redes neuronales convolucionales (Xu, 2017) [11], que clasifica las diferentes posiciones de un conjunto de imágenes de manos para, con él, determinar si la imagen de entrada está realizando un gesto conocido.

Entre diversos problemas que suelen surgir en este ámbito, uno es el reconocimiento de secuencias continuas de gestos, dado que al tener una tasa de frames de una velocidad constante y suficiente para que un vídeo se muestre natural al ojo humano, pueden darse múltiples frames que por sus características no permitan el reconocimiento del gesto y rompa la continuidad de la secuencia a reconocer.

Sin duda una de las técnicas más estudiadas e incluso utilizadas es el reconocimiento de la mano en 3D (Caputo, Denker, Dumus, Umlauf, 2012) [3]. Para esto, se puede tratar de extraer un modelo 3D a partir de una imagen 2D de la mano, algo complicado y no siempre exitoso, o se puede utilizar una cámara especial, o bien una cámara estéreo o 3D, o bien una cámara junto con sensores de rango, como puede ser Kinect o Leap Motion, que son productos cerrados.

Figura 3: Kinect (izquierda). Figura 4: Leap Motion (derecha)

Al tener una mayor cantidad de información con la que trabajar, sea por contar con dos imágenes, o sea por contar con una imagen y un conjunto de datos auxiliares como la profundidad de cada píxel, las posibilidades y las técnicas a utilizar se incrementan sustancialmente y por ello suele ofrecer resultados más fiables y consistentes.

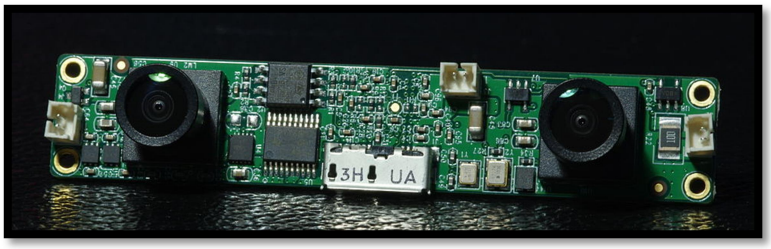

Figura 5: Cámara estéreo o 3D

Para el caso que acontece, en el que únicamente se va a utilizar una imagen 2D para detectar la mano y su gesto, existen múltiples librerías y frameworks entre los diversos lenguajes de programación más usados. En particular, para Java, es común utilizar librerías como JAI (Java Advanced Imaging), ImageJ, JavaCV, OpenIMAJ u OpenCV. Todas ellas tienen diferente madurez (incluso alguna está descontinuada), pero suficiente para llevar a cabo las operaciones que requiere una análisis como el que se ha de llevar a la hora de detectar una mano. Por ello, cualquiera de las librerías indicadas parece válida para este trabajo, si bien su elección afecta al diseño del algoritmo dado que no todas las operaciones son equivalentes en todas las librerías. Sin embargo, lo positivo de todas ellas es la capacidad de ser ejecutadas sobre cualquier sistema operativo que permita ejecutar Java, ofreciendo una alta portabilidad al sistema.

Muchos de los estudios realizados en la línea del presente utilizan guantes o algún tipo de contraste en la mano para diferenciar de una forma fiable la mano del resto del escenario. Esto se debe al aislamiento de la mano que, en muchos casos, se hace mediante la intensidad de la escala de grises de la imagen, teniendo así múltiples zonas coincidentes con la mano, como puede ser la piel del rostro o del brazo.

Existen proyectos para detectar la piel y su color utilizando mapas de probabilidad (Lee y Yoo, 2002) [7], basando la fiabilidad de este proceso en un entrenamiento mediante un conjunto de imágenes y cálculos probabilísticos.

2. Análisis

2.1. Problema

El análisis de una imagen o matriz de píxeles requiere una serie de cálculos matemáticos diferentes en complejidad. En el caso propuesto, se ha de diseñar un algoritmo que sea capaz de aislar, primeramente, la mano del sujeto. Posteriormente, se debe realizar un análisis morfológico para localizar características únicas de diferentes gestos de la mano, extrayendo así los diferentes comandos de entrada para el sistema propuesto.

Para afrontar este problema surgen diferentes aproximaciones. Para binarizar la imagen de forma que capturemos la región de interés coincidente con la mano, el método más básico consiste en transformar la imagen a escala de grises para identificar el límite o borde de la palma. Aunque inicialmente se presenta como una solución básica y adecuada, hay que tener en cuenta que las intensidades en la imagen son completamente impredecibles dado que dependen del fondo, el primer plano, los colores y la luz de la escena...

Estos factores no se pueden controlar a priori. La intensidad de la luz en la imagen se puede controlar con el control de exposición de la cámara (que generalmente es automático), pudiendo compensar destellos o condiciones de poca luminosidad. Objetos en primer plano o en el fondo cuya intensidad de gris sea similar al de la mano tampoco son controlables y pueden incidir en la selección de nuestra zona de interés.

Para apoyar la selección de la zona de la mano ha dado buen resultado tener en cuenta el color más que la escala de grises de la imagen. De esta manera, se obtiene mayor distinción entre lo que es mano (piel) y lo que no. Es por esto que la selección de un promedio de color ha presentado buenos resultados en la técnica utilizada que se muestra más adelante.

Con todo, otro de los problemas es lidiar con la decisión de cuál de las zonas recogidas por la selección de color es la que interesa. Hay que tener en cuenta que son varias las zonas de piel que generalmente se exponen: manos, brazos, cara, piernas, pies... Para distinguir entre todas ellas, es necesario realizar un análisis en busca de características que definan unívocamente una mano con respecto al resto de partes del cuerpo.

2.2. Técnica utilizada

2.2.1. Detección de la mano

Como ha sido explicado en el punto anterior, el primer paso es binarizar la imagen de manera que se aísle la piel del resto. Para llevar a cabo esta extracción del fondo, se toma manualmente una muestra del color, en perfil HSV, de la piel, necesaria como punto de partida dado que las condiciones de luz y la exposición de la cámara son totalmente variables.

Para llevar a cabo este primer proceso, tras comprobar la documentación de la librería OpenCV [6], se utiliza la función `Core.inRange(src, lowerb, upperb, dst) que` lleva a cabo la binarización de acuerdo a la siguiente especificación:

$$dst(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0 \wedge lowerb(I)_1 \leq src(I)_1 \leq upperb(I)_1$$

Ecuación 1

Es decir, *dst(I)* se establece a 255 (blanco puro) dentro de la matriz 2D que recibe como parámetro siempre que esté dentro de los valores umbral. Esto devuelve una máscara en blanco y negro, que será objeto de análisis en el siguiente paso.

Posteriormente, para asegurar unos contornos suaves y libres de agujeros internos en la medida de lo posible, se realiza un filtro de mediana (`Imgproc.medianBlur`). Este filtro consiste en ordenar los valores en la vecindad de cada píxel de menor a mayor y seleccionar el valor en la posición intermedia (mediana).Con ello, se consigue redondear las esquinas de los contornos y perder detalle en aquellas zonas que son de menor tamaño hasta conseguir que desaparezcan.

Tras este proceso puede ocurrir que se obtenga más de un contorno. Por las características de interacción que se proponen en este trabajo, la toma de decisión del contorno que favorablemente es una mano consiste en coger aquél con mayor área, ya que la disposición corporal hace que la mano sea un elemento más cercano a la cámara que, por ejemplo, el rostro u otras partes del cuerpo, en situaciones habituales.

Figura 6: imagen binarizada y suavizada

```
public static Mat deteccionDePiel(Mat orig, Scalar s, int margen) {
	Mat hsv = new Mat();
	Imgproc.cvtColor(orig, hsv, Imgproc.COLOR_BGR2HSV);

	// Buscamos limites inferior y superior para binarizar
	Scalar lInferior = new Scalar(s.val[0] - margen, s.val[1] - margen, s.val[2] -
margen);
	Scalar lSuperior = new Scalar(s.val[0] + margen, s.val[1] + margen, s.val[2] +
margen);

	// Binarizacion entre dos limites
	Core.inRange(hsv, lInferior, lSuperior, hsv);

	// Suavizamos y quitamos ruido
	Imgproc.medianBlur(hsv, hsv, 7);

	return hsv;
}
```

Una vez se tiene identificado y suavizado el contorno con el que se va a trabajar, un primer alcance a la forma de la mano consiste en hallar el polígono que aproxima el contorno de la misma. Esta operación es importante, dado que los vértices de esta figura serán futuramente los candidatos a los extremos de los dedos. Además, se obtiene el rectángulo que circunscribe el contorno, que ayudará en análisis posteriores. OpenCV dispone de la función `Imgproc.approxPolyDP` que hace uso del **algoritmo de Douglas-Peucker** [2] [10] cuya finalidad es utilizar el menor número de puntos para aproximar de la forma más fiel una curva. Es un algoritmo que se basa en la técnica de divide y vencerás. A continuación, se introduce una breve especificación del mismo para un mejor entendimiento de esta técnica:

Partiendo de una ruta inicial formada por un conjunto de $\boldsymbol{n}$ puntos, $\boldsymbol{p_0}$ a $\boldsymbol{p_{n-1}}$ se busca, de entre los puntos intermedios, el que esté más alejado de la recta que une p_0 y $\boldsymbol{p_{n-1}}$. Sea este punto $\boldsymbol{p_i}$ y $\boldsymbol{d(i,\ 0,\ n-1)}$ la distancia entre $\boldsymbol{p_i}$ y la recta que une $\boldsymbol{p_0}$ con $\boldsymbol{p_{n-1}}$:

- Si $\boldsymbol{d(i,\ 0,\ n-1)}$ supera un valor umbral que llamaremos $\boldsymbol{\varepsilon}$, se realizan dos llamadas recursivas, una que ejecuta el algoritmo entre los puntos $\boldsymbol{p_0}$ y $\boldsymbol{p_i}$ y otra que ejecuta el algoritmo entre los puntos $\boldsymbol{p_i}$ y $\boldsymbol{p_{n-1}}$.
- Si $\boldsymbol{d(i,\ 0,\ n-1)}$ no supera el valor umbral de $\boldsymbol{\varepsilon}$, se descartan de la ruta final todos los puntos intermedios entre $\boldsymbol{p_0}$ y $\boldsymbol{p_{n-1}}$, es decir, se descartan todos los puntos $\boldsymbol{p_j}$ tales que $i < j < n-1$.

Nótese que es necesario probar con diferentes $\boldsymbol{\varepsilon}$, para ajustar a las necesidades de la imagen (la mano en este caso). Con ello, encuentra el polígono con menos vértices posibles cuya distancia es menor o igual que dicho umbral.

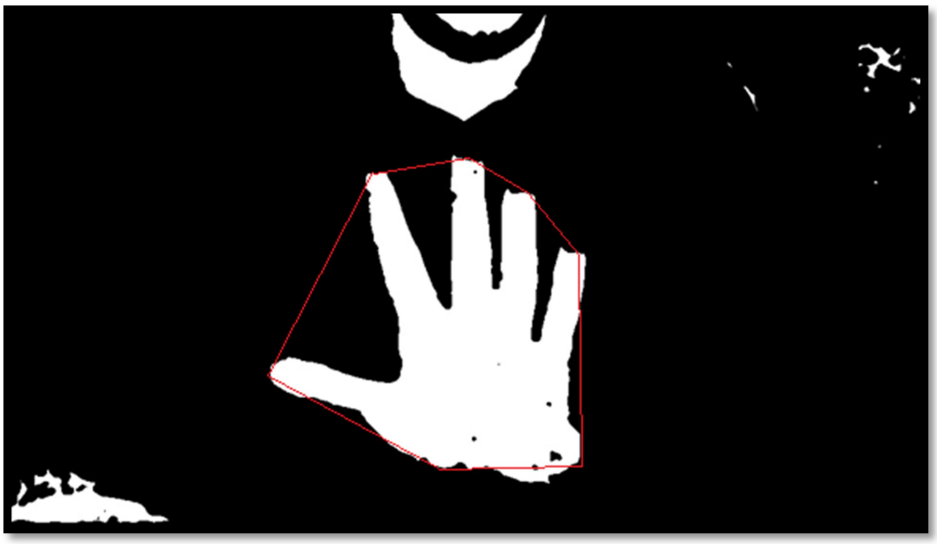

Figura 7: polígono obtenido por el algoritmo Douglas-Peucker

```
Imgproc.approxPolyDP(contorno2f, approxContour2f, 30, true);
```

El siguiente paso consiste en localizar todos los puntos convexos del contorno, apoyándonos en el polígono que hemos obtenido previamente. Un defecto convexo consiste en una cavidad en un contorno que se localiza por fuera del mismo. Es decir, el área que pertenece al polígono circundante hallado previamente pero que no pertenece al contorno. A partir de ahí, los puntos convexos son aquellos que están más alejados

del polígono. OpenCV utiliza el **algoritmo de Sklansky** (1982) [9] para localizarlos, que tiene un coste demostrado $n \log(n)$.

Esencialmente, el algoritmo de Sklansky recibe una lista circular de puntos y la recorre en una cola, en sentido anti horario, empezando por el vértice de menor coordenada. En una pila se van añadiendo los vértices que son giros a la izquierda y desapilando (eliminando la cima) en los vértices que son giros a la derecha, a medida que se van consumiendo los puntos de la cola previamente mencionada.

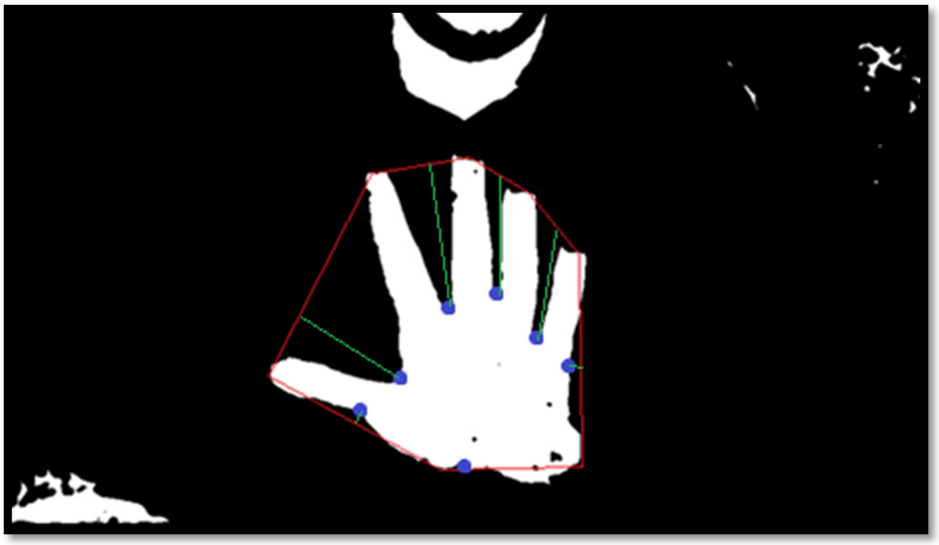

Figura 8: Puntos y defectos convexos del contorno

```
Imgproc.convexityDefects(contornoMasGrande, hull, matDefectosConvexos);
```

Para localizar los dedos que estuvieran extendidos, el siguiente paso consiste en analizar los puntos convexos más alejados de cada par de vértices vecinos. Cada defecto convexo, como se aprecia en la figura 9, devuelve cuatro valores que son relevantes para ello: el punto de inicio (*pStart*), el punto más alejado (*pFar*) y el punto final (*pEnd*), más la profundidad de dicho defecto (o distancia desde *pFar* hasta el polígono).

Por tanto, OpenCV devuelve una lista de arrays accesible como matriz, donde se puede encontrar, para cada uno de los defectos convexos que el algoritmo ha encontrado, los puntos que lo definen, así como la profundidad del mismo. Estos valores son esenciales y se utilizarán para realizar el análisis estructural de la mano y definir los dedos, pues en este caso puede haber oquedades propias de la forma de la mano que no sean necesariamente el espacio entre los dedos.

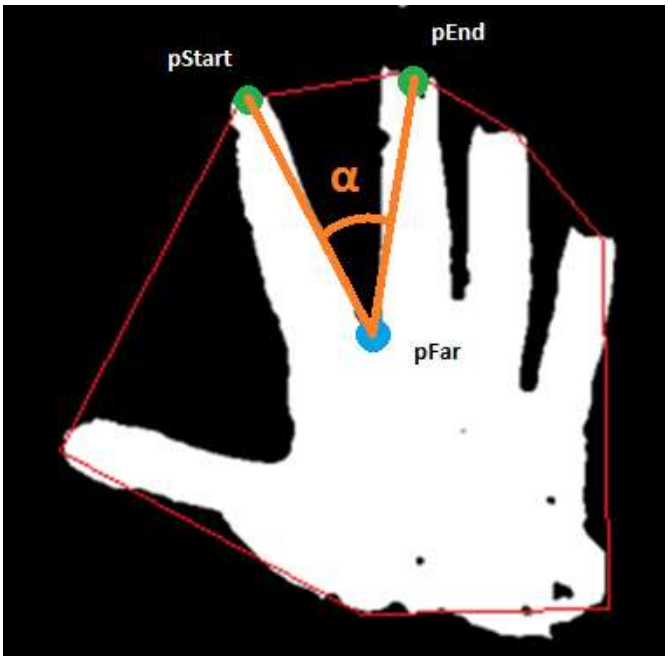

Figura 9: Puntos de un defecto convexo

A continuación se va a llevar a cabo un análisis de los puntos mencionados sobre los defectos convexos. Para ello se establecen algunos umbrales, como la distancia entre el punto convexo (*pFar*) y sus vértices (*pStart* y *pEnd*), y el ángulo (α) que forman entre los tres. El valor de este último ha dado buenos resultados con un umbral superior de 95º. Además, se ha de tener en cuenta que estos puntos estén a una cierta distancia de los bordes del rectángulo que circunscribe la mano, con el fin de evitar falsos positivos. Tras este análisis se guardarán las tuplas o defectos convexos que se hayan encontrado siguiendo los criterios mencionados: dos vértices y el punto convexo más alejado de ellos. Este conjunto contiene todos los extremos de los dedos y los puntos de unión entre dedo y dedo.

Figura 10: defectos convexos de la mano

```
private static ArrayList<int[]> analizaContornos(Mat camImage, List<Point> ptsContornoMasGrande, Rect
boundingRect, MatOfInt4 matDefectosConvexos) {

	int umbralProfundidad = 5;
	double numDefectos = matDefectosConvexos.size().height;
	ArrayList<int[]> defectosFinales = new ArrayList<int[]>();
	for (int i = 0; i < numDefectos; i++) {
		int[] defectoConvexo = new int[4];
		matDefectosConvexos.get(i, 0, defectoConvexo);
		if (defectoConvexo[3] / 256 > umbralProfundidad) {
			int tolerancia = boundingRect.height / 5;
			float anguloTolerancia = 95;
			Point pStart = ptsContornoMasGrande.get(defectoConvexo[0]);
			Point pEnd = ptsContornoMasGrande.get(defectoConvexo[1]);
			Point pFar = ptsContornoMasGrande.get(defectoConvexo[2]);

			if (distanciaEntrePuntos(pStart, pFar) > tolerancia && distanciaEntrePuntos(pEnd,
pFar) > tolerancia && getAngulo(pStart, pFar, pEnd) < anguloTolerancia) {
				if (pEnd.y > (boundingRect.y + boundingRect.height - boundingRect.height
/ 4)) {
				} else if (pStart.y > (boundingRect.y + boundingRect.height -
boundingRect.height / 4)) {
				} else {

					// Segundo descarte
					if (pFar.x > 0 && pFar.x < 1024 && pFar.y > 0 && pFar.y < 768) {
						defectosFinales.add(defectoConvexo);
						if(debugMode) Imgproc.circle(camImage, pFar, 6, new
Scalar(255, 255, 255));
					}

				}
			}

		}
	}
	return defectosFinales;
}
```

Aunque la técnica elegida para este trabajo ha sido la mencionada, no ha sido la única valorada. Otra aproximación consiste en combinar los defectos convexos vecinos, calculando el ángulo entre los puntos convexos de ambos y el vértice común que forman. Traducido a la mano, se trata del ángulo entre el extremo del dedo y los dos puntos de la base del mismo que definen su anchura. A pesar de que esta medida puede ser buena dado que es una morfología única sobre la mano, los resultados no fueron los esperados. Por ese motivo, se siguió con la técnica original.

Llegando a este punto, restaría identificar los dedos dentro del mencionado conjunto, donde nos quedaríamos con los vértices del polígono que coinciden con el extremo de un dedo. Con esta información, podemos determinar mediante la proporción del rectángulo que circunscribe el contorno y el número de dedos detectado si efectivamente hemos encontrado una mano o no.

```
/**
 * Analiza los defectos desde el punto de inicio del mismo, el mas profundo y en final (pStart,
pFar, pEnd)
 *
 * @param defectos
 * @param puntosContorno
 * @param boundingRect
 * @param camImage
 * @param hull
```

```
 * @return ArrayList de puntos finales de los dedos
 */
private static ArrayList<Point> getDedos(ArrayList<int[]> defectos, List<Point> puntosContorno,
Rect boundingRect, Mat camImage, MatOfInt hull) {
        ArrayList<Point> dedos = new ArrayList<Point>();
        for (int i = 0; i < defectos.size(); i++) {
                Point pStart = puntosContorno.get(defectos.get(i)[0]);
                Point pEnd = puntosContorno.get(defectos.get(i)[1]);

                if (i == 0) {
                        dedos.add(pStart);
                        dedos.add(pEnd);
                } else {
                        dedos.add(pEnd);
                }
        }
        if (dedos.size() == 0) {
                dedos = getUnDedo(defectos, puntosContorno, boundingRect, camImage, hull);
        }

        return dedos;
}
```

Para localizar el centro de la palma de la mano, la mejor aproximación encontrada consiste en localizar el centro de la circunferencia circundante entre los puntos convexos previamente encontrados y el punto medio de la base del rectángulo que circunscribe el contorno.

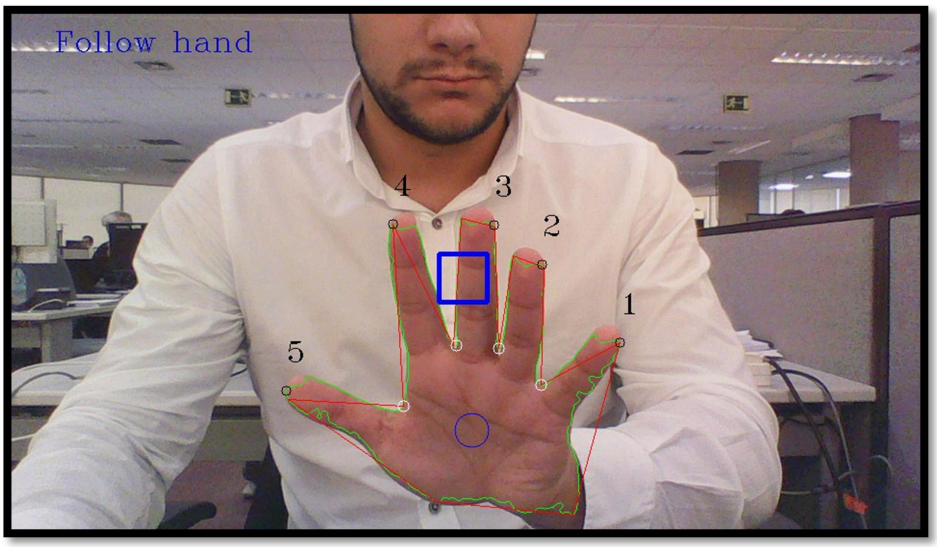

Figura 11: detección de dedos y centro de la palma

Una vez calculado el centro de la mano (representado en azul) y, para hacer seguimiento de los posibles cambios de exposición en la imagen, tomamos este punto de referencia y cogemos una muestra del rectángulo que rodea este punto, para así renovar la media del color de la piel, que se usará para detectar la mano el siguiente frame.

```java
/**
 * Dibuja el círculo mínimo que contiene a los puntos de convexion y el punto medio de la base
del rectangulo que
 * incluye la mano
 * @param imagen
 * @param puntosContorno
 * @param defectos
 * @param boundingRect
 * @return el centro de la mano
 */
public static Point centroMano(Mat imagen, List<Point> puntosContorno, ArrayList<int[]>
defectos, Rect boundingRect) {

        ArrayList<Point> puntos = new ArrayList<Point>();

        for (int i = 0; i < defectos.size(); i++) {
                Point pFar = puntosContorno.get(defectos.get(i)[2]);
                puntos.add(pFar);
        }

        if(boundingRect != null) {
                Point p = getMedioBaseRect(boundingRect);
                puntos.add(p);
        }

        MatOfPoint2f pr = new MatOfPoint2f();
        Point center = new Point();
        float[] radius = new float[1];
        pr.create((int) (puntos.size()), 1, CvType.CV_32S);
        pr.fromList(puntos);

        if (pr.size().height > 0) {
                Imgproc.minEnclosingCircle(pr, center, radius);

                if(debugMode) Imgproc.circle(imagen, center,(int) radius[0]/5, new
Scalar(255,0,0));
        }
        return center;

}
```

2.2.2. Detección del gesto

Una vez detectado que hay una mano en la imagen y teniendo información sobre los dedos y el centro de la palma, se pueden establecer determinados criterios sobre los que actuar al encontrar una disposición geométrica concreta.

En casos básicos se puede contabilizar:

- El número de dedos.
- La posición de la mano en la pantalla.
- El ángulo de inclinación de los dedos respecto del eje vertical.
- El ángulo de inclinación de los dedos respecto del centro de la mano (si la cámara no está alineada con el horizonte).
- La distancia entre los dedos más alejados.
- El área total de la mano.
- Por supuesto, una combinación de estos parámetros.

Elevando la complejidad, se puede tener en cuenta una secuencia de todas estas características con respecto del tiempo, utilizando un búfer. Es decir, un gesto consistiría en la detección consecutiva de diferentes posiciones de la mano. Un ejemplo puede ser el gesto de un saludo, que alternaría inclinaciones sobre la palma de la mano que, tras reconocer tres cambios de inclinación, aceptaría la secuencia.

```
if (centroMano.x < camImage.width() / 3) {
        side = "Left";
} else if (centroMano.x > camImage.width() * 2 / 3) {
        side = "Right";
}
```

```
switch (numDedos) {
        case 2: action = "Take photo";
                break;
        case 3: action = "Circle";
                break;
        case 4: action = "Hover";
                break;
        case 5: action = "Follow hand";
        break;
}
```

Ejemplos básicos de análisis de la información obtenida

3. Prueba de concepto

3.1. Tecnologías utilizadas

En esta sección se enumeran y detallan cada una de las tecnologías utilizadas en la implementación de la prueba de concepto de detección gestual, quedando fuera del alcance de este trabajo algunos de los aspectos relativos a la Ingeniería de Software: casos de uso, configuración de proyecto, diseño de interfaz, etc.

3.1.1. Sistema operativo

A pesar de que las tecnologías a las que se ha recurrido son multiplataforma, se ha utilizado Windows 10 por ser la última versión estable del sistema operativo de Microsoft, en su arquitectura de 64 bits.

3.1.2. OpenCV versión 3.3.1

Se trata de un conjunto de librerías desarrolladas en C y C++ orientadas a la visión artificial, que cuenta con gran prestigio por la madurez y la eficiencia de las operaciones que permite realizar sobre una imagen, y la cantidad de sistemas que las han adoptado en diferentes campos de la percepción computacional.

Para el uso de OpenCV es necesaria la instalación y configuración de sus librerías en el entorno de ejecución. Existe documentación oficial al respecto.

3.1.3. Java 1.8

La prueba conceptual ha sido programada en Java en su totalidad, debido a que es compatible con el uso de las librerías de OpenCV y, dado el extenso uso de este lenguaje y sus múltiples frameworks, permite una alta adaptabilidad a diferentes tecnologías.

3.1.4. Java Swing

Java Swing es un paquete de Java que aporta una herramienta de diseño de interfaces gráficas de usuario, conocido usualmente como GUI, del inglés *graphical user interface*. Su sencillez, modularidad, y su extensa cantidad de componentes lo convierten en la herramienta ideal para desarrollar una interfaz escalable de manera ágil.

3.1.5. IDE

Para la implementación de esta prueba de concepto, como entorno de desarrollo integrado, se ha elegido utilizar Eclipse: un software de código abierto que proporciona una inmensa funcionalidad gracias a extensiones, en inglés plug-ins. Además de ser un

programa muy completo en cuanto a prestaciones se refiere, se ha convertido en el entorno de programación habitual del participante en este proyecto.

3.1.6. Tuleap

Tuleap es una aplicación web open source que permite la organización de proyectos de una manera sencilla. Destaca entre sus características la utilización de la metodología kanban, basada en la representación de los proyectos como columnas que contienen tres listas de tareas: pendientes, en progreso y completadas. Las listas contienen tarjetas, cada una de ellas representa una tarea y uno o varios usuarios que la deben realizar. Las tarjetas deben pasar de una lista a la siguiente según el estado en el que se encuentren.

3.2. Especificación de requisitos

Para llevar a cabo el estudio e implementación sobre esta técnica de visión artificial se propusieron una serie de requisitos que serían la base sobre la cual desarrollar la prueba de concepto. Muchos de ellos surgen orientados a la propia técnica de análisis de imagen que se ha llevado a cabo, mientras que otros son intrínsecos de la propia finalidad y el propósito que tiene el estudio que se propone.

3.2.1. Requisitos funcionales

i. El sistema debe **tomar como entrada una imagen** y reconocer en ella el gesto de una mano, si la hubiere, para devolver el comando como salida. Es la idea fundamental del desarrollo, para que pueda ser introducido en sistemas que no cuentan con una interfaz de usuario.

ii. La aplicación debe mostrar en una **interfaz gráfica** la imagen captada por la cámara para mejor comprensión del usuario. Para adaptar esta prueba de concepto y como única salida del sistema se requiere de este requisito para su estudio.

iii. Debe contar con una interfaz que muestre la **orden o gesto realizado** con la mano, si lo hubiere, tras realizar a cabo en análisis de la imagen.

iv. Se debe **dibujar toda la información** obtenida en el análisis. Esto ayudará a comprender los cálculos y el proceso llevado a cabo en la identificación de la mano, como la binarización, la obtención del contorno o los defectos convexos.

v. La **información de análisis** debe poder ser habilitada o deshabilitada para mostrar una imagen limpia. Para ello se establecerá un botón dedicado que oculte bajo necesidad las líneas trazadas sobre la imagen captada por la cámara.

vi. Por las limitaciones de la cámara, se debe contar con un **selector de color** de piel, es decir, una zona dedicada de la imagen captada por la cámara que debe ser cubierta por el color de piel que se desee seleccionar.

vii. Debe haber un botón que acompañe a la zona del selector de color que, al ser accionado, obtenga la media de color del interior del recuadro para comenzar la búsqueda de la mano del usuario.

viii. El sistema debe hacer **seguimiento del color** capturado para mitigar posibles cambios en las condiciones lumínicas de la escena.

3.2.2. Requisitos no funcionales

i. El sistema debe tener un **rendimiento** suficiente para llevar a cabo un análisis en tiempo real. Es fundamental dado que la finalidad es utilizar este desarrollo como herramienta de integración con la interacción humana, por lo que no debe haber un gran retardo entre la acción del usuario y la respuesta que éste espera del sistema.

ii. Debe ser posible ejecutar la aplicación en **diferentes plataformas** Windows o Linux. Esto amplía el espectro potencial de dispositivos que pueden adoptar esta tecnología, siendo Linux una de las plataformas más comunes en sistemas empotrados.

iii. La cantidad de **recursos** consumidos por la aplicación debe ser la menor posible dado el carácter modular del software, que potencialmente puede ser utilizado en procesadores de alta y baja potencia.

iv. La **interfaz de usuario** de la prueba de concepto debe ser sencilla, de fácil uso y comprensión para un usuario, ya que el objeto de estudio es la técnica a utilizar, y el análisis de la imagen debe ser por ello visualmente simple.

3.3. Ejecución

3.3.1. Entorno

Para el entorno de ejecución de la prueba de concepto se ha utilizado Windows 10. Las librerías de OpenCV 3.3.1 están instaladas e integradas en la aplicación Java. Esta aplicación muestra una interfaz simple donde destacan tres elementos:

- **Rectángulo de selección de color:** se representa en azul, en el dentro del cuadro de imagen.
- **Botón "Pick colour":** este botón toma una muestra de la media de color que hay en el interior del rectángulo antes mencionado. Es un paso fundamental en esta prueba de concepto.
- **Botón "Debug":** sirve para mostrar el dibujo de los contornos y puntos convexos con los que se trabaja en las diferentes fases del análisis de la imagen, mostrando también la localización de los dedos y el número de los mismos.
- **Comando:** sobre el cuadro de imagen se dibuja el comando que corresponde al gesto que se está realizando, si lo hubiere.

Se ha elegido un proceso de selección de color manual para poder realizar diferentes pruebas, como pueden ser usando guantes, colocando un sujeto con la piel más clara o más oscura.

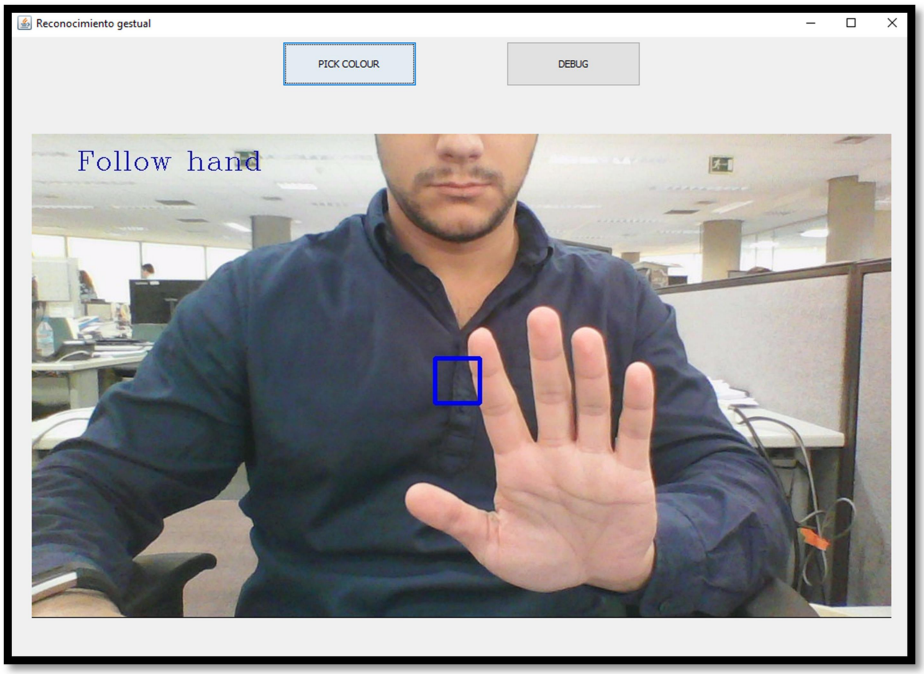

Figura 12

3.3.2. Selección de color

Al inicializar la aplicación, se toma una muestra del color que hay en el medio de la imagen, es decir, el interior del rectángulo azul. En el siguiente ejemplo, se ve cómo ha seleccionado el color de la camisa. Es por esto que selecciona la región que corresponde a este color y no muestra ninguna información, dado que se reconoce una mano. En la siguiente imagen se aprecia este suceso, activado el modo debug:

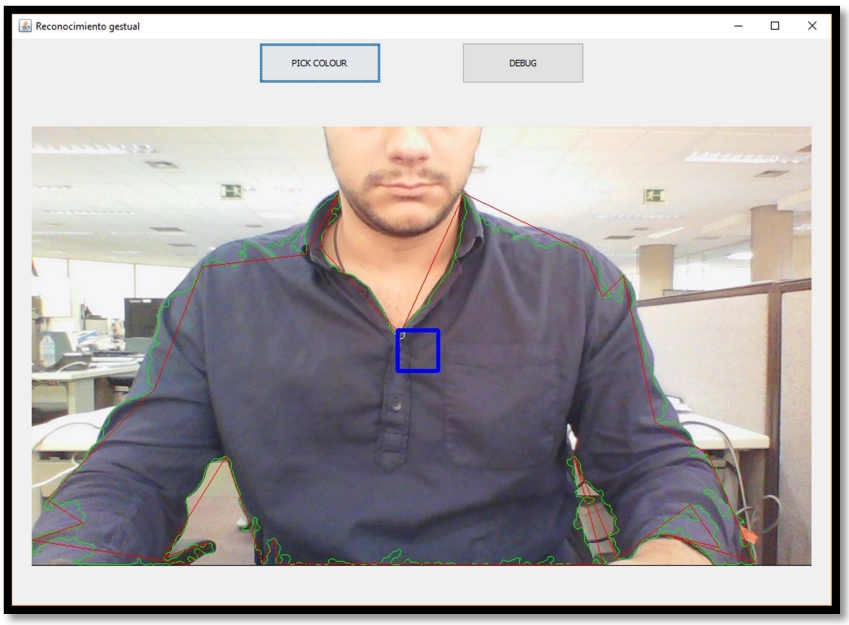

Figura 13: binarización sin selección manual

Por ello, para comenzar, se debe colocar la mano tapando el rectángulo azul central, y pulsar el botón "Pick colour":

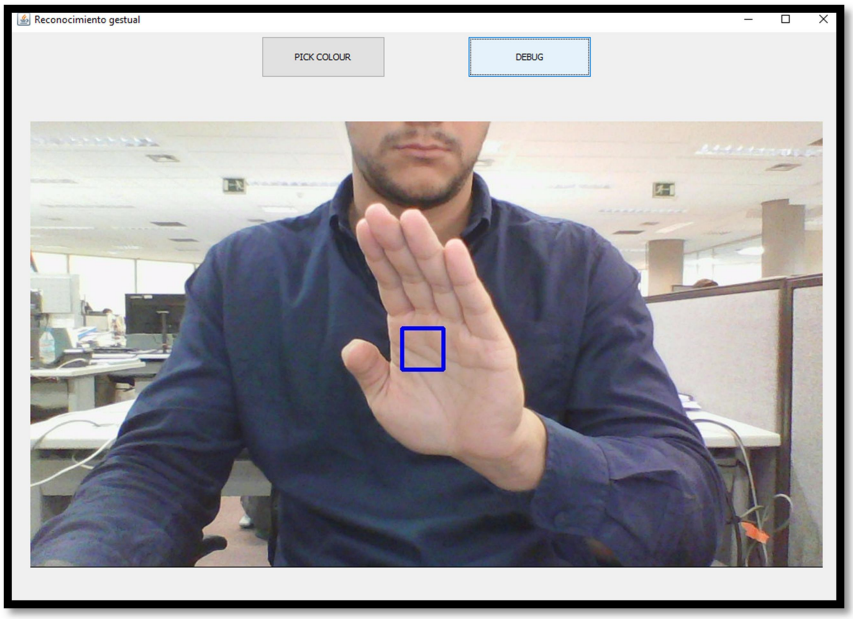

Figura 14: recubrimiento del selector de color con la mano

Automáticamente la aplicación binariza la piel utilizando el color medio del interior del rectángulo y comienza el proceso de análisis, mostrando en la esquina superior izquierda el gesto que se está realizando, si lo hay:

Figura 15

3.3.3. Modo debug

Para mostrar las líneas de los contornos, vértices, defectos convexos, dedos reconocidos y el centro de la palma de la mano, hay que pulsar sobre el botón "Debug". Al volver a pulsar, las líneas dejan de verse, mostrando el cuadro de imagen limpio.

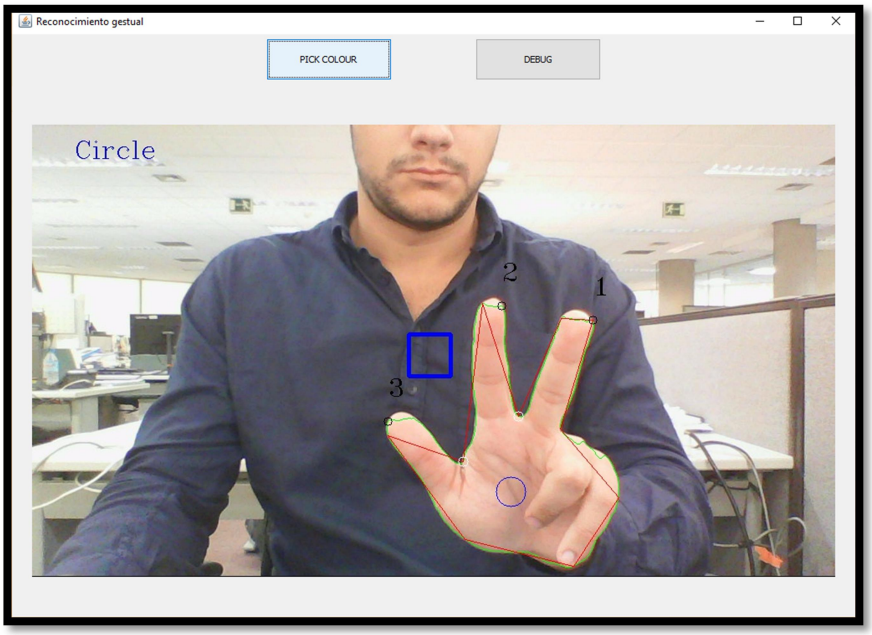

Figura 16: modo "debug" activo, mostrando todos los trazos

3.3.4. Comandos

Utilizando información geomátrica de la disposición y el número de dedos, se puede reconocer y disponer como salida un comando en particular previamente definido. Para este ejemplo, se han propuesto comandos que podrían ser realizables por un dron o aeronave no tripulada:

- **Tomar una foto (Take photo):**

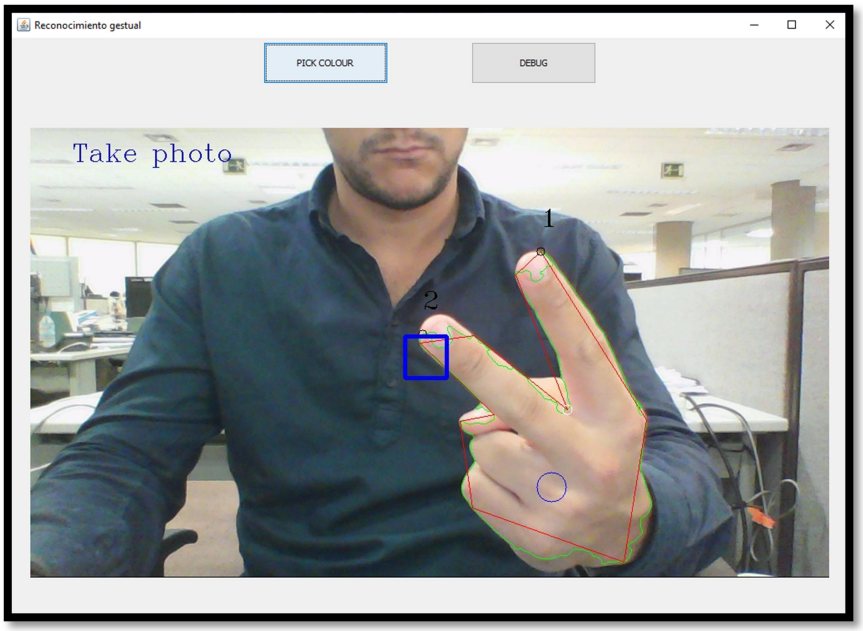

Figura 17

- **Orbitar en círculo (Circle):**

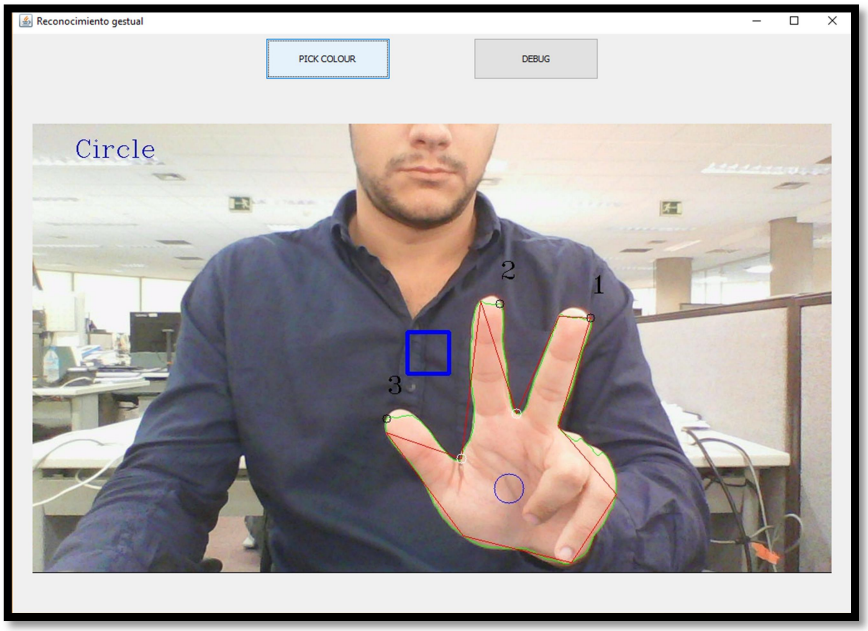

Figura 18

- **Orbitación estacionaria (Hover):**

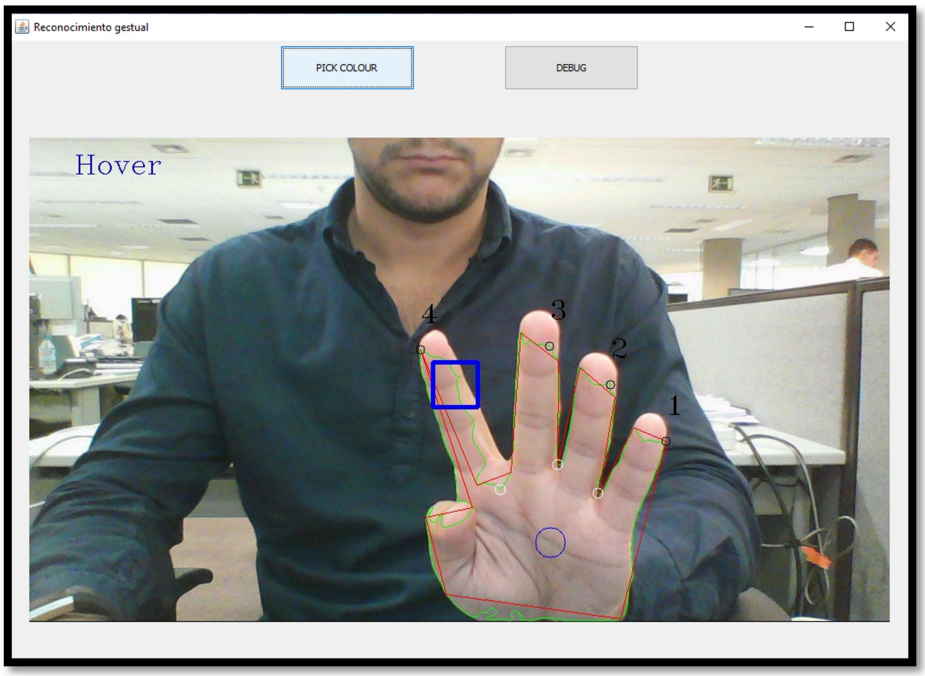

Figura 19

Originalmente y pensando en un vehículo aéreo no tripulado, la función de detenerse es fundamental y por ello es una solución natural realizar el gesto de la figura 19 que podría acompañarse de un gesto hacia abajo para que el vehículo aterrizase en el suelo.

En este comando se podría valorar también la rotación de la mano, utilizando la posición de los extremos de los dedos respecto de la palma de la mano, ya que un gesto natural pueden ser cuatro dedos desplegados lateralmente, como un saludo, detención, indicación, etc.

3.3.5. Seguimiento de la mano

También se tiene en cuenta la posición relativa de la mano para realizar un seguimiento de la misma. Esto puede servir para reposicionar la cámara de un dron de manera que apunte siempre a la mano y dé continuidad a la interacción con el sistema.

Podría servir de la misma manera para cualquiera de los sistemas que utilicen esta tecnología, siempre y cuando cuenten con una cámara robotizada que permita la rotación vertical y horizontal.

- **Mano en el lado izquierdo (Follow hand - Left):** la cámara debe reposicionarse girando a la izquierda.

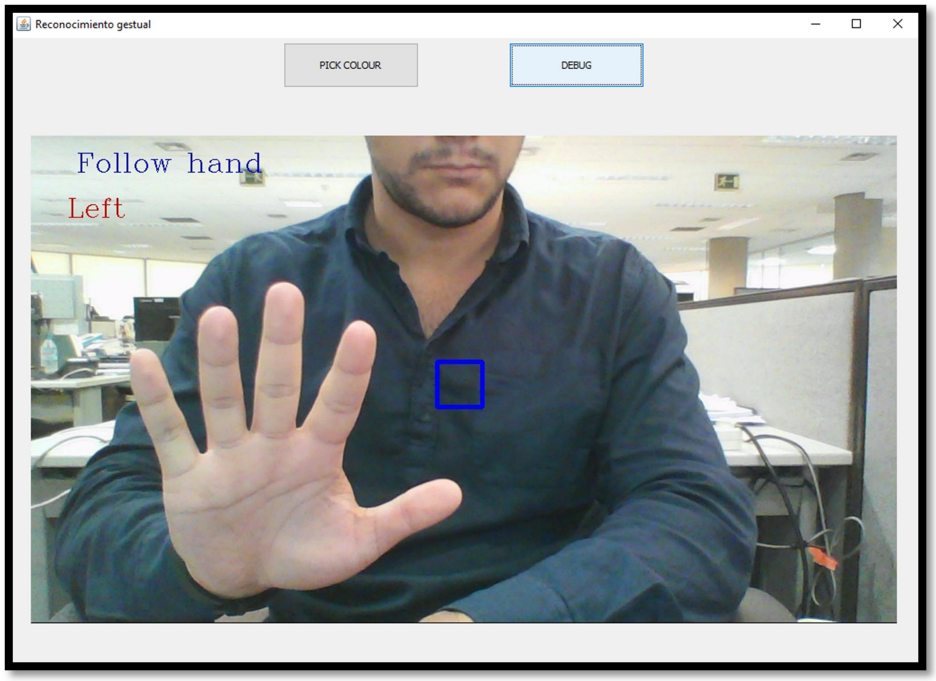

Figura 20: comando "seguir mano", rotar a la izquierda

- **Mano en el lado derecho (Follow hand - Right):** la cámara debe reposicionarse girando hacia la derecha, para centrar la mano en la imagen.

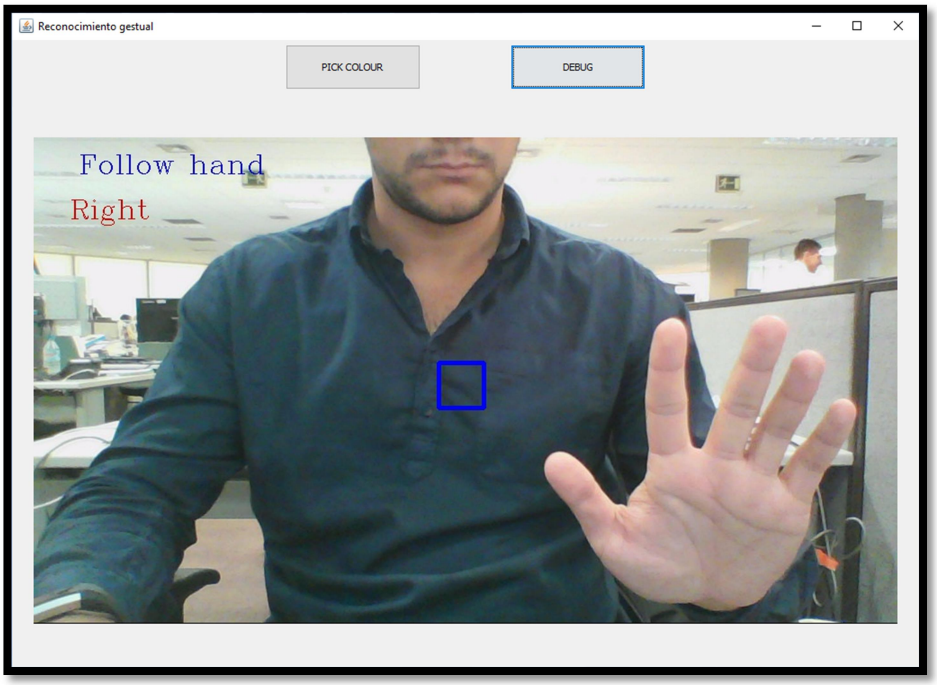

Figura 21: comando "seguir mano", rotar a la derecha

3.4. Resultados

Tras analizar numerosas ejecuciones con diferentes umbrales y entornos, se pueden extraer varios resultados sobre la prueba de concepto realizada:

- **Las condiciones del entorno** afectan en gran medida al éxito de la detección gestual. Son numerosos los factores que pueden interferir: una piel más oscura o más pálida, el contraste con la ropa, las sombras, los colores de la escena, vestir de manga corta...

- **La calidad de la cámara** afecta por tres factores: el control automático de exposición, el rango dinámico y la resolución. El primero es habitual en las cámaras integradas y generalmente no es bloqueable, por lo que cambia el valor de la exposición dependiendo del promedio de intensidad de la imagen obtenida. Por ello, si existen cambios en escena (evidentes por el movimiento de la mano o de diferentes sujetos) la exposición de la imagen varía (figura 22-A sobreexpuesta y 22-B subexpuesta) y con ello también puede variar el equilibrio de blancos o la tonalidad del color (figura 22-C con tonalidad verde-cálida y figura 22-D con tonalidad magenta-fría) que se ha tomado como muestra para la mano, obligando a tomar una nueva muestra.

Figura 22: Diferentes exposiciones y equilibrios de blancos

El rango dinámico también es relevante, siendo generalmente bajo en cámaras integradas en dispositivos, ya que pierde detalles en las sombras en favor del color negro, y perdiendo detalles en las altas luces en favor del color blanco puro, afectando al contorno de la mano en condiciones en las que la luz es menos homogénea. La figura 23-A muestra una imagen con un rango dinámico

medio-bajo, dado que apenas hay detalle en las altas luces ni en las sombras. Por lo contrario, la figura 23-B muestra una imagen con un rango dinámico mayor dado que no satura lo mencionado anteriormente hacia blancos o negros puros, teniendo así detalle sobre toda la imagen y un histograma bastante completo.

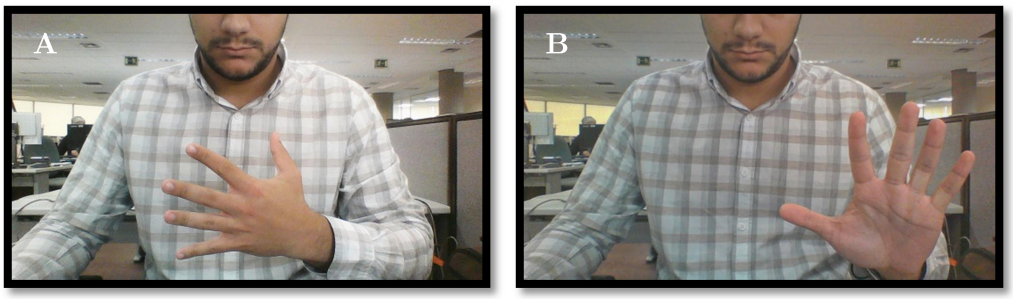

Figura 23: Diferentes exposiciones y equilibrios de blancos

En lo relativo a la resolución, cuanto mayor es la resolución más costosas son las operaciones de cálculo dado que la matriz sobre la que se realizan es más grande, si bien es cierto que la cantidad de información disponible es mayor y esto garantiza mejores resultados.

- La **binarización** de la imagen sobre un **color** en lugar de sobre un valor de intensidad de gris aísla de forma más efectiva la región de interés dado que descarta zonas que podrían aportar la misma intensidad sin ser piel, que es el elemento de búsqueda principal.

- Para afrontar este problema de cambio de exposición y condiciones de luz, ha sido una **buena solución recalcular la media de color de la piel en el centro de la palma de la mano,** siempre que se detecta una mano en la imagen. Este parámetro se utilizará para encontrar la mano en el siguiente frame o cuadro de imagen. De esta forma se realiza un seguimiento de los cambios de luz que no sean extremadamente repentinos o acusados, sin perder la figura de la mano.

- El **método utilizado en la obtención de los dedos** desfavorece la obtención de un único dedo. Fueron estimadas diferentes aproximaciones para ello, como calcular el ángulo entre el vértice y los puntos convexos vecinos (operación contraria a la que finalmente se ha adoptado). Todas ellas mostraron ventajas e inconvenientes. Finalmente la elegida muestra tal defecto, que probablemente

requiera de otra técnica de apoyo que haga la detección más precisa sin elevar demasiado el coste operacional.

- **Utilizar un color uniforme en la mano** que contraste con el resto de la imagen favorece el éxito de la detección, ya que el contorno es más identificable y con ello los puntos sobre los que se lleva a cabo el análisis se aproximan mejor a la realidad. Para comprobar esto, se puede utilizar un guante, por ejemplo.

- **Detectar los gestos con configuraciones geométricas** totalmente **diferentes** o disjuntas hacen que el sistema de detección sea fiable y muestre el comando elegido de forma continua sin mostrar imprecisiones decidiendo entre un comando u otro.

- El uso de **OpenCV** simplifica el número de líneas de código y cálculos a realizar debido a su extensa librería con múltiples operaciones, abstrayendo al desarrollador de la teoría y las operaciones matemáticas que hay detrás de todas ellas. Además, estas operaciones son resueltas de forma eficiente permitiendo el análisis en tiempo real tras décadas de optimización.

- La adopción de **la mano como elemento de entrada** gestual al sistema sobre otras características del cuerpo es una decisión acertada dada la forma geométrica variable de la misma, por encima del rostro. Esto hace más sencillo analizar e identificar sus características.

- Esta técnica no **requiere de entrenamiento ni clasificación**, siendo positivo en este aspecto dado que son tareas que requieren de tiempo y no garantizan siempre los mismos resultados. Sin embargo, se puede considerar una posible ampliación mediante técnicas sobre redes neuronales para entornos hardware que cuenten con potencia suficiente de procesamiento.

- **En general** se pueden conseguir **buenos resultados** teniendo en cuenta que la única información sobre la que trabaja el sistema es una matriz de puntos, es decir, no existe ningún otro elemento de apoyo o entrada al sistema que se pueda utilizar para orientar la búsqueda de la mano o descartar otros objetos.

4. Conclusión y futuras líneas

El desarrollo realizado resuelve de una manera sencilla y poco costosa el problema del reconocimiento gestual de la mano. En este punto es importante la sencillez, porque la finalidad es deducir hasta qué punto es una tecnología alcanzable desde los desarrollos más modestos, donde son necesarios la eficiencia y el bajo coste.

Para la selección del color de la piel del usuario, podría realizarse un proceso de reconocimiento facial que, una vez detectara el rostro, tomara una muestra del color de la tez, y de esta manera apoyase al proceso de detección de la mano ante posibles cambios de luz cuando el usuario no estuviera mostrándola a la cámara.

Los resultados obtenidos en general son positivos, por lo que se deduce que es una técnica suficiente para un análisis de primer nivel. Múltiples factores pueden afectar al resultado final, pero algunos de ellos se pueden afinar para lograr un máximo rendimiento. Como se ha visto en los resultados, un control manual de exposición, unas condiciones de luminosidad constantes, un contraste alto de imagen, si pueden ser controlados, aportan fiabilidad al conjunto.

Además de esto, se pueden presentar segundos niveles de análisis que mejoren la precisión del sistema utilizando otras tecnologías. Una de las opciones es utilizar una cámara dual o estéreo para que, combinando las imágenes, se obtenga información que relacionar entre ellas para una mejor identificación. También se pueden incorporar sensores de profundidad, que suelen constar de un emisor IR junto con otra cámara IR. Con ello, dado que la mano es más próxima que el resto del cuerpo, se puede cruzar la información de la imagen con estos datos para garantizar que la mano haya sido bien detectada. Ésta es, de hecho, la tecnología usada en el dron que se introduce en la motivación de este trabajo.

En lo referente a gestos, una vez detectada la morfología de la mano hay amplitud de posibilidades, más aún si cabe introduciendo las secuencias de gestos, a pesar de los problemas ya mencionados. En este ámbito el uso de un segundo nivel de análisis no solo aporta fiabilidad y precisión, sino que también puede aportar otra dimensión a la hora de realizar dichos gestos, dado que al medir la profundidad se puede representar el gesto en el espacio en lugar de únicamente dos dimensiones.

La interacción humana con sistemas tecnológicos puede aprovechar este sistema de entrada como una vía alternativa válida, donde no es necesario tocar para ejecutar órdenes y comandos e, incluso, puede marcar una diferencia en términos de accesibilidad. Aquellas personas con discapacidades pueden adaptar un gesto que realice una combinación de tareas sobre un sistema que, de otra forma, no podrían llevar a cabo.

En los sistemas de interacción más básicos, se podría utilizar la mano como un ratón de ordenador, u otras aplicaciones como videojuegos, presentación de diapositivas, e incluso aplicaciones de realidad aumentada. Sin embargo, en lo referente a la motivación de este trabajo, también se podría aplicar a un dron de diseño propio. Por ejemplo, se podría incorporar como módulo en un sistema PX4, un proyecto de hardware y software libre para drones o UAVs que está incluido en otros proyectos más grandes como Dronecode [3] o FlytOS. Esto podría ampliar las funcionalidades de un dron que incorporase este firmware para emular el vuelo gestual del DJI Spark. Con ello, se permitiría el uso de la tecnología de control gestual a un bajo coste, resolviendo el reto computacional que supone el análisis en tiempo real de la forma de la mano.

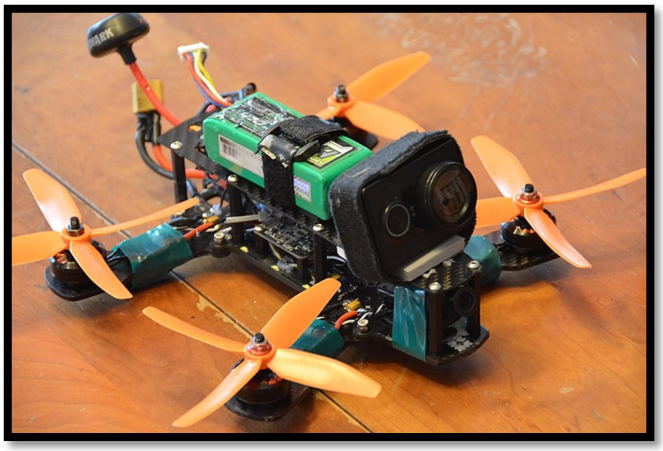

Figura 24: dron casero o de fabricación por componentes

Cambiando de ámbito completamente, y siguiendo otra motivación de carácter humanitario y dirigida hacia la accesibilidad, esta técnica de reconocimiento gestual puede ser utilizada para traducir en tiempo real el **lenguaje de signos**, de manera que personas que parecen sordera y se comunican en este lenguaje puedan ser entendidas por gente que no lo domine.

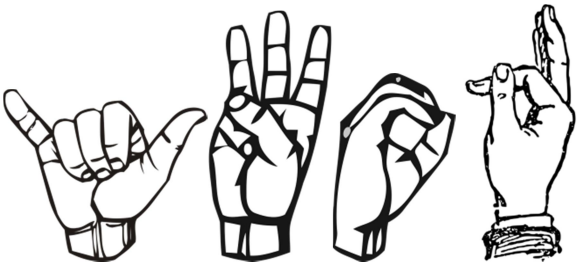

Figura 25: algunos gestos potencialmente reconocibles por la técnica propuesta

Para ello, bastaría con apuntar al hablante con la cámara y que el sistema formase las palabras según fuera reconociendo gestos, mostrándolos por pantalla o incluso reproduciéndolas mediante síntesis de voz. Este reto es probablemente mayor dado que el lenguaje de signos está establecido y no se pueden adaptar de una forma conveniente para mitigar las limitaciones del reconocimiento gestual, pero mediante el uso de análisis de segundo nivel u otros dispositivos de ayuda al análisis que se han comentado en este trabajo podría ser algo alcanzable y que mejoraría la calidad de vida de personas con dificultades auditivas y su entorno.

Todo hace indicar que cada vez más empresas y estudios tecnológicos comenzarán a apostar por esta tecnología, pues es claro que se disponen de múltiples soluciones para resolver el problema de una forma ágil, y las posibilidades que aporta justifican su implementación en diferentes sistemas de interacción humana, especialmente si se enfoca desde un objetivo con carácter humanitario donde se logre hacer más fácil la vida cotidiana de mucha gente.

5. Bibliografía

[1] Aashni Haria, Archanasri Subramanian, Nivedhitha Asokkumar, Shristi Poddar, Jyothi S Nayak. Hand Gesture Recognition for Human Computer Interaction. Procedia Computer Science, Volume 115, 2017, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.09.092. (http://www.sciencedirect.com/science/article/pii/S1877050917319130)

[2] Blog de Avelino Herrera Morales. Algoritmo de Ramer-Douglas-Peucker de simplificación de rutas. (2018). [online] Disponible en http://avelino.atlantes.org/blog/index.php?entry=entry140307-095351 [Fecha de acceso: 5 de junio de 2018].

[3] Caputo, M., Denker, K., Dums, B., & Umlauf, G. (2012). 3D Hand Gesture Recognition Based on Sensor Fusion of Commodity Hardware. In mensch & Computer (Vol. 2012, pp. 293-302).

(http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.663.1235&rep=rep1&type=pdf)

[4] GitHub. (2018). Dronecode. [online] Disponible en: https://github.com/Dronecode/ [Fecha de acceso: 5 Jun. 2018].

[5] Ippolito, G. (2018). Open Source Drones. [online] Yolinux.com. Disponible en http://www.yolinux.com/TUTORIALS/Drones.html#FLIGHT_SOFTWARE [Fecha de acceso: 5 Jun. 2018].

[6] Itseez (2018). OpenCV API Reference. [online] Disponible en: https://docs.opencv.org/3.0-beta/index.html [Fecha de acceso: 6 Jun. 2018]

[7] Lee, J., Yoo, (2002). S. An elliptical boundary model for skin color detection. School of Computer Science and Engineering, Seoul National University. (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.215.725&rep=rep1&type=pdf)

[8] Said Yacine Boulahia, Eric Anquetil, Franck Multon, Richard Kulpa. Dynamic hand gesture recognition based on 3D pattern assembled trajectories. IPTA 2017 - 7th IEEE International Conference on Image Processing Theory, Tools and Applications, Nov 2017, Montreal, Canada. pp.1-6, <https://hal.archives-ouvertes.fr/hal-01666377/document>. <hal-01666377>

[9] VIII encuentros de geometría computacional: Castellón de la Plana, España, 7-9 de julio de 1999. Publicacions de la Universitat Jaume I, 1999.

[10] Wikipedia contributors (2018). Ramer–Douglas–Peucker algorithm. En *Wikipedia, The Free Encyclopedia*. Disponible en https://en.wikipedia.org/w/index.php?title=Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm&oldid=839205082 [Fecha de acceso: 6 Jun. 2018]

[11] Xu, P. (2017). A Real-time Hand Gesture Recognition and Human-Computer Interaction System. arXiv preprint arXiv:1704.07296.

(https://arxiv.org/pdf/1704.07296.pdf)