

Tecnológico de Costa Rica  
Escuela de Ingeniería Electrónica



**Detección de manos en imágenes de profundidad mediante el  
uso de bosques de decisión aleatorios**

Documento de tesis sometido a consideración para optar por el grado académico de  
Maestría en Electrónica con Énfasis en Procesamiento Digital de Señales

Edison Fernández Alvarado

Cartago, 19 de junio de 2015



Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Edison Fernández Alvarado

Cartago, 19 de junio de 2015

Céd: 3-0422-0672



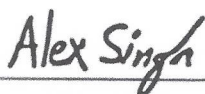
Instituto Tecnológico de Costa Rica  
Escuela de Ingeniería Electrónica  
Tesis de Maestría  
Tribunal evaluador  
Acta de evaluación

Tesis de maestría defendida ante el presente Tribunal Evaluador como requisito para optar por el grado académico de maestría, del Instituto Tecnológico de Costa Rica.

Estudiante: Edison Fernández Alvarado

Nombre del Proyecto: **Detección de manos en imágenes de profundidad mediante el uso de bosques de decisión aleatorios**

Miembros del Tribunal



Alexander Singh Alvarado, Ph.D.

Profesor lector



Dr. Juan Luis Crespo Mariño

Profesor lector



Dr. Pablo Alvarado Moya

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Nota Final de tesis : 95

Cartago, 19 de junio de 2015

Instituto Tecnológico de Costa Rica  
Escuela de Ingeniería Electrónica  
Tesis de Maestría  
Tribunal evaluador

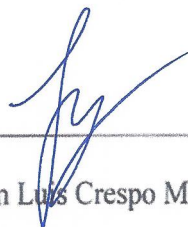
Tesis de maestría defendida ante el presente Tribunal Evaluador como requisito para optar por el grado académico de maestría, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal



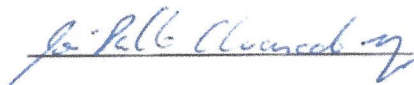
Alexander Singh Alvarado, Ph.D.

Profesor lector



Dr. Juan Luis Crespo Mariño

Profesor lector



Dr. Pablo Alvarado Moya

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 19 de junio 2015

# Resumen

En el presente trabajo se propone un sistema para la detección de manos en imágenes de profundidad. Para realizar dicho cometido, se hace uso de bosques de decisión aleatoria con los cuales se hace una segmentación de la escena en 5 clases definidas: cabeza, brazos, cuerpo, manos y fondo.

Una vez segmentada la escena, mediante el algoritmo de componentes conectados, se procede a agrupar conjuntos de píxeles clasificados como pertenecientes a la misma clase en regiones. A partir de estas regiones se genera una lista de candidatos a mano mediante la validación de los componentes obtenidos.

Mediante la utilización del algoritmo de Dijkstra, se encuentran puntos a una distancia geodésica máxima de 50 cm del candidato a mano y sobre esta ruta se obtiene un histograma de clases el cual es utilizado como descriptor para que, mediante el uso de máquinas de vectores de soporte, se clasifiquen los candidatos como mano o no mano.

Con la propuesta realizada se alcanzan tasas de reconocimiento del 83,05 % sobre una base de datos sintética de 80000 imágenes.

**Palabras clave:** Kinect, imágenes en profundidad, bosques de decisión aleatorios, componentes conectados, distancias geodésicas, Dijkstra, máquinas de vectores de soporte.





# Abstract

In the present work a system for hand detection in depth images is proposed. To perform this task, the scene is segmented into 5 different classes: head, arms, body, hands and background, using the pixel-wise classification of random decision forests.

Once the scene is segmented, the connected components algorithm is applied in order to group sets of pixels of the same class into regions. From these regions, a list of hand candidates is generated by validating the obtained components.

Using Dijkstra's algorithm, the points at a geodesic distance of up to 50 cm from the center of the hand candidate are found, along the found route, a histogram of classes is generated and used as a descriptor for the final classification, which is performed with support vector machine.

With this proposal, a recognition rate of 83,05% is reached over a data base of 80000 synthetic images.

**Keywords:** Kinect, Depth Images, Random Decision Forests, Connected Components, Geodesic Distances, Dijkstra, Support Vector Machines.



*a mis queridos padres*



# Agradecimientos

Quiero aprovechar este espacio para dar mi más sincero agradecimiento a todos los que hicieron posible la culminación de este trabajo. Primero que nada a Dios por haber hecho posible el llegar hasta este punto. A mi familia por el incondicional apoyo, al Dr. Pablo Alvarado Moya por su paciencia y dedicación, a los profesores lectores por su anuencia a colaborar con la realización de este proyecto y a RidgeRun tanto por el apoyo económico como por la flexibilidad ofrecida durante la duración del programa.

Edison Fernández Alvarado

Cartago, 19 de junio de 2015



# Índice general

Índice de figuras	III
Índice de tablas	V
Lista de símbolos y abreviaciones	VII
<b>1. Introducción</b>	<b>1</b>
1.1. Objetivos y estructura del documento . . . . .	3
<b>2. Marco teórico</b>	<b>5</b>
2.1. Bosques de decisión aleatoria (RDF) . . . . .	5
2.2. Componentes conectados . . . . .	8
2.3. Distancias geodésicas . . . . .	11
2.3.1. Algoritmo de Dijkstra . . . . .	11
2.3.2. Cálculo de la distancia geodésica . . . . .	13
2.4. Máquinas de vectores de soporte (SVM) . . . . .	15
2.5. Conceptos estadísticos . . . . .	18
2.5.1. Matriz de confusión . . . . .	18
2.5.2. Sensibilidad . . . . .	20
2.5.3. Especificidad . . . . .	21
2.5.4. Precisión . . . . .	21
2.5.5. Espacios ROC . . . . .	21
2.5.6. Frente de Pareto . . . . .	22
2.6. Estado del arte . . . . .	23
<b>3. Sistema de detección de manos en imágenes de profundidad</b>	<b>29</b>
3.1. Base de datos para entrenamiento . . . . .	30
3.2. Entrenamiento de los bosques de decisión aleatoria . . . . .	31
3.3. Predicción de clases . . . . .	33
3.4. Validación de regiones de mano . . . . .	33
3.5. Búsqueda de candidatos a mano . . . . .	34
3.6. Extracción de descriptores de ruta . . . . .	35
3.6.1. Obtención de la ruta más larga . . . . .	35
3.6.2. Extracción del histograma . . . . .	36
3.7. Clasificación . . . . .	36

3.7.1. Entrenamiento del SVM . . . . .	37
3.7.2. Clasificación . . . . .	37
<b>4. Resultados y análisis</b>	<b>39</b>
4.1. Segmentación de la escena mediante bosques de decisión aleatoria . . . . .	39
4.1.1. Efecto de la variación del desplazamiento de caja y tamaño de región	41
4.1.2. Efecto de la cantidad de niveles . . . . .	46
4.1.3. Efecto de la cantidad de árboles . . . . .	46
4.2. Validación de regiones mano . . . . .	48
4.3. Búsqueda de candidatos a mano . . . . .	52
4.4. Extracción de descriptores de ruta . . . . .	53
4.4.1. Obtención de la ruta más larga . . . . .	54
4.4.2. Extracción del histograma . . . . .	54
4.5. Clasificación mediante Máquinas de Vectores de Soporte (SVM) . . . . .	55
4.5.1. Entrenamiento de la SVM . . . . .	55
4.5.2. Clasificación . . . . .	58
4.6. Comportamiento del clasificador ante imágenes con ruido . . . . .	60
<b>5. Conclusiones</b>	<b>65</b>
<b>Bibliografía</b>	<b>67</b>
<b>Índice alfabético</b>	<b>71</b>



# Índice de figuras

1.1. Diagrama general del sistema . . . . .	2
1.2. Diagrama general del detector de manos . . . . .	2
2.1. Estructura básica de un árbol de decisión . . . . .	6
2.2. Ejemplo de un árbol de decisión . . . . .	6
2.3. Ejemplo de vecindades de pixeles. . . . .	9
2.4. Ejemplos de componentes conectados . . . . .	9
2.5. Ejemplos de distancias geodésicas[30] . . . . .	13
2.6. Ejemplo de un conjunto de vértices. . . . .	14
2.7. Ejemplo de un vértice y sus vecinos . . . . .	14
2.8. Ejemplo de distancia geodésica . . . . .	15
2.9. Ejemplo de clasificador lineal . . . . .	16
2.10. Ejemplo de hiperplanos en un SVM . . . . .	19
2.11. Ejemplo de un espacio ROC . . . . .	22
2.12. Ejemplo de un frente de Pareto . . . . .	23
2.13. Ejemplo de patches utilizados en [19] . . . . .	25
2.14. Ilustración del descriptor de mano propuesto en [16] . . . . .	25
2.15. Ilustración de descriptor usado por [28] . . . . .	26
3.1. Diagrama de bloques de la solución propuesta . . . . .	29
3.2. Subdivisión de la escena en cinco clases . . . . .	30
3.3. Características usadas por el RDF[25] . . . . .	32
3.4. Ejemplo de validación de regiones mano . . . . .	34
3.5. Resultado de la clasificación . . . . .	37
4.1. Ejemplo de división de clases . . . . .	39
4.2. Precisión en la segmentación ante una variación en la cantidad de imágenes y muestras por imagen en el entrenamiento . . . . .	40
4.3. Predicción para un $BD$ y $RS$ de 10 pxm utilizando 3 árboles . . . . .	41
4.4. Predicción para distintos $BD$ y $RS$ . . . . .	43
4.5. Variación de la precisión y sensibilidad en la clase mano ante un cambio en el desplazamiento máximo de caja (BD) y el tamaño máximo de región (RS) . . . . .	44
4.6. Mejores resultados para distintos $BD$ y $RS$ con 3 árboles y 15 niveles . . . . .	45
4.7. Distribución del desplazamiento de caja para un bosque con $BD : RS$ de 20 pxm : 5 pxm . . . . .	45

4.8. Resultados de la variación de la cantidad de niveles . . . . .	46
4.9. Resultado cualitativo de la variación de la cantidad de niveles . . . . .	47
4.10. Resultados de la variación de la cantidad de árboles . . . . .	47
4.11. Comparación de resultados usando 3 y 10 árboles . . . . .	48
4.12. Variación de la cantidad de candidatos a mano y manos reales en función del valor de $\tau_s$ . . . . .	49
4.13. Variación de la cantidad de objetos mano de acuerdo con su tamaño . . . .	49
4.14. Efecto del tamaño de objeto en la separación de clases . . . . .	50
4.15. Error producido por el tamaño del umbral . . . . .	51
4.16. Predicción con un $\tau_s = 100$ pixeles . . . . .	51
4.17. Candidatos a mano . . . . .	52
4.18. Candidatos a mano después de eliminar objetos aislados . . . . .	53
4.19. Ejemplo de una mano real eliminada . . . . .	53
4.20. Variación de la sensibilidad y especificidad en la detección en función de la longitud de la ruta ( $L_{max}$ ) . . . . .	54
4.21. Histogramas para las rutas mostradas en la figura 4.18 . . . . .	55
4.22. Sensibilidad y especificidad vrs $C$ y $\gamma$ para el SVM . . . . .	56
4.23. Sensibilidad y especificidad vrs cantidad de imágenes en SVM con $C = 100$ y $\gamma = 0,2$ . . . . .	57
4.24. Ejemplo de falso positivo debido a clasificación errónea . . . . .	58
4.25. Ejemplo de un falso positivo debido a la posición de la mano . . . . .	59
4.26. Espacio ROC para una variación del valor de $\tau_s$ . . . . .	59
4.27. Efecto de $\tau_s$ en la cantidad de falsos positivos de la clasificación . . . . .	60
4.28. Ejemplos de predicciones con ruido ante la variación de $\sigma$ . . . . .	61
4.29. Predicciones con ruido después de componentes conectados vrs $\sigma$ . . . . .	61
4.30. Predicciones con ruido después de clasificación vrs $\sigma$ . . . . .	62
4.31. Curva ROC para una variación de la cantidad de ruido . . . . .	62
4.32. Efecto de la variación de $\sigma$ en la cantidad de falsos positivos de la clasificación	63

# Índice de tablas

2.1. Ejemplo de una matriz de confusión . . . . .	19
2.2. Ejemplo de una matriz de confusión normalizada por columnas . . . . .	20
2.3. Ejemplo de una matriz de confusión normalizada en las filas . . . . .	20
3.1. Coeficientes para el modelo de ruido en $x$ , $y$ y $z$ . . . . .	31
4.1. Sensibilidad para un $BD$ y $RS$ de 10 pxm . . . . .	41
4.2. Precisión para un $BD$ y $RS$ de 10 pxm . . . . .	42
4.3. Precisión para un $BD$ y $RS$ de 10pxm sin tomar en cuenta la clase fondo .	42
4.4. Sensibilidad para un $BD$ y $RS$ de 20 pxm . . . . .	42
4.5. Precisión para un $BD$ y $RS$ de 20 pxm . . . . .	43



# Capítulo 1

## Introducción

Acompañando el avance tecnológico el ser humano ha buscado nuevas formas de mejorar la interacción del hombre con la máquina. Un ejemplo común es el ratón utilizado para interactuar con una computadora personal. Sin embargo, las investigaciones se han venido orientando cada vez más a la eliminación de la necesidad de la intervención de dispositivos externos para la interacción con las máquinas.

Han surgido diversos esfuerzos por hacer esta interacción cada vez más amigable con el usuario, lo que incluye la utilización de elementos externos como lo puede ser un guante, hasta la utilización de sistemas complejos de cámaras con los cuales se logra ubicar e identificar las acciones del usuario para convertirlas en acciones por parte de los aparatos que se desea controlar. Aunque se han dado avances en este campo, los dispositivos basados en cámaras de video presentan numerosos inconvenientes cuando se trata de interactuar con un usuario: niveles de iluminación, color de piel, entre otros.

Desde el lanzamiento de sensores como el Kinect[38] en noviembre de 2010 han proliferado sus aplicaciones[11]. Una de ellas es la detección y seguimiento de manos. Este tipo de sensores viene a resolver varios de los problemas presentes en sistemas tradicionales basados en color ya que no son afectados por la iluminación y permiten obtener una imagen tridimensional de la escena.

Aunque la iluminación ya no es un factor a tomar en cuenta, factores como lo son la distancia a la cámara, oclusión de objetos e incluso la posición de estos frente a la cámara son elementos a tomar en cuenta a la hora de implementar un sistema basado en este tipo de imágenes.

El presente trabajo es parte de una propuesta que pretende, basado en la utilización de imágenes de profundidad, detectar manos de diversos usuarios, así como la determinación de las distintas poses de estas para finalmente, basado en las manos detectadas y la pose determinada, ajustar un modelo en el que se muestre la posición de las articulaciones. El diagrama general se muestra en la figura 1.1.

La etapa implementada en el presente trabajo se muestra resaltada y se encarga de la

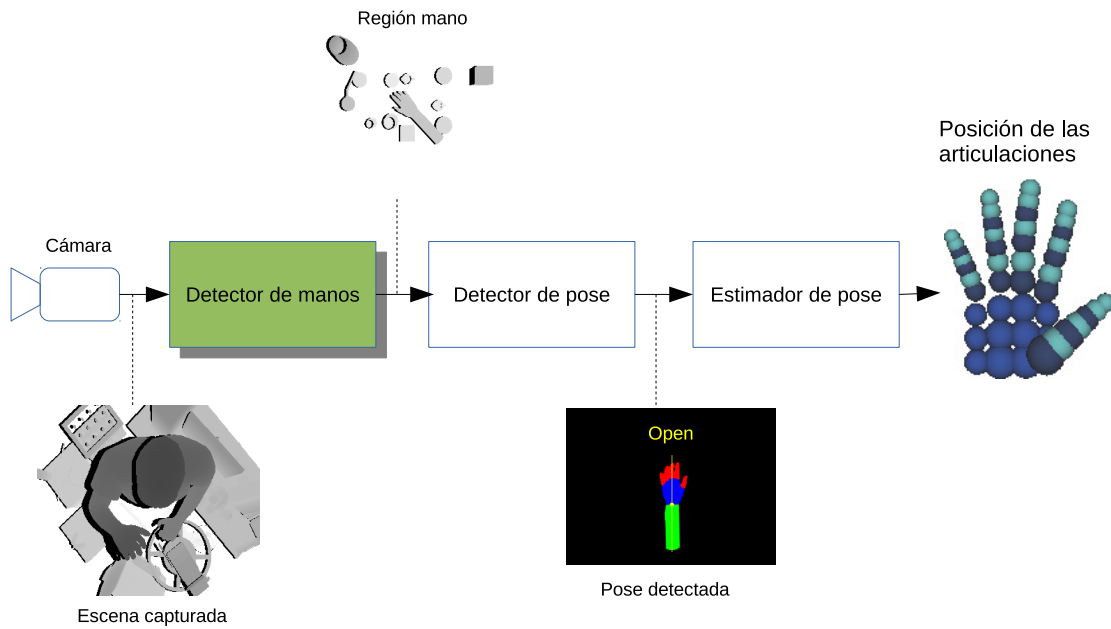


Figura 1.1: Diagrama general del sistema

localización de las manos en las imágenes. Un diagrama general se muestra en la figura 1.2. Para lograr dicho cometido se parte de un clasificador basado en bosques de decisión aleatoria con los cuales se segmenta la escena en 5 clases diferentes. Seguidamente, mediante la utilización del algoritmo de componentes conectados, se agrupan los diferentes componentes presentes en la escena con base en la clasificación realizada por el predictor. Después de esto, se validan los objetos obtenidos. Posteriormente, mediante la utilización de distancias geodésicas, se realiza un histograma de clases el cual es utilizado como descriptor para finalmente, mediante la utilización de máquinas de vectores de soporte, clasificar las posibles manos como mano o no mano.

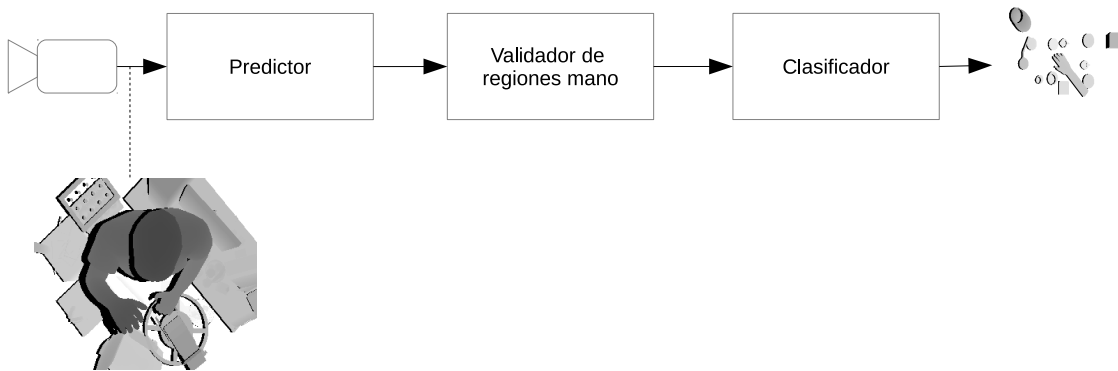


Figura 1.2: Diagrama general del detector de manos

## 1.1. Objetivos y estructura del documento

El presente proyecto tiene como objetivo la detección de manos en imágenes de profundidad mediante la utilización de bosques de decisión aleatoria asumiendo una vista superior de la escena. Para lograr dicho cometido, el sistema debe ser capaz de segmentar la escena en diferentes clases las cuales deben ser agrupadas de acuerdo con la clasificación realizada. Además debe ser capaz de calcular descriptores a partir de los candidatos a mano para finalmente realizar la clasificación de estas como mano o no mano.

En el capítulo 2 se presenta un breve repaso por los distintos aspectos teóricos que concierne al presente proyecto así como los avances realizados hasta la fecha en el tema aquí tratado. Seguidamente, en el capítulo 3, se da una descripción del sistema implementado. En el capítulo 4, se presentan y analizan los resultados obtenidos tras la implementación del sistema propuesto. Finalmente, el capítulo 5 resume las conclusiones obtenidas así como recomendaciones para futuras implementaciones.





# Capítulo 2

## Marco teórico

A continuación se cubren los conceptos teóricos utilizados a lo largo de este trabajo. Inicialmente se da una descripción de los bosques de decisión aleatoria (RDF por sus siglas en inglés) utilizados en la clasificación de píxeles. Posteriormente se describe el algoritmo de componentes conectados con el cuál se agrupan los píxeles clasificados en componentes u objetos, luego se analiza la técnica utilizada para el cálculo de distancias geodésicas en una imagen de profundidad a partir de la utilización del algoritmo de Dijkstra la cuál es utilizada como base para la obtención del descriptor utilizado para la clasificación de manos. Posteriormente se presenta una descripción de las máquinas de vectores de soporte (SVM por sus siglas en inglés) con las que se realiza la clasificación final de los objetos como mano o no mano seguido de un repaso de los indicadores estadísticos usados en la evaluación de resultados para finalizar con un repaso de los distintos esfuerzos realizados hasta la fecha relacionados con el enfoque de este proyecto.

### 2.1. Bosques de decisión aleatoria (RDF)

La imagen de profundidad es utilizada como entrada para un clasificador del tipo bosque de decisión aleatoria el cual debe clasificar cada uno de los píxeles como perteneciente a una de 5 clases.

Los RDF son una combinación de árboles de decisión de tal forma que cada árbol ha sido entrenado con parámetros aleatorios muestreados independientemente y con la misma distribución para todos los árboles en el bosque [4, 8].

Un árbol es una colección de nodos y aristas organizados en una estructura jerárquica como se puede ver en la figura 2.1 [8]. Los nodos son divididos en nodos internos y nodos terminales u hojas. En la figura 2.1 se pueden ver los nodos internos como círculos y los nodos terminales como cuadrados.

Un árbol de decisión es un árbol utilizado para tomar decisiones [8], y codifica una jerarquía de preguntas que es usada para mapear un valor de entrada multidimensional

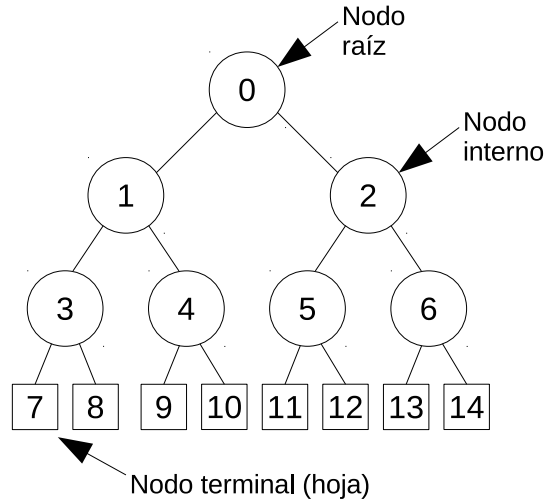


Figura 2.1: Estructura básica de un árbol de decisión

a una salida escalar. Esta salida puede ser un valor escalar (regresión) o una etiqueta (clasificación) [10]. La figura 2.2 muestra un ejemplo de un árbol de decisión que puede formar parte de un bosque.

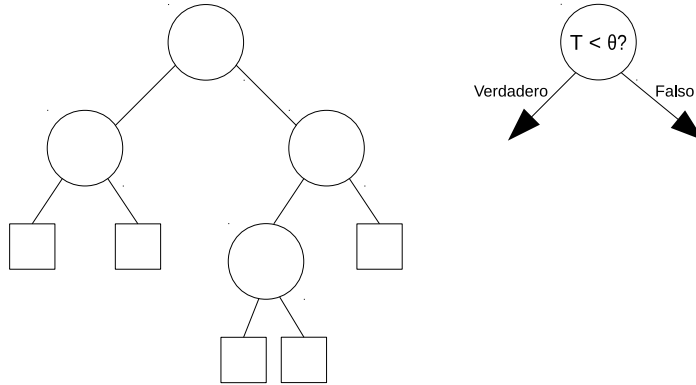


Figura 2.2: Ejemplo de un árbol de decisión

Sea  $\underline{\mathbf{x}} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$  un vector de datos genéricos con las componentes  $x_i$  representando las respuestas a características escalares. Esas características son aleatoriamente muestreadas del conjunto de todas las  $d$  posibles características con una función  $\phi(\underline{\mathbf{x}})$  que selecciona un subconjunto de  $d'$  características de interés:

$$\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}, \text{ con } d' \ll d \quad (2.1)$$

El entrenamiento de un árbol de decisión involucra enviar todos los datos de entrenamiento  $\{\underline{\mathbf{x}}\}$  al árbol y optimizar los parámetros de los nodos de tal forma que se optimice una función de energía escogida.

El funcionamiento de un árbol de decisión se puede dividir en una etapa fuera de línea (entrenamiento) y una etapa en tiempo real (prueba). La etapa de prueba consiste en

que, a partir de un punto no visto  $\underline{\mathbf{v}}$ , el árbol de decisión aplica jerárquicamente una serie de pruebas predefinidas (en el entrenamiento). Iniciando del nodo raíz, cada nodo aplica su función de decisión asociada a  $\underline{\mathbf{v}}$ . Dependiendo del resultado de la prueba, se avanza al nodo izquierdo si el resultado es positivo o al derecho de lo contrario y el proceso se repite hasta alcanzar el nodo hoja  $l_n(\underline{\mathbf{v}})$  el cual almacena la etiqueta de una clase o la distribución  $p(c|l_n(\underline{\mathbf{v}}))$  sobre las etiquetas de las clases  $c \in \underline{\mathbf{C}}$ . Los resultados de las hojas alcanzadas en el bosque  $F$  son combinados para generar una sola clasificación[10]. Un método común para la obtención del valor final es el voto de la mayoría o el promedio de todas las distribuciones de probabilidad definida por:

$$p(c|F, \underline{\mathbf{v}}) = \frac{1}{n} \sum_{n=1}^n p(c|l_n(\underline{\mathbf{v}})) \quad (2.2)$$

Por otra parte, la etapa de entrenamiento se encarga de optimizar los parámetros asociados con las funciones de decisión de cada uno de los nodos internos así como los predictores de las hojas.

Sea  $\mathcal{S}_1$  un subconjunto de datos de entrenamiento en el nodo 1 y  $\mathcal{S}_1^L, \mathcal{S}_1^R$  los subconjuntos de datos que van a los nodos hijos izquierdo y derecho respectivamente. Dado un conjunto de datos de entrenamiento  $\mathcal{S}_0 \{ \underline{\mathbf{v}} \}$  y sus asociadas etiquetas (clase a la que pertenecen), los parámetros del árbol son elegidos de tal forma que se optimiza la función de energía escogida. En el caso de bosques con  $T$  árboles, el entrenamiento es repetido independientemente para cada árbol.

Una posible función de energía a optimizar se denomina ganancia de información y está definida como

$$I = H(\mathcal{S}) - \sum_{i \in \{1,2\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i) \quad (2.3)$$

con la entropía de Shannon definida matemáticamente como

$$H(\mathcal{S}) = - \sum_{c \in \mathcal{C}} p(c) \log(p(c)) \quad (2.4)$$

con  $I$  una medida de la ganancia de información obtenida al dividir el conjunto de entrenamiento y  $p(c)$  es calculado como el histograma empírico normalizado de las etiquetas correspondientes a los puntos de entrenamiento en  $\mathcal{S}$ .

Cada nodo  $j$  es asociado con una función de clasificación binaria

$$h(\underline{\mathbf{v}}, \underline{\theta}_j) \in \{0, 1\} \quad (2.5)$$

con 0 indicando falso y 1 indicando verdadero (como ejemplo). Esta función está caracterizada por sus parámetros  $\underline{\theta} = (\phi, \underline{\psi}, \underline{\tau})$  donde  $\underline{\psi}$  define la primitiva geométrica utilizada para separar los datos (un hiperplano alineado al eje, un hiperplano oblicuo, una superficie, etc), el vector  $\underline{\tau}$  captura los umbrales para las desigualdades usadas en la prueba binaria y la función  $\phi$  selecciona algunas características del vector  $\underline{\mathbf{v}}$ .

Durante la etapa de entrenamiento, los parámetros  $\underline{\theta}_j^*$  del nodo  $j$  necesitan ser calculados. Esto se lleva a cabo maximizando la función objetivo de ganancia de información

$$\underline{\theta}_j^* = \arg \max_{\underline{\theta}_j} I_j \quad (2.6)$$

con

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R, \underline{\theta}_j) \quad (2.7)$$

donde  $\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R$  denotan los conjuntos de entrenamiento antes y después del nodo.

Un punto clave en los bosques de decisión es que cada árbol es aleatoriamente independiente del otro. Esto lleva a una no correlación entre las predicciones de los árboles individuales y por lo tanto a una mejora en la generalización y haciéndolo robusto con respecto al ruido en los datos[8].

En el caso de los nodos hoja, durante el entrenamiento, la información que es útil para la predicción será aprendida para todos. En el caso de la clasificación, cada hoja puede almacenar la distribución empírica sobre las clases asociadas a un subconjunto de datos de entrenamiento que ha alcanzado esa hoja. El modelo probabilístico del predictor de hoja para el árbol  $t$  es por lo tanto

$$p_t(c|\mathbf{v}) \quad (2.8)$$

con  $c \in \{c_k\}$  indexando las clases.

El proceso de entrenamiento se puede detener cuando, por ejemplo, un máximo número de niveles  $D$  es alcanzado, una ganancia de información  $I$  mínima es obtenida o la cantidad de datos de entrenamiento en un nodo es menor a un umbral dado.

En [4, 8] se demuestra cómo los bosques de decisión aleatoria no tienen sobreajuste conforme se agregan más árboles y limitan el error de generalización. Además en [8] se demuestra cómo al aumentar la cantidad de niveles del árbol se aumenta la confianza de la predicción; sin embargo, se puede tender a un sobre ajuste lo cual puede ser compensado agregando más árboles al bosque.

## 2.2. Componentes conectados

Los pixeles clasificados como pertenecientes a una determinada clase son utilizados como entrada para el algoritmo de componentes conectados con el cual se agrupan formando objetos.

Dos pixeles se dicen conectados si son vecinos que comparten una propiedad común que define un componente la cual puede ser color, brillo, rango de brillo, entre otras. Los pixeles pueden estar conectados en vecindades de 4 o de 8 como se ilustra en la figura 2.3 donde se muestran ejemplos de un pixel (marcado en gris) y sus vecinos (de color negro) para ambos casos[23].

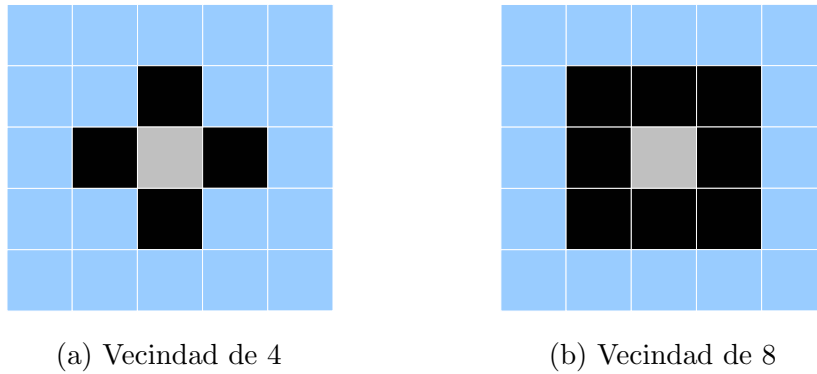


Figura 2.3: Ejemplo de vecindades de pixeles.

Un conjunto  $S$  de pixeles es un componente conectado si hay al menos un camino en  $S$  que une cada par  $\{p, q\}$  de pixeles en  $S$ . El camino debe contener únicamente pixeles en  $S$ .

En la figura 2.4 se muestran ejemplos de componentes conectados para vecindades de 4 (negro y verde) y 8 pixeles (rojo).

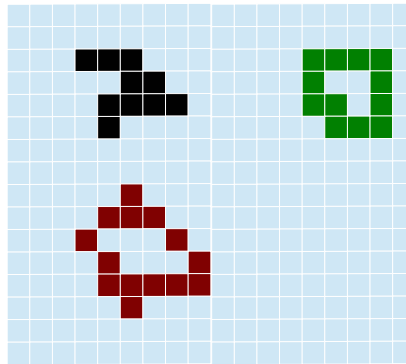


Figura 2.4: Ejemplos de componentes conectados

El algoritmo de etiquetado de componentes conectados segmenta el dominio de una imagen en particiones que corresponden a los componentes conectados [22] asignándole a cada componente una etiqueta.

El algoritmo 1 ejemplifica el algoritmo básico para una imagen binaria [20].

En este se asume que los únicos valores en la imagen son 0 y 1 y a cada uno de los componentes encontrados se le asigna una nueva etiqueta *NuevaEtiqueta* con lo que al culminar el algoritmo se sabe la cantidad de componentes presentes en la imagen. Aunque este ejemplo aplica para imágenes binarias este puede ser fácilmente extendido a imágenes a color donde como condición para agrupar pixeles se utiliza que sean del mismo color o se encuentren en un rango determinado. En este proyecto se forman objetos agrupando pixeles que fueron clasificados como pertenecientes a la misma clase, lo que implica que tienen el mismo valor.

---

**Algorithm 1** Algoritmo Clásico de componentes conectados

---

```

//lp, lq, lx : etiquetas asignadas a p, q, x
//Primer escaneo
for  $i = 1; i < NROWS - 1; i++$  do
  for  $j = 1; j < NCOLS - 1; j++$  do
    if  $I[i, j] == 1$  then
       $lp = I[i - 1, j];$ 
       $lq = I[i, j - 1];$ 
      if  $lp == 0 \ \&\& \ lq == 0$  then
         $NuevaEtiqueta++;$ 
         $lx = NuevaEtiqueta;$ 
      end if
      else if  $(lp \neq lq) \ \&\& \ (lp \neq 0) \ \&\& \ (lq \neq 0)$  then
        //Registrar equivalencia (lp, lq)
         $lx = lq;$ 
      else if  $lq \neq 0$  then
         $lx = lq;$ 
      else if  $lp \neq 0$  then
         $lx = lp;$ 
      end if
       $I[i, j] = lx;$ 
    end for
  end for
end for

```

---

## 2.3. Distancias geodésicas

Sobre la superficie de los objetos encontrados, mediante la utilización de distancias geodésicas, se obtienen rutas con el fin de formar descriptores que posteriormente serán utilizados para la clasificación.

Para la obtención de las distancias geodésicas se utiliza como base el algoritmo de Dijkstra.

### 2.3.1. Algoritmo de Dijkstra

Según definido por Dijkstra [9] para obtener el camino más corto entre cualesquiera dos nodos  $P$  y  $Q$  en un grafo, se deben resolver 2 problemas: construir un árbol de las distancias totales mínimas entre los  $n$  nodos y encontrar el camino con la distancia total mínima entre  $P$  y  $Q$ .

Se asume que existe al menos un camino entre cualesquiera dos nodos del grafo. Los vértices se dividen en tres grupos:

- I. Aristas asignadas al árbol.
- II. Aristas de las cuales será elegida la siguiente a ser asignada a I.
- III. Las demás aristas.

Por otra parte los nodos se dividen en dos grupos:

- A. Nodos conectados por las aristas en I.
- B. Los nodos restantes.

La construcción del árbol se inicia eligiendo de forma arbitraria un nodo como el único miembro de A y poniendo todas las aristas que llegan a este nodo en II. Se inicia con I vacío.

A partir de este punto, el algoritmo consiste en los siguientes pasos:

1. La arista más corta de II se remueve de este grupo y se agrega a I. Como resultado, un nodo es transferido de B a A.
2. Considerando las aristas que van del nodo que acaba de ser transferido al conjunto A, a los nodos que aun están en el conjunto B, si la arista en consideración es más larga que la correspondiente arista en el conjunto II, se rechaza. Si esta es más corta, reemplaza la arista en el conjunto II y la última es rechazada. Luego se vuelve al paso 1 y se repite el proceso hasta que los conjuntos II y B están vacíos.

Una vez construido el árbol, se procede a encontrar la distancia total mínima entre dos nodos dados  $P$  y  $Q$ . Para ello se parte del hecho de que si  $R$  es un nodo en el camino mínimo de  $P$  a  $Q$ , el conocimiento del último implica el conocimiento del camino mínimo de  $P$  a  $R$ .

Para encontrar la solución, los nodos son subdivididos en tres conjuntos:

- A. Los nodos para los cuales se conoce el camino de longitud mínima desde  $P$ . Nodos serán agregados a este conjunto para incrementar la longitud del camino mínimo desde el nodo  $P$ .
- B. Los nodos de los cuales será seleccionado el siguiente nodo a ser agregado al conjunto A. Este conjunto comprende todos los nodos que están conectados al menos a un nodo del conjunto A pero aun no pertenecen a este.
- C. Los demás nodos.

De igual forma, las aristas son divididas en tres conjuntos:

- I. Las aristas del camino mínimo desde  $P$  a los nodos en el conjunto A.
- II. Las aristas de las cuales será seleccionada la siguiente a ser agregada al conjunto I.
- III. Las aristas restantes.

Para iniciar, todos los nodos se encuentran en el conjunto C y todas las aristas se encuentran en el conjunto III. Se transfiere el nodo  $P$  al conjunto A y se ejecutan los siguientes pasos:

- I. Considerando todas las aristas  $r$  conectando el nodo que se acaba de transferir al conjunto A con nodos  $R$  en los conjuntos B o C. Si el nodo  $R$  pertenece al conjunto B, se investiga si el uso de la arista  $r$  da lugar a un camino más corto de  $P$  a  $R$  que el camino conocido que usa la correspondiente arista en el conjunto II. Si este no es el caso, la arista  $r$  es rechazada. Si por lo contrario, al utilizar la arista  $r$  se obtiene un camino más corto, se reemplaza la correspondiente arista en el conjunto II y la existente es rechazada. Si el nodo  $R$  pertenece al conjunto C, este es agregado al conjunto B y la arista  $r$  es agregada al conjunto II.
- II. Cada nodo en el conjunto B puede estar conectado al nodo  $P$  en solo una forma si nos restringimos a aristas del conjunto I y una del conjunto II. En este sentido, cada nodo en el conjunto B tiene una distancia desde el nodo  $P$ : el nodo con mínima distancia desde  $P$  es transferido del conjunto B al conjunto A y la correspondiente arista es transferida del conjunto II al conjunto I. Luego se repite el paso 1 y se continúa el proceso hasta que el nodo  $Q$  haya sido transferido al conjunto A.



### 2.3.2. Cálculo de la distancia geodésica

Una geodésica es una curva restringida a una superficie la cual minimiza localmente la distancia intrínseca entre dos puntos de tal forma que satisface la ecuación [30]:

$$0 = \frac{d}{dt} \int_I \|\dot{\gamma}(t)\| dt \quad (2.9)$$

para una curva parametrizada  $\gamma$  en el intervalo  $I : [a, b]$  donde  $\|\cdot\|$  denota la norma.

En la figura 2.5 se puede ver un ejemplo de geodésicas. En ella se pueden identificar tres rutas entre dos punto.

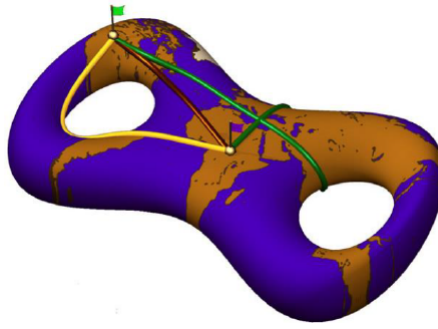


Figura 2.5: Ejemplos de distancias geodésicas[30]

Sea  $I$  una imagen de profundidad de dimensiones  $(w, h)$  y sea  $q$  un pixel cualquiera en la posición  $(u, v)$  con  $0 \leq u < w$  y  $0 \leq v < h$ . Además sea  $I(q)$  la profundidad en milímetros en el pixel  $q$ .

Para realizar el cálculo de la distancia geodésica entre dos puntos  $V_1$  y  $V_2$  en la imagen  $I$  es necesario convertir la posición del pixel  $q$  al punto correspondiente en 3D. Para ello se hace uso del procedimiento descrito en [14] donde a partir de las coordenadas  $(u, v)$  y la profundidad de dicho punto  $I(q)$  se calculan sus coordenadas  $(x, y, z)$  con:

$$\begin{aligned} x &= \frac{I(q) * (u - p_x)}{f_x} \\ y &= \frac{I(q) * (v - p_y)}{f_y} \\ z &= I(q) \end{aligned} \quad (2.10)$$

donde  $p_x$ ,  $p_y$ ,  $f_x$  y  $f_y$  son los parámetros intrínsecos de un modelo de cámara estenopéico. Los valores  $p_x$  y  $p_y$  corresponden al punto principal y  $f_x$  y  $f_y$  la distancia focal.

El siguiente paso para el cálculo de la distancia geodésica entre dos puntos  $V_1$  y  $V_2$  en  $I$  es encontrar la distancia mínima entre ellos a travez de todos los puntos de  $I$  que los separa utilizando el algoritmo de Dijkstra [9].

La figura 2.6 muestra un ejemplo de un conjunto de vértices. En ella se puede observar  $V$  como uno de los nodos o vértices y  $W$  como el costo de desplazarse de un vértice al otro.

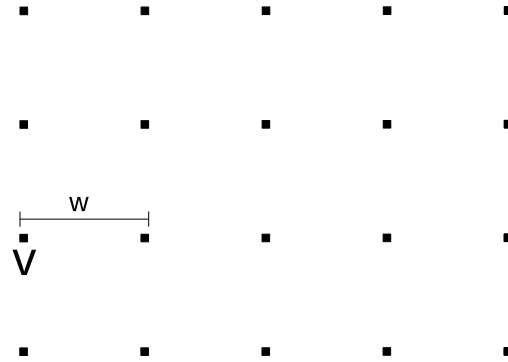


Figura 2.6: Ejemplo de un conjunto de vértices.

Para aplicar el algoritmo de Dijkstra a la imagen de profundidad se considera cada uno de los puntos de la imagen como uno de los vértices del grafo. Cada uno de estos puntos está unido a sus 8 vecinos por una arista y la distancia euclídeana en el espacio 3D entre estos es el peso de desplazarse a cada uno de ellos. Esto se ilustra en la figura 2.7.

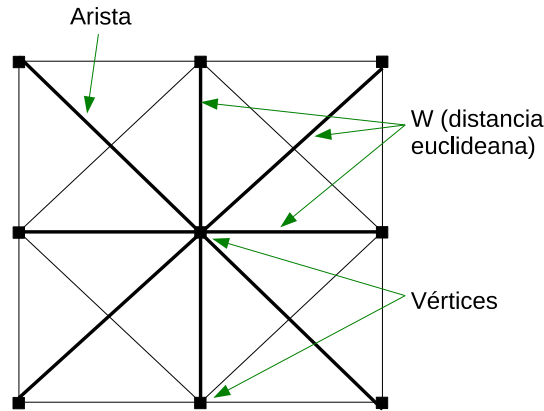


Figura 2.7: Ejemplo de un vértice y sus vecinos

Para realizar este grafo es necesario aplicar ciertas restricciones para la asignación del peso  $W$  de desplazarse de un vértice a otro vecino. Una de ellas es que si la distancia euclídeana entre ellos es mayor a un umbral dado se asigna un valor de  $W = \infty$ , esto con el fin de mantener la ruta encontrada sobre una sola superficie u objeto. Además, si un pixel vecino tiene un valor inválido de profundidad, la distancia entre ellos también se considera infinita.

Teniendo estas restricciones en cuenta se aplica el algoritmo de Dijkstra al grafo y la distancia geodésica entre dos puntos  $V_1$  y  $V_2$  dentro de la imagen es el costo de desplazarse de  $V_1$  a  $V_2$  por la ruta encontrada utilizando Dijkstra. Esto se puede definir matemáticamente como:

$$d(V_2) = W_1 + W_2 + \dots + W_n \quad (2.11)$$

donde  $W_i$  corresponde al peso de desplazarse entre cada uno de los vértices que comprenden la ruta entre  $V_1$  y  $V_2$ . En la figura 2.8 se ilustra el concepto gráficamente.

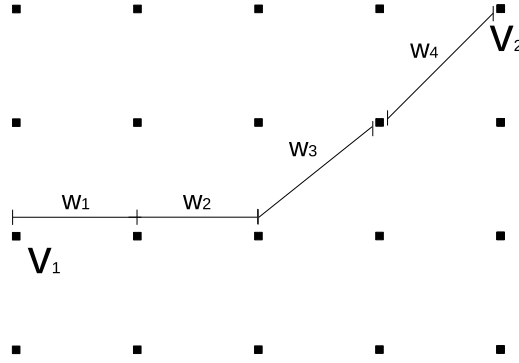


Figura 2.8: Ejemplo de distancia geodésica

La ruta encontrada presenta la característica de minimizar la distancia entre los puntos  $V_1$  y  $V_2$  y se encuentra sobre la superficie del cuerpo al que pertenecen. La información contenida por los píxeles en ella es posteriormente utilizada en la etapa final de clasificación para la separación de los objetos mano de los no mano.

## 2.4. Máquinas de vectores de soporte (SVM)

Descriptores calculados a lo largo de la ruta encontrada con distancias geodésicas se utilizan como entrada para clasificadores estadísticos que deben discernir entre objetos mano y no mano. En este trabajo se emplean máquinas de vectores de soporte. Una máquina de vectores de soporte (SVM) es un método de clasificación introducido en 1992 por Boser, Guyon y Vapnik y pertenece a la categoría de métodos de kernel [2].

Antes de definir las SVM es necesario dar una breve introducción a un clasificador lineal. Este está definido por el vector  $\underline{\mathbf{w}}$  normal a un hiperplano y un desplazamiento  $b$  como parámetros del hiperplano

$$\{\underline{\mathbf{x}} | (\underline{\mathbf{w}} \cdot \underline{\mathbf{x}}) + b = 0\} \quad (2.12)$$

que parte el espacio de características en dos. La figura 2.9 muestra un ejemplo de un clasificador lineal. En ella se puede observar el hiperplano (la línea punteada) y el vector  $\underline{\mathbf{w}}$  normal a este además del margen, el cual se define como la distancia mínima de cualquier punto del entrenamiento al hiperplano. La clasificación de un punto como perteneciente

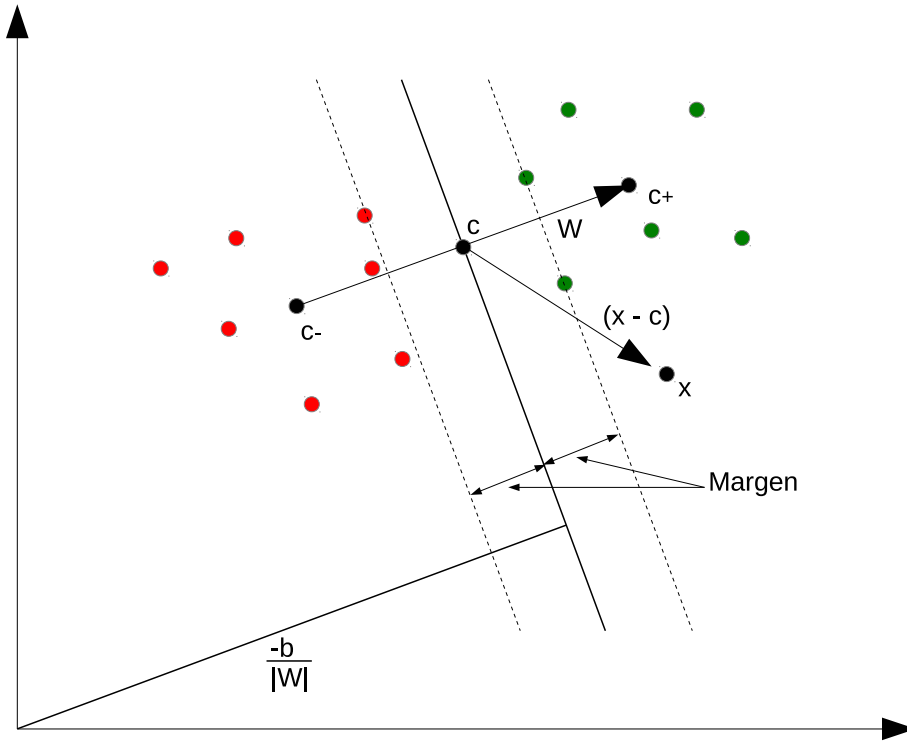


Figura 2.9: Ejemplo de clasificador lineal

a cualquiera de las dos clases (los dos extremos del hiperplano) se realiza mediante la ecuación

$$f(x) = \text{sgn}((\underline{\mathbf{w}} \cdot \underline{\mathbf{x}}) + b) \quad (2.13)$$

No obstante, el clasificador lineal mostrado en la figura 2.9 es un caso trivial, ya que para casos con mayor cantidad de dimensiones se requiere de funciones no lineales para lograr la separación de los datos.

Sea un mapeo no lineal el definido por

$$\begin{aligned} \Phi : \mathbb{R}^N &\rightarrow \mathcal{F} \\ \underline{\mathbf{x}} &\mapsto \Phi(\underline{\mathbf{x}}) \end{aligned} \quad (2.14)$$

Los datos  $\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \dots, \underline{\mathbf{x}}_n \in \mathbb{R}^N$  (espacio de entrada) son mapeados a un espacio de características  $\mathcal{F}$  potencialmente de mayores dimensiones. De esta forma, para un problema de aprendizaje dado, se puede considerar ahora el algoritmo de clasificación en el espacio  $\mathcal{F}$  en lugar del espacio  $\mathbb{R}^N$ . De esta forma se trabaja con las muestras

$$(\Phi(\underline{\mathbf{x}}_1), y_1), \dots, (\Phi(\underline{\mathbf{x}}_n), y_n) \in \mathcal{F} \times Y \quad (2.15)$$

Dado un conjunto de datos no separables linealmente, el mapeo encontrado  $\Phi$  transforma este conjunto de modo que los datos son separables linealmente en  $\mathcal{F}$ , con lo que se tiene la ventaja de poder realizar una separación lineal en  $\mathcal{F}$  cuando en el espacio de entrada se requiere de una separación no lineal.

Conforme la cantidad de dimensiones crece, aunque se pueda mantener control sobre la complejidad estadística, la complejidad del algoritmo crece. Es por esto que para ciertos espacios  $\mathcal{F}$  y sus correspondientes mapeos  $\Phi$  hay un truco efectivo para calcular el producto escalar en el espacio de características utilizando funciones kernel. Si por ejemplo se tiene

$$\begin{aligned}\Phi : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)\end{aligned}\tag{2.16}$$

el producto escalar entre dos vectores del espacio de características puede ser formulado en términos del kernel  $k$

$$\begin{aligned}(\Phi(\underline{\mathbf{x}}) \cdot \Phi(\underline{\mathbf{y}})) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^T \\ &= ((x_1, x_2)(y_1, y_2)^T)^2 \\ &= (\underline{\mathbf{x}} \cdot \underline{\mathbf{y}})^2 \\ &:= k(\underline{\mathbf{x}}, \underline{\mathbf{y}})\end{aligned}\tag{2.17}$$

con esto se tiene que para  $\underline{\mathbf{x}}, \underline{\mathbf{y}} \in \mathbb{R}^N$  y  $d \in \mathbb{N}$  la función kernel

$$k(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = (\underline{\mathbf{x}} \cdot \underline{\mathbf{y}})^d\tag{2.18}$$

calcula el producto escalar. Además, si  $k : \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}$  es un kernel continuo de un operador integral positivo en un espacio de Hilbert  $L_2(\mathcal{C})$  en un conjunto compacto  $\mathcal{C} \subset \mathbb{R}^N$ , existe un espacio  $\mathcal{F}$  y un mapeo  $\Phi : \mathbb{R}^N \rightarrow \mathcal{F}$  tal que  $k(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = (\Phi(\underline{\mathbf{x}}) \cdot \Phi(\underline{\mathbf{y}}))$ . El punto interesante de los kernels es que se puede realizar el producto escalar de dos vectores en  $\mathcal{F}$  sin conocer el mapeo  $\Phi$ . Un tipo común de kernel es el Gaussiano RBF que está definido por

$$k(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \exp\left(\frac{-\|\underline{\mathbf{x}} - \underline{\mathbf{y}}\|^2}{c}\right)\tag{2.19}$$

Volviendo al clasificador lineal, las consideraciones para una clasificación sin error de entrenamiento están definidas por

$$y_i((\underline{\mathbf{w}} \cdot \underline{\mathbf{x}}_i) + b) \geq 1, i = 1, \dots, n\tag{2.20}$$

y si se considera un clasificador lineal en  $\mathcal{F}$ , se sustituye  $\underline{\mathbf{x}}_i$  por  $\Phi(\underline{\mathbf{x}}_i)$  en (2.21), por lo que las condiciones para un clasificador perfecto son

$$y_i((\underline{\mathbf{w}} \cdot \Phi(\underline{\mathbf{x}}_i) + b) \geq 1, i = 1, \dots, n\tag{2.21}$$

El objetivo de aprender (entrenar el clasificador) es encontrar un  $\underline{\mathbf{w}} \in \mathcal{F}$  y  $b$  de tal forma que el error esperado se minimice. En [17] se demuestra cómo, con funciones kernel  $k(\underline{\mathbf{x}}_i, \underline{\mathbf{x}}_j)$ , este problema consiste en un problema de optimización cuadrático dual

$$\begin{aligned}\max_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k(\underline{\mathbf{x}}_i, \underline{\mathbf{x}}_j) \\ \text{sujeto a } & \alpha_i \geq 0, i = 1, \dots, n, \\ & \sum_{i=1}^n \alpha_i y_i = 0\end{aligned}\tag{2.22}$$

Así resolviendo el problema de optimización, se obtienen los coeficientes  $\alpha_i, i = 1, \dots, n$ , lo cual lleva a la función de decisión

$$\begin{aligned} f(x) &= \text{sgn} \left( \sum_{i=1}^n y_i \alpha_i (\Phi(\underline{\mathbf{x}}) \cdot \Phi(\underline{\mathbf{x}}_i)) + b \right) \\ &= \text{sgn} \left( \sum_{i=1}^n y_i k(\underline{\mathbf{x}}, \underline{\mathbf{x}}_i) + b \right) \end{aligned} \quad (2.23)$$

El caso anteriormente descrito se aplica a conjuntos separables con un error empírico de 0. Sin embargo, en casos de datos con ruido, este puede no ser el caso, por lo que se definen las condiciones a cumplir como

$$\begin{aligned} \min_{\underline{\mathbf{w}}, b, \xi} \quad & \frac{1}{2} \underline{\mathbf{w}}^T \underline{\mathbf{w}} + C \sum_{i=1}^n \xi_i \\ \text{sujeto a } & y_i ((\underline{\mathbf{w}} \cdot \Phi(\underline{\mathbf{x}}_i)) + b) \geq 1 - \xi_i, \xi_i \geq 0, \\ & i = 1, \dots, n \end{aligned} \quad (2.24)$$

donde se introduce el nuevo término  $\xi_i$  con  $C > 0$  el parámetro de regularización (para un desarrollo detallado ver [17]).

La figura 2.9 muestra un ejemplo de un hiperplano en un espacio 2D que separa de manera perfecta los datos. Sin embargo, como se puede deducir, el hiperplano mostrado no es el único que permite la separación de los datos. Entre todos los posibles hiperplanos existe uno óptimo distinguido por el margen máximo de separación entre cualquier punto del entrenamiento y el hiperplano. La figura 2.10 muestra un ejemplo de estos hiperplanos, como se puede ver, estos son definidos sobre vectores de la frontera a los que se le conoce como vectores de soporte, lo cual da el nombre de máquinas de vectores de soporte a este clasificador.

## 2.5. Conceptos estadísticos

A continuación se da un repaso de los indicadores estadísticos utilizados en la evaluación de resultados.

### 2.5.1. Matriz de confusión

La matriz de confusión es una herramienta de visualización que se emplea en aprendizaje supervisado. En ella cada columna representa el número de predicciones de cada clase mientras las filas representan las instancias de la clase real. La tabla 2.1 muestra un ejemplo de una matriz de confusión. Como se puede ver, si se suman los elementos de la fila se observa que en el experimento se tiene un total de 8 muestras de carros de las cuales 2 fueron identificadas como carros, 5 como motos y 1 como bus. Si se suman los

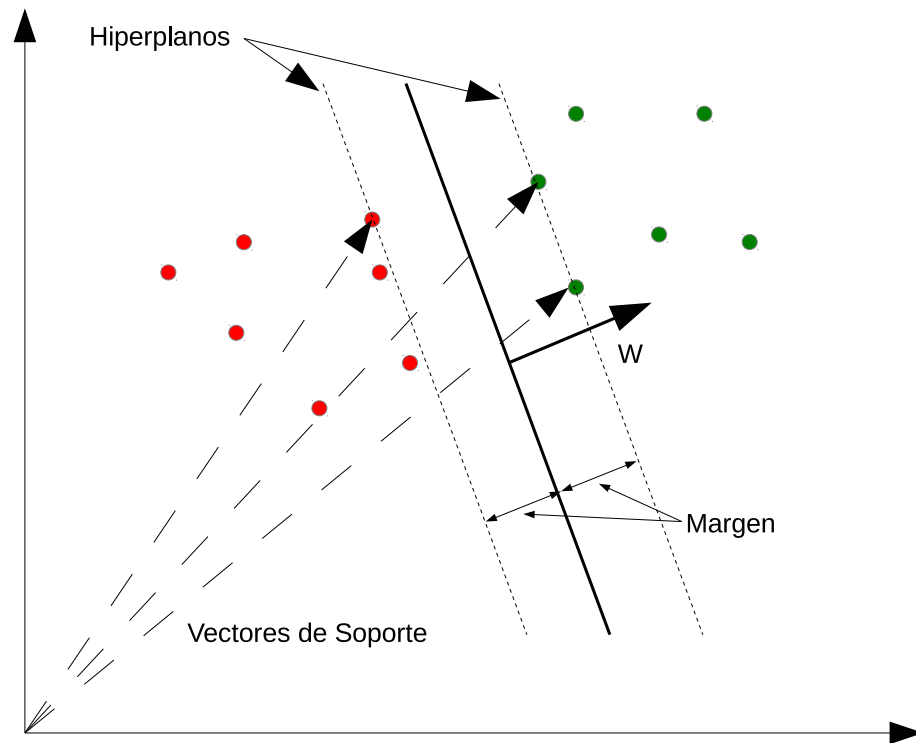


Figura 2.10: Ejemplo de hiperplanos en un SVM

	Carro	Moto	Bus
Carro	2	5	1
Moto	3	7	2
Bus	4	1	8

Tabla 2.1: Ejemplo de una matriz de confusión

elementos de la columna se puede ver que se identificaron un total de 9 elementos como carro de los cuales 2 son carros reales, 3 son motos y 4 son buses.

Es usual normalizar la matriz de tal modo que los resultados se presenten como porcentajes en lugar de cifras absolutas. De esta forma se pueden tener dos tipos de normalizaciones: normalización por columnas y normalización por filas.

En la normalización por columnas, los elementos se normalizan por el número total de ocurrencias en su columna permitiendo presentar los resultados como un porcentaje del total de predicciones. La tabla 2.2 muestra la matriz de la tabla 2.1 normalizada por columnas. Como se puede ver, cada elemento es representado como un porcentaje del total de predicciones en cada clase. Así por ejemplo, del total de muestras predichas como carro, un 22,2 % son realmente carros, mientras un 33,3 % son motos y un 44,5 % son buses.

Finalmente, al normalizar por filas se presenta cada resultado como un porcentaje del total de elementos reales en la clase. La tabla 2.3 muestra la matriz de la tabla 2.1

	Carro	Moto	Bus
Carro	0,222	0,385	0,091
Moto	0,333	0,538	0,182
Bus	0,445	0,077	0,727
Total	1	1	1

**Tabla 2.2:** Ejemplo de una matriz de confusión normalizada por columnas

	Carro	Moto	Bus	Total
Carro	0,250	0,625	0,125	1
Moto	0,250	0,583	0,167	1
Bus	0,308	0,077	0,615	1

**Tabla 2.3:** Ejemplo de una matriz de confusión normalizada en las filas

normalizada por filas. Como se puede ver, cada uno de los resultados es presentado como un porcentaje del total de elementos pertenecientes a esa clase. Así por ejemplo, del total de carros presentes en el experimento un 25,0 % fue identificado como carro, un 62,5 % fue identificado como moto y un 15,5 % fue identificado como bus.

Con lo anterior se puede deducir que, sin importar el tipo de normalización usada, la predicción es mejor cuanto más cerca esté la matriz de confusión de la matriz identidad, representando esta un resultado de clasificación perfecto.

### 2.5.2. Sensibilidad

La sensibilidad, también conocida como la tasa de verdaderos positivos (TPR), indica la capacidad de un clasificador binario de dar como resultados positivos los resultados realmente positivos. Esta se puede expresar como:

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \quad (2.25)$$

donde  $TP$  indica la cantidad de verdaderos positivos o elementos correctamente identificados y  $FN$  la cantidad de falsos negativos o elementos incorrectamente descartados.

Desde el punto de vista de una matriz de confusión, al normalizar por filas se está mostrando la sensibilidad de un clasificador n-ario, ya que se está mostrando el porcentaje de elementos clasificados como pertenecientes a determinada clase dentro del total de elementos pertenecientes a esa clase.



### 2.5.3. Especificidad

La especificidad de un clasificador binario, también conocida como tasa de verdaderos negativos (TNR), es un indicador de la proporción de negativos que son correctamente identificados como tal, o lo que es igual, el porcentaje de elementos que son correctamente descartados. Esta se puede expresar como:

$$\text{Especificidad} = \frac{TN}{TN + FP} \quad (2.26)$$

donde  $TN$  indica la cantidad de verdaderos negativos o elementos correctamente descartados y  $FP$  indica la cantidad de falsos positivos o elementos incorrectamente identificados.

### 2.5.4. Precisión

La precisión de un clasificador binario, también conocida como el valor positivo predicho, permite determinar del total de elementos identificados como pertenecientes a determinada clase, qué porcentaje realmente pertenece a esa clase. Esto se puede expresar como:

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (2.27)$$

donde  $TP$  indica la cantidad de verdaderos positivos o elementos correctamente identificado y  $FP$  indica la cantidad de falsos positivos o elementos incorrectamente identificados.

Desde el punto de vista de la matriz de confusión, al normalizar por columnas se está mostrando la precisión de un clasificador n-ario, ya que con esta normalización se está presentando cada resultado como el porcentaje de elementos pertenecientes a determinada clase dentro del total de elementos clasificados como pertenecientes a esa clase.

### 2.5.5. Espacios ROC

Un espacio ROC (Receiver Operating Characteristic por sus siglas en inglés) es una representación gráfica de la sensibilidad o tasa de verdaderos positivos (TPR) en función de  $(1 - \text{especificidad})$  o tasa de falsos positivos (FPR) para un sistema clasificador binario donde cada punto en el espacio ROC representa un resultado de la predicción después de la variación de un umbral de discriminación. Con el análisis de la posición de los puntos en el gráfico se pueden encontrar modelos óptimos o descartar no óptimos al variar el umbral.

La figura 2.11 muestra un ejemplo de un espacio ROC. Como se indica en la figura, entre mayor sea el TPR y menor el FPR, mejor se considera el clasificador, ya que es este el caso en que se da la mayor tasa de elementos correctamente identificados unido a la menor cantidad de elementos erróneamente descartados. Por otro lado, mientras mayor

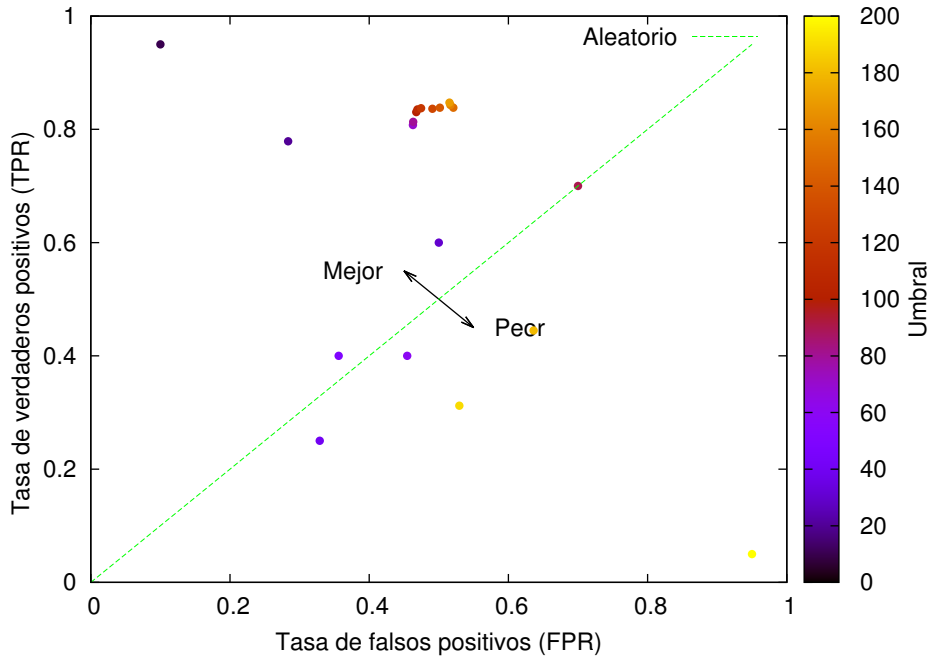


Figura 2.11: Ejemplo de un espacio ROC

sea el FPR y menor el TPR, peor se considera el clasificador, ya que en este caso es donde se presenta la mayor cantidad de elementos erróneamente descartados unido a la menor cantidad de elementos correctamente identificados. Finalmente, los casos que se encuentran cerca de la línea punteada central (similar TPR y FPR) se consideran como decisiones aleatorias, ya que se cuenta con una tasa de elementos correctamente identificados similar a la tasa de elementos erróneamente descartados. Así para el ejemplo mostrado en la figura 2.11 el mejor resultado se presenta para el umbral menor utilizado mientras los peores resultados se presentan para los umbrales altos.

### 2.5.6. Frente de Pareto

Los problemas de optimización multiobjetivo (MOP) son aquellos en los cuales múltiples criterios deben ser satisfechos al mismo tiempo [24]. Matemáticamente hablando, un MOP minimiza los componentes de un vector  $f(\underline{\mathbf{x}})$  donde  $\underline{\mathbf{x}}$  es un vector de variables de decisión  $n$ -dimensional en algún universo  $\mathcal{U}$ . En general[34]

$$\begin{aligned} &\text{minimiza } f(\underline{\mathbf{x}}) = (f_1(\underline{\mathbf{x}}), \dots, f_p(\underline{\mathbf{x}})) \\ &\text{sujeto a } g_i(\underline{\mathbf{x}}) \leq 0, i = 1, \dots, m \end{aligned} \quad (2.28)$$

por lo que un MOP consiste en  $n$  variables,  $m$  restricciones y  $p$  objetivos ( $p \geq 2$ ).

Múltiples métodos fueron desarrollados para determinar las soluciones óptimas dados los objetivos y restricciones de un MOP, sin embargo la optimalidad de Pareto es el punto clave entre todos[34]. Antes de definir la optimalidad de Pareto es necesario definir la

dominancia de Pareto que matemáticamente se escribe como

$$\forall i \in \{1, \dots, p\}, u_i \leq v_i \wedge \exists i \in \{1, \dots, p\} : u_i < v_i \quad (2.29)$$

o lo que es igual, un vector  $\mathbf{u} = (u_1, \dots, u_p)$  se dice que domina a  $\mathbf{v} = (v_1, \dots, v_p)$  si y solo si  $\mathbf{u}$  es parcialmente menor que  $\mathbf{v}$ . Sabiendo esto, una solución  $x_u \in \mathcal{U}$  se dice ser Pareto óptima si y solo si no hay un  $x_v \in \mathcal{U}$  para el cual  $v = f(x_v) = (v_1, \dots, v_p)$  domina  $u = f(x_u) = (u_1, \dots, u_p)$ [34].

Las soluciones óptimas de Pareto son también llamadas no inferiores, admisibles o eficientes y sus correspondientes vectores son llamados no dominados, cuya expresión, al ser graficada en un espacio de criterio, es conocida como frente de Pareto[34]. La figura 2.12 ilustra un frente de Pareto.

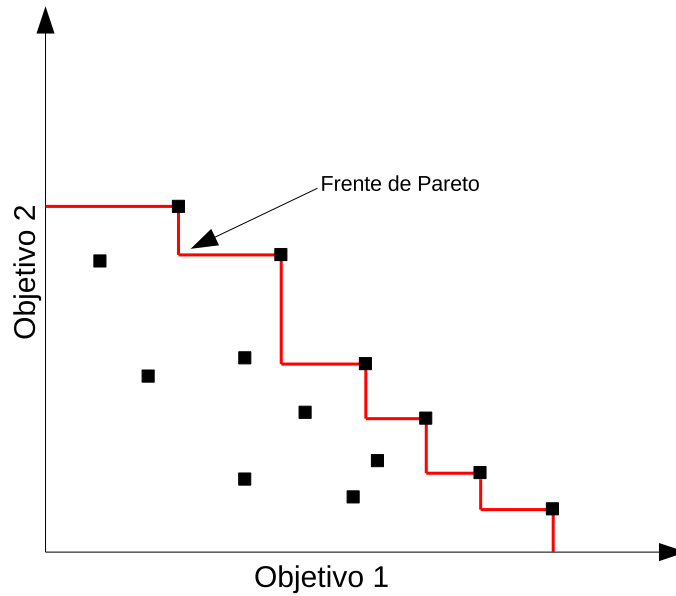


Figura 2.12: Ejemplo de un frente de Pareto

## 2.6. Estado del arte

El problema de la detección de manos para la interacción humano-máquina se ha enfrentado de distintas maneras. Desde la utilización de guantes [37] hasta la implementación de sistemas complejos con múltiples cámaras [21] que necesitan de calibración. La utilización de dispositivos externos, como lo puede ser un guante, tal y como es presentado en [37] facilita la detección, sin embargo, al ser dependiente del color es susceptible a factores ambientales como lo son los cambios en la iluminación.

En [26] se realiza el seguimiento de dos manos, las cuales son separadas del fondo de la imagen utilizando segmentación de color y el seguimiento de estas se realiza mediante la

utilización de filtros de Kalman. No se requieren de dispositivos en las manos, sin embargo son dependientes del color, lo cuál limita sus aplicaciones.

Sistemas como el presentado en [21] vienen a presentar una propuesta diferente al utilizar dos cámaras para la detección de las manos pero aun es dependiente del color. En ella el fondo de la imagen en ambas cámaras es substraído mediante segmentación por color, posteriormente ambas manos son detectadas de forma independiente para finalmente mezclar ambos resultados.

Desde el surgimiento de sensores de profundidad tales como el Kinect de Microsoft en 2010 [38] se cuenta con una nueva forma de enfrentar el problema. Con este tipo de sensores, la iluminación del ambiente ya no viene a ser un factor determinante en la detección.

Numerosos trabajos han sido presentados en este campo, desde sistemas donde se asume a un único usuario sentado frente a la cámara [1, 13, 18, 32, 12] hasta sistemas donde se realiza una detección de distintas partes del cuerpo [28, 29, 3].

En sistemas como el presentado en [1] se asume que la persona está sentada frente a la cámara, sin nada más frente a ella y se ubica la mano como el punto más cercano. Este enfoque presenta una baja complejidad y debido a sus restricciones, sus posibles aplicaciones son limitadas.

Por otra parte, en [13], se presenta un sistema en el que de igual forma se asume que el usuario está sentado frente a la cámara y solo la parte superior del cuerpo está presente en la toma. Para la detección de la mano se realiza una extracción del fondo de la imagen de forma que solo el cuerpo queda en esta, con esta nube de puntos se realiza un grafo con el cuál se calcula un histograma de distancias y debido a las restricciones impuestas, se asume la mano como el 1 % de los puntos más lejanos en el histograma.

En [19] se asume que la persona está frente a la cámara. Mediante el uso de distancias geodésicas se encuentran puntos extremos en la nube de puntos (candidatos a manos, cabeza y pies). En estos puntos se extraen parches (regiones de la imagen de  $40 \times 40$  píxeles) los cuales son clasificados mediante el uso de un clasificador previamente entrenado con parches teóricos. La figura 2.13 muestra ejemplos de los parches utilizados.

En otros sistemas se requiere de la interacción del usuario para su inicialización. En [6], el seguimiento de la mano se realiza con el usuario realizando un movimiento con sus dedos frente a la cámara el cual es detectado y a partir de este se deja crecer una región, la cual es rastreada a lo largo de la secuencia de imágenes. Así mismo, en propuestas como la presentada en [3] se requiere que el usuario se coloque en una posición de T para inicializar el sistema. En esta posición se localizan los puntos extremos (cabeza, manos y pies) y a partir de ahí se realiza el seguimiento.

En propuestas como la de [33] se hace uso de una combinación de información de color y de profundidad. En este se asume que el sujeto está sentado frente a la cámara. Para la detección de las manos se hace uso de una combinación de segmentación de color y de imágenes de profundidad. Se parte de la detección de la cara de la cual se toma el color de la piel para actualizar, en tiempo real, el algoritmo de segmentación de color pre-entrenado

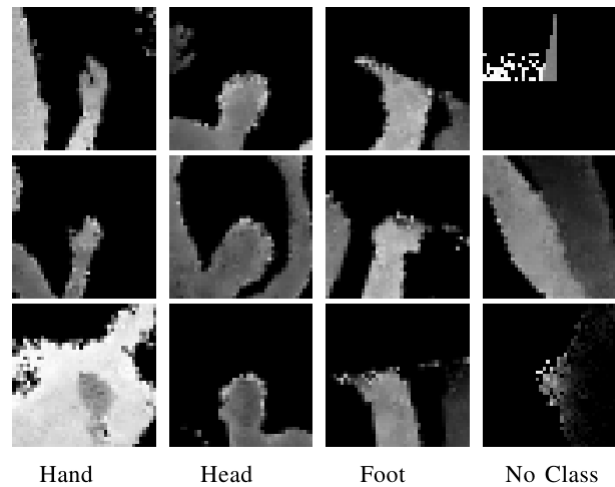


Figura 2.13: Ejemplo de patches utilizados en [19]

para la detección de las manos que, con ayuda de la información de profundidad, logra diferenciar la mano de otras partes del cuerpo con color similar de piel como lo es la cara.

En otros trabajos como el descrito en [31] se requiere de información de los cuadros anteriores para realizar el seguimiento de la mano lo cual lo hace susceptible a perder el rastro de esta.

En sistemas como el presentado en [15] se asume que la persona está frente a la cámara y que esta ocupa la mayor parte de la escena. Mediante el uso de componentes conectados se segmenta la escena y se asume el cuerpo como el componente más grande en la escena. Una vez realizado esto se realiza el seguimiento de las manos utilizando información de los cuadros anteriores.

Además de estos sistemas se presentan propuestas más complejas que se basan en el uso de clasificadores pre-entrenados para segmentar las distintas partes del cuerpo [16, 28, 29, 12, 32, 27].

En [32] se asume que el usuario está frente a la cámara y se entrena un bosque de decisión aleatoria para separar las manos del fondo de la imagen. Por otra parte, en implementaciones como la propuesta en [16], para la clasificación de las manos, como descriptor se utiliza un histograma basado en distancias polares como el mostrado en la figura 2.14.

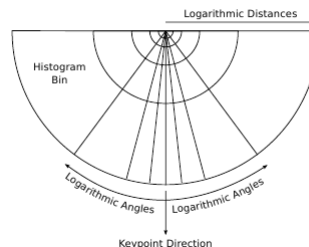


Figura 2.14: Ilustración del descriptor de mano propuesto en [16]

En el procedimiento descrito se define una distancia máxima  $d_{max}$  para la cual un punto se considera como local. Teniendo esto, se calcula un vector de dirección  $k$  (keypoint) que apunta en la dirección del promedio de los puntos locales. Para cada uno de los puntos locales se calcula un vector de dirección en 2D con respecto a  $k$  para finalmente construir el histograma con todos aquellos puntos cuya distancia no sea mayor a  $\frac{\pi}{2}$  de  $k$  ya que se asume que la información de importancia está contenida en la dirección de  $k$ . Una vez calculado el descriptor se procede a la clasificación.

En [28, 29] como descriptor se utiliza la diferencia de profundidad entre dos pixeles a una distancia dada del pixel  $q$  como se describe en

$$f_{\theta}(I, q) = d_I \left( q + \frac{u}{d_I(q)} \right) - d_I \left( q + \frac{v}{d_I(q)} \right) \quad (2.30)$$

donde  $\theta = (u, v)$  describe los desplazamientos  $u$  y  $v$  y  $d_I(q)$  representa la profundidad en el pixel  $q$  de la imagen  $I$ . El factor  $\frac{1}{d_I(q)}$  da al descriptor invarianza a la la profundidad. La figura 2.15 ilustra este descriptor.

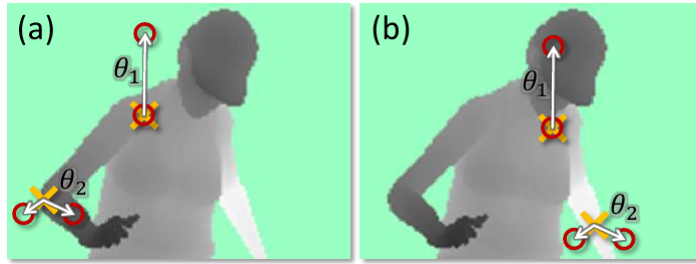


Figura 2.15: Ilustración de descriptor usado por [28]

La clasificación de las distintas partes del cuerpo se realiza haciendo uso de bosques de decisión aleatoria.

En propuestas como la presentada en [35] se utiliza un procedimiento similar, solo que en lugar de pixeles individuales, se utiliza el area en ventanas rectangulares. Por otra parte, en [36] se propone un método basado en etapas donde también se hace uso de ventanas rectangulares. Las primeras etapas, con una menor complejidad, tienen la finalidad de descartar la mayor cantidad posible de objetos, para en etapas posteriores más complejas, reafinar la detección.

Finalmente, en uno de los trabajos más recientes [27], para la detección se parte de la posición brindada por el mismo Kinect. Esta posición se toma como una aproximación inicial a partir de la cual, mediante el uso de un clasificador basado en pixeles, se segmenta la región en mano y brazo para su posterior extracción de la escena. Para realizar la segmentación el clasificador es aplicado a una región dentro de un radio en 3D a partir de la posición inicial brindada por el sensor.

En este proyecto se hace uso de una combinación de algunas de las técnicas hasta ahora utilizadas para la detección y extracción de manos en imágenes de profundidad, eliminando

restricciones como lo son el color, la iluminación o la cercanía del usuario frente a la cámara. En la siguiente sección se presenta una descripción del sistema implementado, sin embargo, ninguno de los sistemas expuestos pueden lidiar con los problemas impuestos por la rotación de las personas en vistas superiores.





## Capítulo 3

# Sistema de detección de manos en imágenes de profundidad

El sistema propuesto está compuesto por cinco etapas: segmentación de la escena a partir de la imagen de profundidad, validación de regiones de mano a partir de la predicción, búsqueda de clases candidatas a manos, extracción de descriptores y clasificación mediante máquinas de vectores de soporte (SVM por sus siglas en inglés) [5]. La figura 3.1 muestra gráficamente la estructura del sistema.

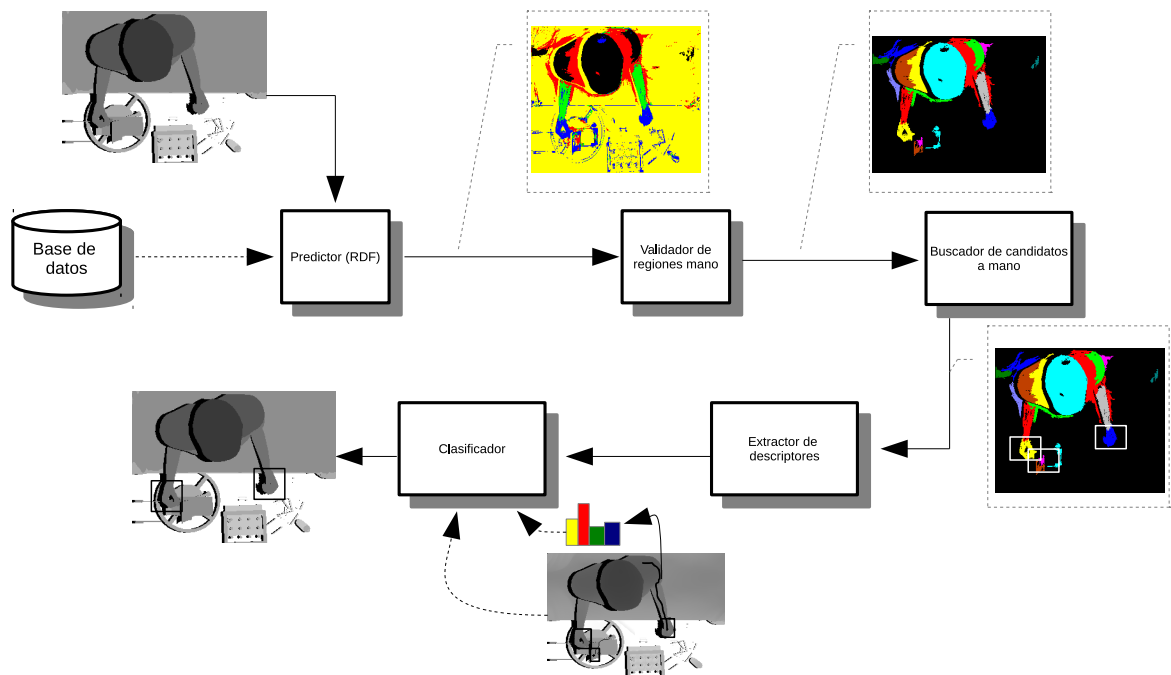


Figura 3.1: Diagrama de bloques de la solución propuesta

A continuación se presenta una descripción de cada una de las etapas que comprenden la implementación.

### 3.1. Base de datos para entrenamiento

Para segmentar la escena se utiliza como predictor un bosque de decisión aleatorio que requiere un cuerpo de datos para su entrenamiento. En este proyecto se trabaja con una base de datos de 80000 imágenes sintéticas de 12 individuos de ambos géneros todos con diferentes texturas físicas en 2 diferentes escenas. Estas imágenes tienen una resolución de  $320 \times 240$ , la cámara se encuentra a una distancia de 2,5 m del suelo con un campo visual horizontal de  $57^\circ$  y vertical de  $43^\circ$ .

Cada una de las tomas está compuesta por 2 imágenes: la imagen de profundidad y una imagen con etiquetas de las clases en que se ha subdividido la escena. Para este proyecto la escena se subdivide en cinco clases: cabeza, cuerpo, brazos, manos y fondo.

La figura 3.2 muestra un ejemplo de la subdivisión de la escena en las cinco clases. En esta se puede ver la cabeza en color rojo, las manos en color amarillo, los brazos en color azul, el cuerpo en color verde y el fondo en color cian.

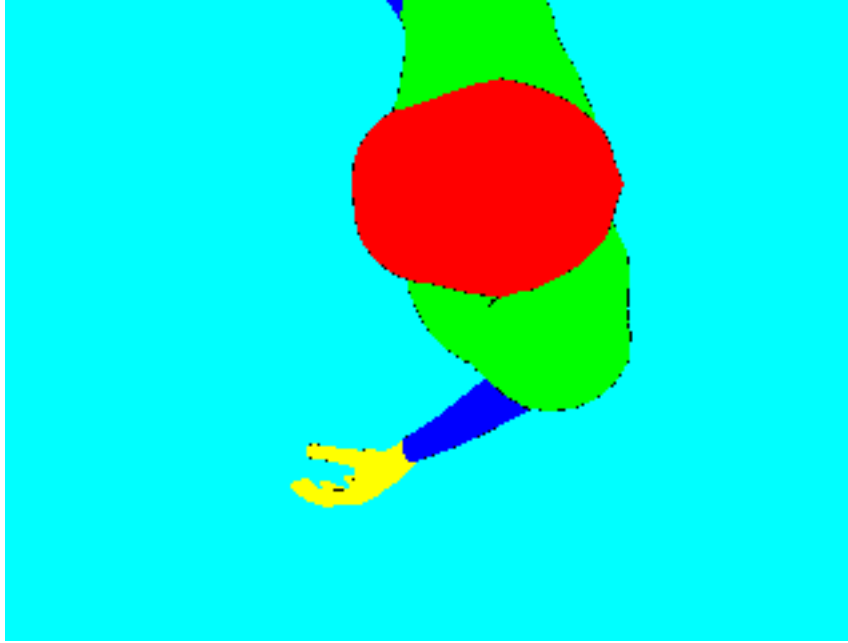


Figura 3.2: Subdivisión de la escena en cinco clases

base de datos, una sin ruido y una agregando ruido gaussiano a la imagen de profundidad con el fin de evaluar el desempeño del sistema ante imágenes más cercanas a las que se obtendrían de un sensor real.

Para la generación de la base de datos de imágenes con ruido se hace uso del procedimiento descrito en [7]. En ella la desviación estandar del error está descrita por

$$\sigma(i, j, z) = \beta_1 j^2 + \beta_2 i^2 + \beta_3 z^2 + \beta_4 j i + \beta_5 i z + \beta_6 z j + \beta_7 j + \beta_8 i + \beta_9 z + \beta_{10} \quad (3.1)$$

donde  $(i, j)$  corresponde a la posición del pixel dentro de la imagen y  $z$  corresponde al valor de la profundidad en la posición  $(i, j)$ . Además, ya que el error se considera igual en

	x	y	z
$\beta_1$	$6,3801e^{-01}$	$6,3038e^{-01}$	$2,0000e^{-05}$
$\beta_2$	$1,1225e^{-01}$	$2,6496e^{-01}$	$2,0000e^{-05}$
$\beta_3$	$3,5751e^{-06}$	$1,3279e^{-06}$	$1,2500e^{-06}$
$\beta_4$	$-4,0645e^{-03}$	$1,5000e^{-02}$	$2,0000e^{-06}$
$\beta_5$	$-1,4951e^{-04}$	$9,0174e^{-05}$	$3,5000e^{-09}$
$\beta_6$	$7,0336e^{-05}$	$3,3417e^{-04}$	$3,5000e^{-09}$
$\beta_7$	$-5,6762e^{+00}$	$-5,9320e^{+00}$	$-1,0002e^{-02}$
$\beta_8$	$-8,0153e^{-01}$	$-2,4411e^{+00}$	$-1,0002e^{-02}$
$\beta_9$	$-3,1496e^{-03}$	$3,1239e^{-03}$	$-1,5025e^{-03}$
$\beta_{10}$	$1,2996e^{+01}$	$1,0995e^{+01}$	$1,4515e^{+00}$

**Tabla 3.1:** Coeficientes para el modelo de ruido en  $x$ ,  $y$  y  $z$ [7]

todas las regiones, la posición del pixel es traducida en índices de región y se denotan por

$$i_r = \left\lceil \frac{i}{N_v/N_{vr}} \right\rceil \quad (3.2)$$

$$j_r = \left\lceil \frac{j}{N_h/N_{hr}} \right\rceil \quad (3.3)$$

donde  $N_h$  y  $N_v$  corresponden al número de pixeles horizontales y verticales de la imagen respectivamente y  $N_{hr}$  y  $N_{vr}$  son el número de regiones en las que se divide el campo visual horizontal y vertical respectivamente, que en el caso de [7], es 8. La tabla 3.1 muestra los valores de  $\beta$  utilizados para el cálculo del error en cada una de las dimensiones de la imagen.

En este proyecto, con el fin de evaluar el desempeño del sistema ante imágenes con ruido, se considera únicamente la componente de ruido en  $z$ .

## 3.2. Entrenamiento de los bosques de decisión aleatoria

Esta etapa genera los árboles que serán utilizados para la segmentación de la escena en etapas posteriores.

Cada característica  $f_\theta$ , para un pixel dado  $q$  se calcula como la diferencia del valor promedio en dos regiones rectangulares  $R_1$  y  $R_2$  en la vecindad de  $q$  como se ve en

$$f_\theta(q) := \frac{1}{|R_1(q)|} \sum_{p \in R_1} p - \frac{1}{|R_2(q)|} \sum_{p \in R_2} p \quad (3.4)$$

Estas características están definidas por un conjunto de parámetros: desplazamiento (en  $x$  y  $y$ ), ancho ( $w$ ) y largo ( $h$ ). Estos parámetros están limitados por el tamaño máximo de la región ( $RS$ ), el cuál define el rango del cuál se toman las dimensiones  $w$  y  $h$  de las cajas o regiones  $R_1$  y  $R_2$  siendo este  $[1, RS]$ . El otro parámetro es el desplazamiento máximo de caja ( $BD$ ), el cual define el rango del cual se obtiene el desplazamiento de las regiones  $R_1$  y  $R_2$  ( $\underline{O}_1$  y  $\underline{O}_2$ ) sumando a cada componente  $x$  y  $y$  de  $q$  un valor seleccionado del rango  $[-BD, +BD]$ . Así, al momento del entrenamiento, para cada característica se eligen los parámetros de manera aleatoria limitados por el tamaño máximo de región y desplazamiento máximo de caja seleccionados.

En el caso de imágenes de profundidad, en zonas más lejanas a la cámara es necesario utilizar regiones más pequeñas para representar áreas similares en la escena real. Por eso cada componente  $\underline{O}_i$ ,  $w_i$  y  $h_i$  con  $i \in \{1, 2\}$  es normalizada por la profundidad del pixel  $d(q)$ . La figura 3.3 ilustra gráficamente las dos regiones que comprenden la característica de un pixel dado.

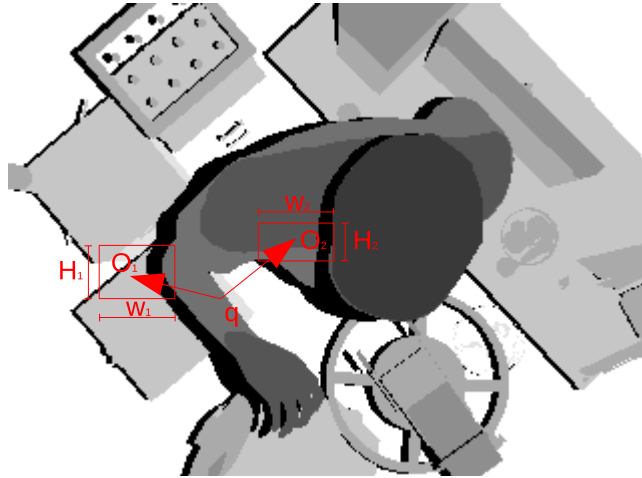


Figura 3.3: Características usadas por el RDF[25]

Debido a que la cantidad de píxeles necesarios para representar determinada área en la escena real varía de acuerdo con la profundidad, estos parámetros no pueden ser expresados en píxeles, es por eso que se introduce como unidad el pixel metro o abreviando pxm, la cual asegura que un mismo valor de píxeles metro representa la misma área en la escena sin importar la profundidad. Esta unidad es el equivalente de multiplicar la cantidad de píxeles por la profundidad en metros. Así por ejemplo, si se hace referencia a un valor de 20 pxm a una profundidad de 1 m es el equivalente a 20 píxeles mientras a una profundidad de 2 m es el equivalente a  $20 \text{ pxm} / 2 \text{ m} = 10$  píxeles.

En esta etapa se entrena el bosque para identificar cada una de las 5 clases en las que se subdivide la escena. Para la generación de los bosques de decisión se selecciona un total de 2000 imágenes de forma aleatoria de la base de datos para cada árbol. Esta cantidad de imágenes es el máximo posible ya que está limitada por la cantidad de memoria RAM disponible en la computadora en la que se realiza, en este caso 8GB.

### 3.3. Predicción de clases

Esta es la primera etapa de la detección de las manos donde se segmenta la imagen de profundidad en las 5 clases en las que se subdivide la escena. Se hace uso de los árboles previamente generados para predecir la clase a la cual pertenece cada pixel analizado donde se tiene como entrada la imagen de profundidad y haciendo uso del RDF, esta es segmentada, teniendo como resultado una matriz de las mismas dimensiones de la imagen de profundidad donde cada posición indica la clase a la cual pertenece el pixel en la misma posición en la imagen de profundidad.

Cada nodo del árbol tiene una característica asociada, la cual es calculada para cada uno de los pixeles de la imagen como se muestra en (3.4). Este resultado es analizado por cada uno de los árboles obteniéndose como resultado en las hojas ( $l_n$ ) la probabilidad  $p(c|l_n)$  de que el pixel analizado corresponda a la clase  $c \in \underline{\mathbf{C}} = \{mano, brazo, cabeza, cuerpo, fondo\}$ . De esta forma, al combinar las probabilidades obtenidas en cada uno de los árboles (ecuación (2.2)) se obtiene una predicción para el pixel analizado.

### 3.4. Validación de regiones de mano

Después de la etapa de predicción se tiene una matriz con la identificación de cada clase. Sin embargo, debido a errores en la predicción, no todos los pixeles son correctamente identificados. Con esta etapa se busca reducir el efecto de ruido por clasificaciones erróneas para continuar con las predicciones con mayores probabilidades de ser una mano. Para esto se hace uso de la técnica de componentes conectados (sección 2.2).

Dos pixeles se agrupan en un mismo objeto si son vecinos y son identificados como pertenecientes a la misma clase. Al buscar componentes conectados se eliminan todos aquellos pixeles aislados o grupos de pixeles (objetos) cuya cantidad de pixeles se encuentre por debajo de un umbral dado  $\tau_s$ .

Para este proyecto se determina experimentalmente el valor óptimo de  $\tau_s$  para el cual se logra reducir el efecto de ruido por clasificaciones erróneas. Una vez hecho esto, se procede a aplicar componentes conectados utilizando este valor.

Sea  $\underline{\mathbf{T}}$  el conjunto de componentes conectados  $t_i$ . Si se define  $|t_i|$  como la cantidad de pixeles (área) de un determinado componente  $t_i$ , el resultado de esta etapa se puede expresar como

$$\underline{\mathbf{Q}} = \{t_i | t_i \in \underline{\mathbf{T}}, |t_i| > \tau_s\} \quad (3.5)$$

con  $\underline{\mathbf{Q}}$  siendo el conjunto de componentes resultantes.

La figura 3.4 muestra el un ejemplo después de aplicar componentes conectados.

En esta imagen cada color representa un objeto o componente diferente. Como se observa, solo hay dos objetos pequeños además del cuerpo los cuales deben ser eliminados. Esto se

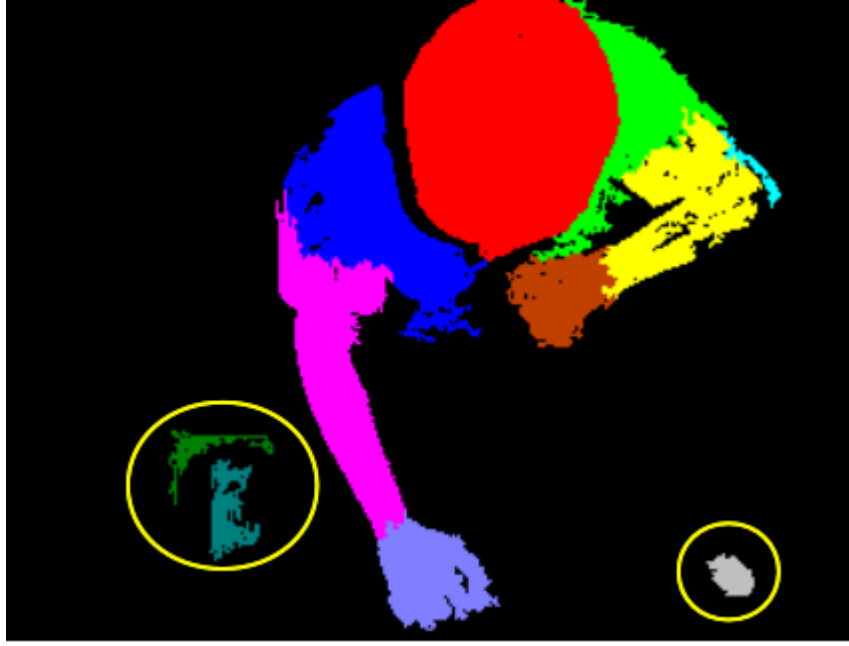


Figura 3.4: Ejemplo de validación de regiones mano

realiza en etapas posteriores.

### 3.5. Búsqueda de candidatos a mano

Una vez realizada la búsqueda de componentes conectados se tiene un conjunto de objetos (grupos de píxeles clasificados como pertenecientes a una determinada clase) y a partir de este punto se trabaja sobre esos objetos en lugar de píxeles individuales. En esta etapa se elimina del conjunto aquellos objetos marcados como mano que no tengan al menos un objeto vecino (que puede ser brazo, cabeza, cuerpo u otra mano) o que no hayan sido clasificados como mano, esto con el fin de finalizar con una lista que contenga únicamente candidatos a mano.

Sea  $\underline{\mathbf{O}} = \{o_1, o_2, \dots, o_M\}$  el conjunto de componentes obtenidos en la etapa anterior, con  $M$  la cantidad de componentes. Sea  $\underline{\mathbf{C}} = \{mano, cabeza, brazo, cuerpo, fondo\}$  el conjunto de clases a las que puede ser asignado un elemento  $o_i$ . Sea

$$\phi : \underline{\mathbf{O}} \rightarrow \underline{\mathbf{C}} \quad (3.6)$$

una función que asigna a cada elemento en  $\underline{\mathbf{O}}$ , uno y solo un elemento en  $\underline{\mathbf{C}}$ . Además sea

$$\psi : \underline{\mathbf{O}}^2 \rightarrow \{0, 1\} \quad (3.7)$$

una función que toma dos elementos de  $\underline{\mathbf{O}}$  y retorna 1 en caso de que sean vecinos o 0 de lo contrario. De esta forma se puede definir el conjunto de candidatos de esta fase como

$$\underline{\mathbf{W}} = \{x | \phi(x) = mano \wedge \exists j | \psi(x, y_j) = 1, \phi(y_j) \neq fondo\} \quad (3.8)$$

donde  $\underline{\mathbf{W}}$  contiene todos aquellos candidatos a mano clasificados como tal con al menos un vecino diferente de fondo. Eliminando todos aquellos elementos que no tienen un vecino se eliminan aquellos objetos aislados erróneamente clasificados como mano ya que toda mano debe tener un brazo y un cuerpo asociado, donde se asume que un objeto aislado tiene una alta probabilidad de no ser mano.

### 3.6. Extracción de descriptores de ruta

En esta etapa se cuenta con una lista de candidatos a mano donde algunos de ellos no corresponden a manos reales y no han podido ser filtrados. Es por eso que se hace necesaria una forma de eliminar todos esos objetos erróneamente clasificados como mano. Para lograr dicho cometido es necesaria la obtención de descriptores en cada uno de los candidatos a mano que permitan, en una posterior etapa, la separación de los objetos que realmente son una mano de los que no lo son.

Para describir los candidatos a mano se hace uso de dos características: la primera de ellas consiste en el uso del algoritmo Dijkstra [9] para la obtención de la ruta más larga a partir del centro del candidato a mano en la imagen de profundidad y a partir de esta se calcula un histograma cuyas celdas corresponden a las distintas clases en las que se clasifica la imagen de profundidad y la segunda de ellas consiste en la longitud de dicha ruta. Con estas dos características se forma un descriptor que será utilizado para la clasificación de los candidatos a mano.

#### 3.6.1. Obtención de la ruta más larga

Esta etapa consiste en la obtención de distancias geodésicas mediante la utilización del algoritmo de Dijkstra. Lo que se pretende es encontrar el camino más largo a partir del centro del candidato a mano sobre la imagen de profundidad, el cual, si se realiza sobre una mano real, partiría de la mano, subiendo por el brazo hasta llegar al cuerpo.

Para lograr dicho cometido, se hace uso del procedimiento descrito en la sección 2.3. Ya que se desconoce el punto  $V_2$  para el que se pretende calcular la distancia geodésica, se itera a partir de centro del candidato a mano ( $V_1$ ). Como criterio de parada se tiene el momento en que se encuentra un punto a una distancia máxima  $L_{max}$  determinada experimentalmente lo cual, aplicando las restricciones descritas en la sección 2.3, permite la obtención de una ruta a lo largo del brazo. En caso de no encontrarse un punto a una la distancia definida  $L_{max}$ , se utiliza el punto más lejano del centro del candidato y como ruta el camino entre ellos.

Lo anterior se puede expresar matemáticamente de la siguiente forma: sea  $\underline{\mathbf{G}} = \{g_1, g_2, \dots, g_P\}$  el conjunto de nodos  $g_i$  de un grafo y  $\underline{\mathbf{A}} = \{\underline{\mathbf{a}}_1, \underline{\mathbf{a}}_2, \dots, \underline{\mathbf{a}}_P\}$  el conjunto de aristas de un grafo con  $\underline{\mathbf{a}}_i$  el conjunto de aristas que llegan al nodo  $g_i$ . Además sea  $V_1 \in G$  y  $\underline{\mathbf{D}} = \{d_1, d_2, \dots, d_P\}$  el conjunto de distancias mínimas de  $V_1$  a cada uno de los nodos en

$\underline{\mathbf{G}}$  calculados mediante la aplicación del algoritmo de Dijkstra, donde  $d_i$  es la distancia de  $V_1$  a  $g_i$ . La distancia más larga buscada, restringida a  $L_{max}$ , es la descrita por

$$\min(d_j | d_j \geq L_{max}, \max(d_j)), \text{ con } j = 1, 2, \dots, P \quad (3.9)$$

Si además se define  $\underline{\mathbf{R}} = \{\underline{\mathbf{r}}_1, \underline{\mathbf{r}}_2, \dots, \underline{\mathbf{r}}_P\}$  como el conjunto de todas las rutas de  $V_1$  a  $g_i$ , la ruta buscada es el correspondiente  $\underline{\mathbf{r}}_j$  para  $j$  que satisfaga (3.9).

### 3.6.2. Extracción del histograma

Una vez que se obtiene la ruta más larga a partir del centro de la posible mano se procede a la generación de un histograma de clases recorriendo esta ruta donde cada celda corresponde a la cantidad de píxeles clasificados como pertenecientes a la misma clase.

Sea  $\underline{\mathbf{r}} \subset \underline{\mathbf{R}}$  el conjunto de los nodos que comprenden la ruta más larga encontrada. Además sea la función

$$\psi : \underline{\mathbf{R}} \times \underline{\mathbf{C}} \rightarrow \{0, 1\} \quad (3.10)$$

la que determina si un elemento  $\underline{\mathbf{r}}_i$  pertenece a la clase

$$c_j \in \underline{\mathbf{C}} = \{mano, cabeza, brazo, cuerpo, fondo\}$$

con 1 indicando pertenencia a la clase  $c_i$  y 0 indicando no pertenencia. Si además se define  $L$  como el histograma a generar donde  $L_{c_j}$  es la celda correspondiente a la clase  $c_j$  se tiene que

$$L_{c_j} = \sum_{i=1}^N \psi(\underline{\mathbf{r}}_i, c_j) \quad (3.11)$$

De esta forma se genera un histograma que describe la distribución de las clases a lo largo de la ruta.

## 3.7. Clasificación

La etapa final es la clasificación de los candidatos a manos como mano o no mano. Para esto se hace uso de máquinas de vectores de soporte (SVM por sus siglas en inglés). Como vectores de características se hace uso del histograma generado a partir de la ruta encontrada así como la longitud de esta y como kernel se utiliza el Gaussiano. Esta etapa consiste en dos procesos, el primero de ellos consiste en el entrenamiento de la máquina la cuál se realiza una única vez, posteriormente, con los resultados del entrenamiento se procede a la clasificación.



### 3.7.1. Entrenamiento del SVM

Para el entrenamiento del SVM se utilizan 500 imágenes seleccionadas aleatoriamente. Como entrada del algoritmo se tiene la imagen con las etiquetas teóricas y la imagen proveniente de la predicción. Utilizando la imagen con las etiquetas teóricas se entrena al SVM para discernir si un candidato a mano es o no una mano real. Además de esto, se entrena a la máquina para descartar todo candidato con un camino cuya longitud es menor a 10 cm con lo cual el SVM aprende a discernir si un candidato a mano es o no una mano real no solo por la distribución de las clases a lo largo de la ruta sino que por la longitud de esta.

### 3.7.2. Clasificación

Finalmente se hace uso de los resultados del entrenamiento del SVM para la clasificación de los candidatos a manos como mano o no mano. Como resultado de esta etapa se tiene una lista de las manos encontradas en cada una de las imágenes indicando el centro y las dimensiones de una caja que la contiene.

La figura 3.5 muestra un ejemplo de dos cajas indicando la posición de las manos. Como



Figura 3.5: Resultado de la clasificación

se observa, se ha eliminado un candidato erróneamente clasificado como mano.



# Capítulo 4

## Resultados y análisis

Los resultados de cada etapa del sistema se presentan a continuación.

### 4.1. Segmentación de la escena mediante bosques de decisión aleatoria

Como un proceso previo a la implementación del sistema se lleva a cabo el entrenamiento de los bosques de decisión aleatoria con el fin de determinar la combinación de parámetros que realiza de mejor forma la segmentación de las 5 clases en las que se divide la escena: mano, brazo, cuerpo, cabeza y fondo. La imagen 4.1 muestra un ejemplo de la división de clases.

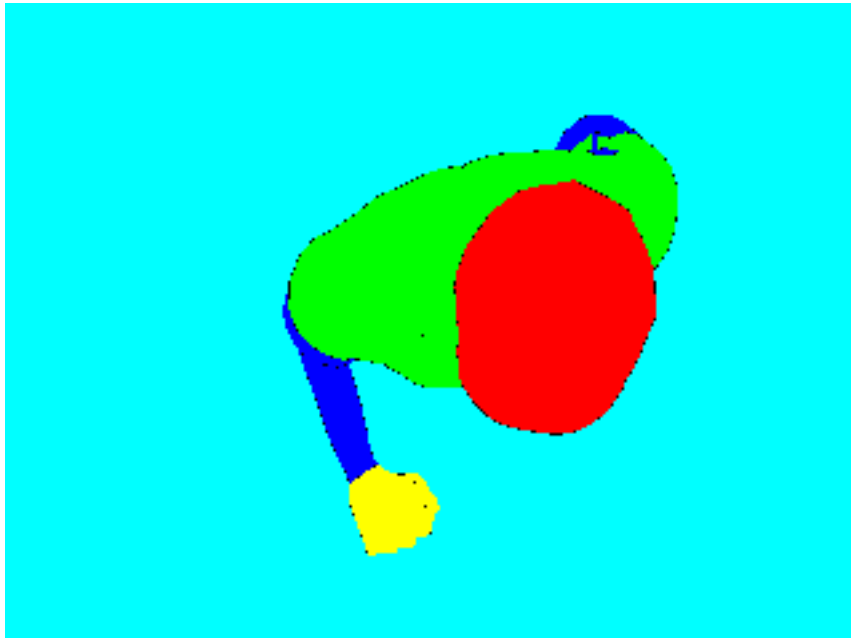


Figura 4.1: Ejemplo de división de clases

Con el objetivo de determinar la combinación óptima de parámetros se procede a variar la cantidad de árboles en el bosque así como la cantidad de niveles en cada árbol, el desplazamiento máximo de las cajas ( $BD$ ) y el tamaño máximo de región ( $RS$ ) utilizado para la obtención de los descriptores con los que se entrenan los árboles. Pero antes se procede a determinar el efecto de la cantidad de imágenes utilizadas en el entrenamiento así como la cantidad de muestras por imagen en la precisión de la detección.

Para ello se procede a realizar el entrenamiento utilizando un total de 100, 500, 1000 y 2000 imágenes con un total de 1500 muestras por imágenes para posteriormente realizar diversos entrenamientos con un total de 1000 imágenes y variando la cantidad de muestras por imagen a 500, 1000, 1500 y 2000. La figura 4.2 ilustra los resultados. Como se puede ver, la precisión en la segmentación aumenta al aumentar tanto la cantidad de imágenes como la cantidad de muestras por imagen, sin embargo, debido a limitaciones en la cantidad de memoria RAM disponible, se limita la cantidad máxima de imágenes para entrenar cada árbol a 2000 y la cantidad de muestras por imagen a 1500 y se procede a entrenar cada árbol de forma individual.

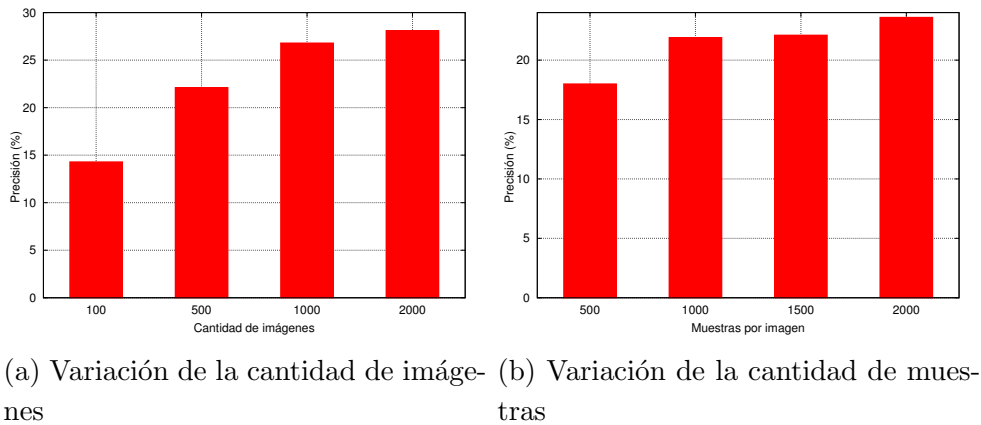


Figura 4.2: Precisión en la segmentación ante una variación en la cantidad de imágenes y muestras por imagen en el entrenamiento

Para el entrenamiento de cada árbol se toman 2000 imágenes de forma aleatoria de la base de datos con lo que para entrenar un bosque compuesto por 3 árboles se generan 3 conjuntos de 2000 imágenes cada uno. Así, el entrenamiento del bosque completo utiliza un total de 6000 imágenes.

Se entrenan los árboles utilizando imágenes sin ruido. Para la validación de los resultados del entrenamiento se utiliza un conjunto de 500 imágenes seleccionadas aleatoriamente de la base de datos utilizando para ello la totalidad de los píxeles de cada imagen.

Etiquetas/Predicción	Fondo	Brazos	Cuerpo	Manos	Cabeza
Fondo	0,914	0,013	0,028	0,040	0,005
Brazos	0,029	0,630	0,165	0,111	0,065
Cuerpo	0,061	0,063	0,704	0,022	0,151
Manos	0,048	0,133	0,058	0,713	0,048
Cabeza	0,007	0,017	0,044	0,025	0,907

**Tabla 4.1:** Sensibilidad para un  $BD$  y  $RS$  de 10 pxm

#### 4.1.1. Efecto de la variación del desplazamiento de caja y tamaño de región

Utilizando 3 árboles con 15 niveles cada árbol, se procede a variar el desplazamiento máximo de las cajas y el tamaño máximo de la región que comprenden las características  $f_\theta$  a extraer de las imágenes, esto con el fin de observar el comportamiento de los árboles ante la variación de estos parámetros. Una vez determinada la mejor combinación de parámetros se procede a incrementar la cantidad de niveles y finalmente la cantidad de árboles utilizados con el fin de determinar su influencia en el resultado de la predicción. Para la evaluación se utilizan matrices de confusión.

La figura 4.3 muestra una imagen predicha utilizando un  $BD$  y  $RS$  de 10 pxm para ilustrar cualitativamente los resultados. La tabla 4.1 muestra la sensibilidad de la predicción. En el caso de las manos, de la totalidad de los pixeles etiquetados como mano en todo el conjunto de evaluación, un 71,3 % fue correctamente detectado.



**Figura 4.3:** Predicción para un  $BD$  y  $RS$  de 10 pxm utilizando 3 árboles

La tabla 4.2 muestra la precisión de la predicción. En el caso de las manos, de la totalidad

Etiquetas/Predicción	Fondo	Brazos	Cuerpo	Manos	Cabeza
Fondo	0,994	0,412	0,372	0,750	0,089
Brazos	0,001	0,384	0,041	0,038	0,021
Cuerpo	0,004	0,123	0,550	0,024	0,151
Manos	0,001	0,055	0,010	0,167	0,010
Cabeza	0,000	0,027	0,028	0,022	0,729

**Tabla 4.2:** Precisión para un  $BD$  y  $RS$  de 10 pxm

Etiquetas/Predicción	Brazos	Cuerpo	Manos	Cabeza
Brazos	0,653	0,065	0,152	0,023
Cuerpo	0,209	0,876	0,094	0,166
Manos	0,094	0,015	0,667	0,011
Cabeza	0,045	0,044	0,087	0,800

**Tabla 4.3:** Precisión para un  $BD$  y  $RS$  de 10pxm sin tomar en cuenta la clase fondo

de pixeles reconocidos como manos, únicamente el 16,7 % corresponde realmente a mano lo que deja un 83,3 % de falsos positivos, lo cual se observa en la figura 4.3 como los pixeles de color azul fuera de las manos.

Si no se considera el fondo de la imagen para la obtención de la precisión en la predicción (ya que el ruido de este debido a errores en la detección se disminuye en etapas posteriores), el porcentaje de falsos positivos disminuye a un 43,3 % lo cual indica que la mayor cantidad de estos resultados están ubicados en el fondo de la imagen. La tabla 4.3 muestra los resultados mostrados en la tabla 4.2 sin considerar la clase fondo. Un 71,3 % de sensibilidad en la detección es suficiente para etapas posteriores ya que lo que se requiere es tener grupos de pixles correctamente asignados a mano lo cual es conseguido con ese resultado.

Una vez obtenidos estos resultados, se procede a aumentar  $BD$  y  $RS$  a 20 pxm con el objetivo de determinar el efecto de tener una mayor cantidad de pixeles en la región a utilizar para el cálculo de las características. Las tablas 4.4 y 4.5 muestran las matrices de confusión para este caso.

Etiquetas/Predicción	Fondo	Brazos	Cuerpo	Manos	Cabeza
Fondo	0,891	0,025	0,036	0,036	0,011
Brazos	0,033	0,677	0,135	0,113	0,042
Cuerpo	0,064	0,049	0,789	0,010	0,087
Manos	0,043	0,100	0,047	0,763	0,047
Cabeza	0,008	0,010	0,028	0,018	0,937

**Tabla 4.4:** Sensibilidad para un  $BD$  y  $RS$  de 20 pxm

Etiquetas/Predicción	Fondo	Brazos	Cuerpo	Manos	Cabeza
Fondo	0,994	0,595	0,416	0,732	0,184
Brazos	0,001	0,295	0,029	0,043	0,013
Cuerpo	0,004	0,069	0,533	0,012	0,082
Manos	0,001	0,030	0,007	0,196	0,009
Cabeza	0,000	0,011	0,015	0,017	0,712

**Tabla 4.5:** Precisión para un  $BD$  y  $RS$  de 20 pxm

Como se observa en la tabla 4.4 la sensibilidad dentro de la clase mano fue mayor al caso de un  $RS$  de 10 pxm (76,3 %). Además, como se ve en la tabla 4.5, el porcentaje de precisión para la clase mano aumentó a un 19,6 % lo cual indica una mejor clasificación disminuyendo la cantidad de falsos positivos de un 83,3 % a un 80,4 %. Además, si no se toma en cuenta la clase fondo, el porcentaje de falsos positivos corresponde al 26,8 %. Estos resultados indican que la mayor parte de los falsos positivos se encuentran distribuidos en el fondo de la imagen, los cuales, como se puede ver, corresponden a grupos aislados de píxeles y ya que en etapas posteriores del sistema se eliminan estos grupos, estos porcentajes son suficientes para continuar. Esto se ilustra en la figura 4.4.

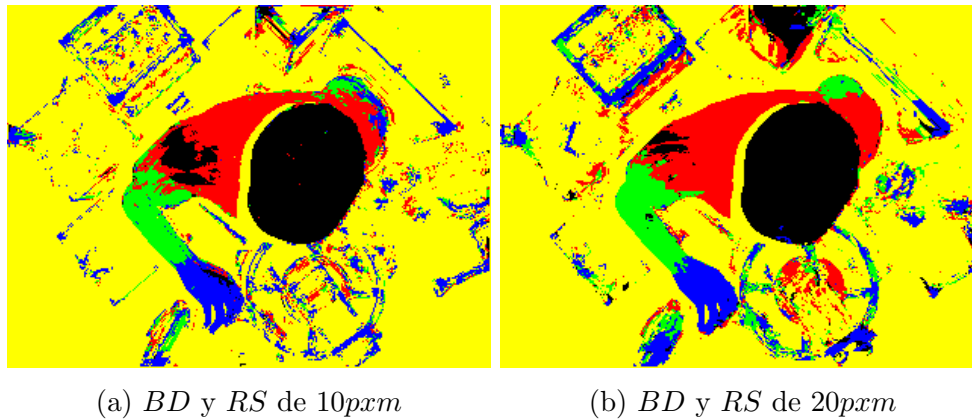


Figura 4.4: Predicción para distintos  $BD$  y  $RS$

Como se puede ver, la cantidad de píxeles azules (mano) fuera de las manos disminuye en relación con el caso de un tamaño de región de 10 pxm (figura 4.4a) lo cual concuerda con el resultado mostrado por la matriz de confusión.

Las figuras 4.5a y 4.5b resumen los resultados tras variar  $RS$  y  $BD$  para un bosque con tres árboles y un total de 15 niveles. Como se puede observar en la figura 4.5a, la combinación  $RS$ - $BD$  que produjo el mayor porcentaje de sensibilidad (76,7 %), es decir, la combinación con la cual se detectó la mayor cantidad de píxeles pertenecientes a la clase mano es de un desplazamiento de caja de 20 pxm y un tamaño de región de 5 pxm. Sin embargo, para esta misma combinación, la precisión de la detección dentro de la clase mano es del 11,9 % lo cual se traduce en un 88,9 % de falsos positivos. Por otro lado, en la figura 4.5b se puede ver cómo la combinación que produjo el mejor resultado (28,1 %) fue utilizando

un  $BD$  de 10 pxm con  $RS$  de 5 pxm. Estos resultados se pueden ver de mejor manera en la figura 4.5c donde se aprecia claramente como hay 3 puntos en el frente de Pareto, siendo las combinaciones con la mejor sensibilidad (76,7 %) y la mejor precisión (28,1 %) parte de ellos. Sin embargo, ya que lo que se busca es la mayor precisión, se selecciona este último que corresponde a  $BD : RS$  de 10 pxm : 5 pxm. Estos valores significan un porcentaje de falsos positivos de 71,8 % y si no se toma en cuenta la clase fondo, se tiene un porcentaje de falsos positivos del 35,6 %. Además se detectó correctamente el 67,6 % de los pixeles pertenecientes a la clase mano lo cual es suficiente para las etapas posteriores. Este resultado se ilustra en la figura 4.6.

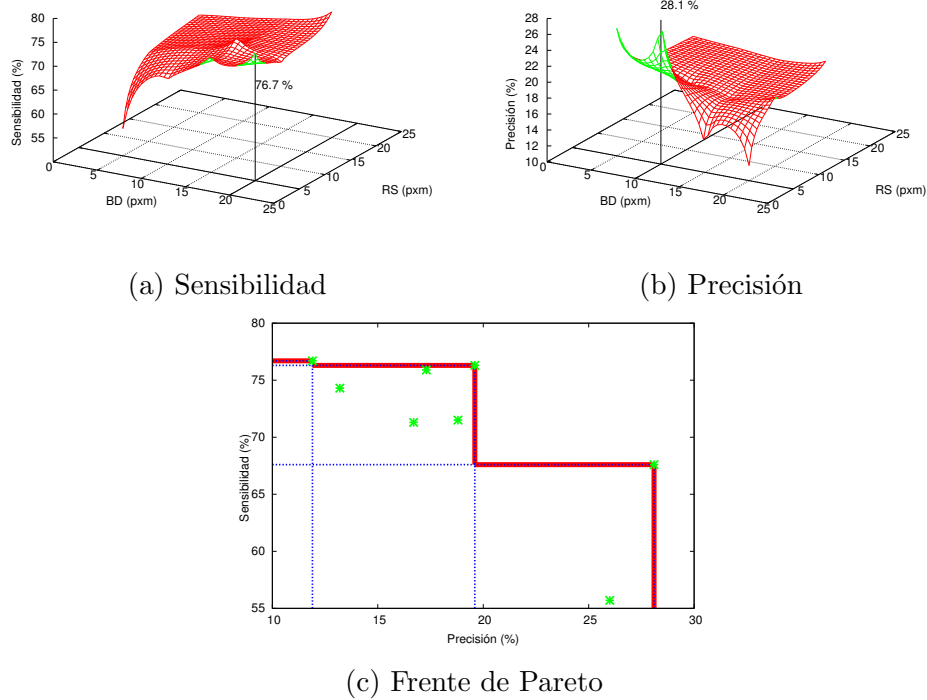


Figura 4.5: Variación de la precisión y sensibilidad en la clase mano ante un cambio en el desplazamiento máximo de caja ( $BD$ ) y el tamaño máximo de región ( $RS$ )

La figura 4.6a muestra el resultado de usar un desplazamiento máximo de caja y tamaño máximo de región de 20 pxm : 5 pxm respectivamente, mientras la figura 4.6b muestra el resultado para valores de 10 pxm : 5 pxm. Como se puede observar, aunque con el  $BD$  de 20 pxm y  $RS$  de 5 pxm se tiene un mayor porcentaje de pixeles detectados como mano, el mejor resultado se observa en la figura 4.6b, ya que este es el caso en el que se presenta una menor cantidad de falsos positivos.

La figura 4.7 muestra la distribución del desplazamiento de caja para un bosque con  $BD : RS$  de 20 pxm : 5 pxm. Como se puede ver, los desplazamientos utilizados por el árbol en todos los nodos presentan una distribución aleatoria a lo largo de todo el rango. Sabiendo esto se puede deducir como el uso de un valor de desplazamiento máximo de caja mayor ocasiona la utilización de un desplazamiento de caja ( $O_1$  y  $O_2$ ) mayor lo cual



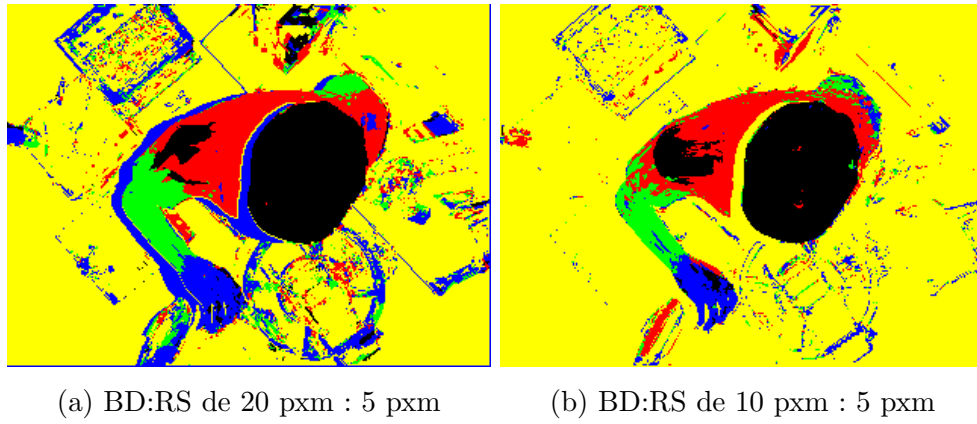


Figura 4.6: Mejores resultados para distintos  $BD$  y  $RS$  con 3 árboles y 15 niveles

explica el comportamiento observado en la figura 4.6a donde los bordes de los brazos son identificados como mano (color azul) ya que al ubicar las regiones en un área mayor ( $BD$  mayor) y al estar la mano formada en su mayoría por bordes las regiones cercanas a los brazos son identificadas como mano.



Figura 4.7: Distribución del desplazamiento de caja para un bosque con  $BD : RS$  de 20 pxm : 5 pxm

Cabe destacar que una variación en el desplazamiento máximo de caja o tamaño máximo de región a utilizar en el cálculo de las características no tiene impacto alguno en la velocidad de ejecución o predicción ya que se están usando imágenes integrales [35], por lo que el cálculo del área de cualquier región, sin importar su tamaño, requiere de un total de dos restas y una suma.

### 4.1.2. Efecto de la cantidad de niveles

Una vez determinada la mejor combinación *BD-RS* se procede a la determinación de la influencia de la cantidad de niveles del árbol en el bosque en el resultado de la predicción. En este caso se utiliza un *BD* y *RS* de 10 pxm y 5 pxm respectivamente ya que se determinó que es la combinación que produce la mejor precisión.

La figura 4.8 muestra los resultados tras variar la cantidad de niveles de cada árbol.

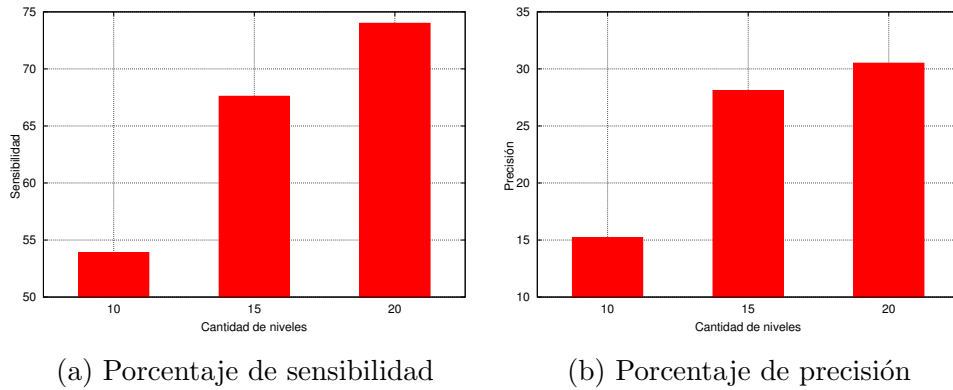


Figura 4.8: Resultados de la variación de la cantidad de niveles

Como se puede ver, al pasar de un total de 10 a 15 niveles se obtiene un incremento mayor que al pasar de 15 a 20 niveles siendo de un 2,4 % para el porcentaje de píxeles mano correctamente detectados (ver figura 4.8b). La figura 4.9 ilustra esta diferencia. La cantidad de falsos positivos es menor en la figura 4.9b que en la figura 4.9a. Sin embargo, si se compara el resultado mostrado en 4.9b con la figura 4.9c no se observa una mejora en la cantidad de píxeles erróneamente clasificados por lo que no se justifica el paso de un árbol con 15 a uno con 20 niveles.

### 4.1.3. Efecto de la cantidad de árboles

Una vez que se determina el efecto de la variación de la cantidad de niveles en la predicción se procede a determinar el efecto de la cantidad de árboles. En este caso se utiliza un total de 3, 5 y 10 árboles con un total de 15 niveles cada uno.

Las figura 4.10 ilustra los resultados obtenidos. A diferencia de los resultados obtenidos tras la variación en la cantidad de niveles, en el caso de la variación de la cantidad de árboles en el bosque presenta un aumento en la precisión menor, pasando de un 28,1 % para el caso de 3 árboles, a un 33,5 % para el caso de 10 árboles (figura 4.10b). Como se puede ver en la sección 2.1, tanto el aumento en la cantidad de árboles como el aumento en la cantidad de niveles del árbol ocasiona una mejora en la confianza y limita el error de generalización de la clasificación, sin embargo, el aumento en la cantidad de niveles puede tender a un sobreajuste mientras el aumento en la cantidad de árboles no. Esto sumado a que se obtiene un mejor resultado aumentando la cantidad de árboles (precisión

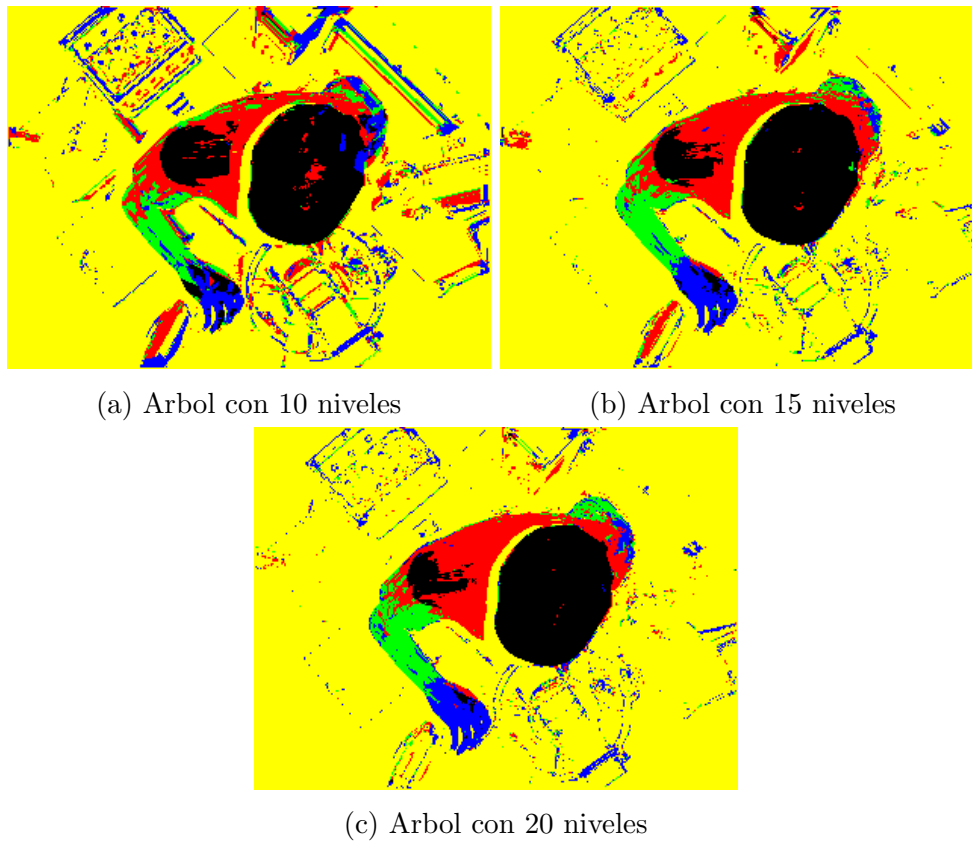


Figura 4.9: Resultado cualitativo de la variación de la cantidad de niveles

de 33,5 %) justifica el uso de 10 árboles con 15 niveles.

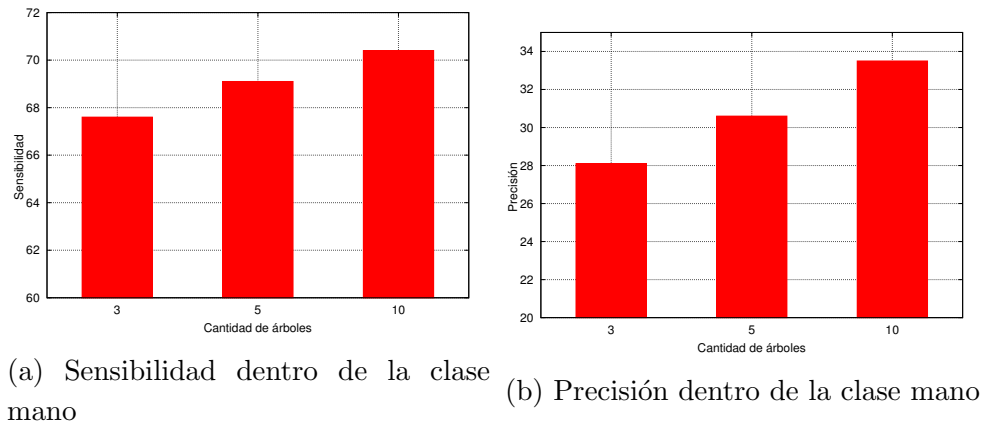


Figura 4.10: Resultados de la variación de la cantidad de árboles

La figura 4.11 muestra cualitativamente la diferencia entre el uso de 3 y 10 árboles. Como se puede ver en la figura 4.11b, la cantidad de falsos positivos es menor que en la figura 4.11a siendo de un 64,5 % si se toman en cuenta las 5 clases o de un 29,6 % si no se toma en cuenta la clase fondo.

Estos resultados se deben a que al aumentar la cantidad de árboles en el bosque se aumenta la confiabilidad de la clasificación lo cual se traduce en un aumento en la precisión del

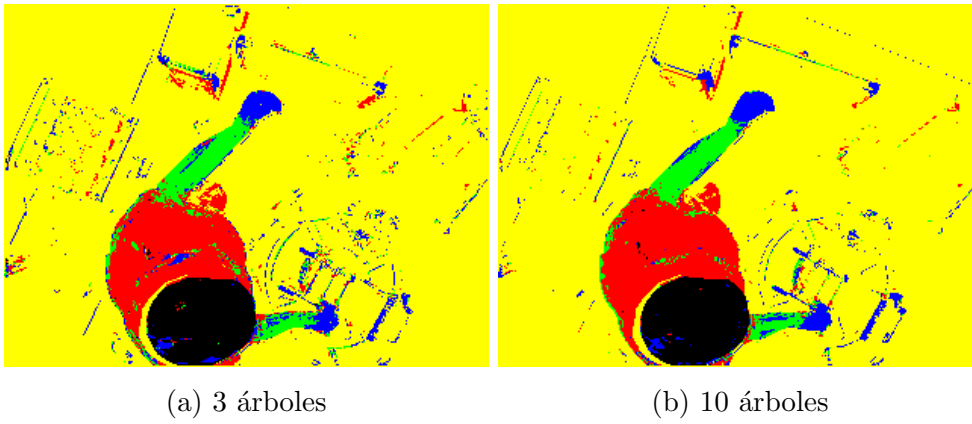


Figura 4.11: Comparación de resultados usando 3 y 10 árboles

bosque.

Un porcentaje de sensibilidad del 70,4 % con una precisión del 33,5 % como los obtenidos para un bosque con 10 árboles, con 15 niveles cada árbol y un  $BD : RS$  de 10 pxm : 5 pxm es suficiente para etapas posteriores donde se requiere de grupos de pixeles correctamente identificados como mano y se realiza un filtrado de los falsos positivos obtenidos.

## 4.2. Validación de regiones mano

En etapas anteriores se trabaja sobre pixeles, buscando aumentar la precisión en la identificación de pixeles mano. Como resultado de estas etapas, se obtiene una imagen con grupos de pixeles clasificados como mano. En esta etapa se trabaja con dichos grupos o regiones de pixeles buscando validar las regiones mano y maximizar el filtrado de los falsos positivos obtenidos.

A partir de este punto se utiliza un bosque con un total de 10 árboles, 15 niveles por árbol con un desplazamiento máximo de caja de 10 pxm y un tamaño máximo de región de 5 pxm ya que en etapas anteriores se determina que es la combinación que presenta mejores resultados.

Para la validación de regiones mano se hace uso de la técnica de componentes conectados donde se agrupan conjuntos de pixeles pertenecientes a la misma clase cuya separación es menor a un valor dado. Para esta sección se utiliza una separación entre pixeles de 0, lo cual quiere decir que para que dos pixeles sean agrupados estos, además de pertenecer a la misma clase, deben ser vecinos. Además de esto, se eliminan los grupos de pixeles que cuenten con un número de elementos menor a un umbral dado  $\tau_s$  con el objetivo de reducir la lista de candidatos a mano manteniendo en la lista la mayor cantidad posible de manos reales.

Para la determinación del umbral óptimo  $\tau_s$  que mejor separa los objetos mano de los no mano, se procede a seleccionar de forma aleatoria un total de 20000 imágenes de la base

de datos. Para cada caso se cuenta con la imagen de referencia (ground truth) y la imagen de profundidad. A cada una de las imágenes de profundidad se le realiza la predicción utilizando un bosque con la configuración descrita. Finalmente, tanto para la imagen de referencia como para la predicción, se realiza un histograma que describe la cantidad de objetos mano en función del tamaño estos. Esto con el objetivo de observar la distribución de los tamaños de manos reales y de candidatos. Además de esto, se procede a determinar la variación de la cantidad de candidatos a mano así como la de manos reales eliminadas en función del umbral utilizado para valores de  $\tau_s$  que van de 20 a 300 píxeles.

La figura 4.12 muestra los resultados tras variar el valor del umbral. Como se puede ver en la figura 4.12a, al aumentar el valor  $\tau_s$ , la cantidad de candidatos en la lista disminuye mientras la cantidad de manos reales que son eliminadas por el umbral (falsos negativos debido al tamaño de  $\tau_s$  utilizado) aumenta siendo el punto de inflexión un umbral de 100 píxeles.

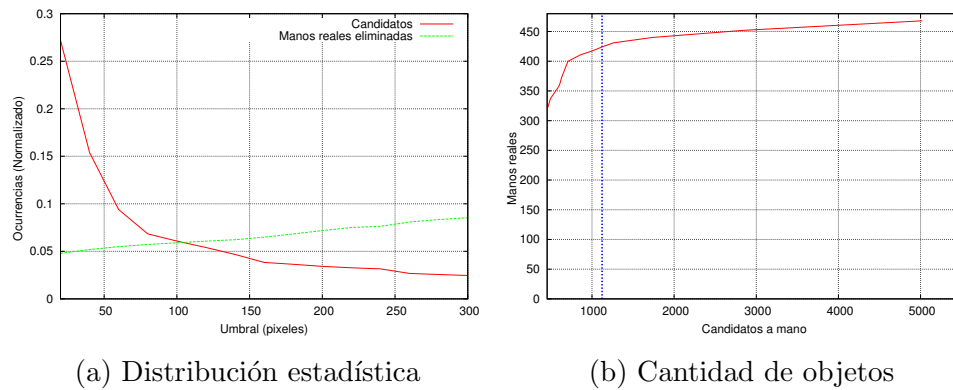


Figura 4.12: Variación de la cantidad de candidatos a mano y manos reales en función del valor de  $\tau_s$

La figura 4.13 muestra la distribución seguida por el tamaño de los objetos mano en las imágenes de referencia (figura 4.13a) así como los candidatos a mano en la predicción (4.13b). Para las imágenes de referencia se tiene un total de 26208 objetos de los cuales,

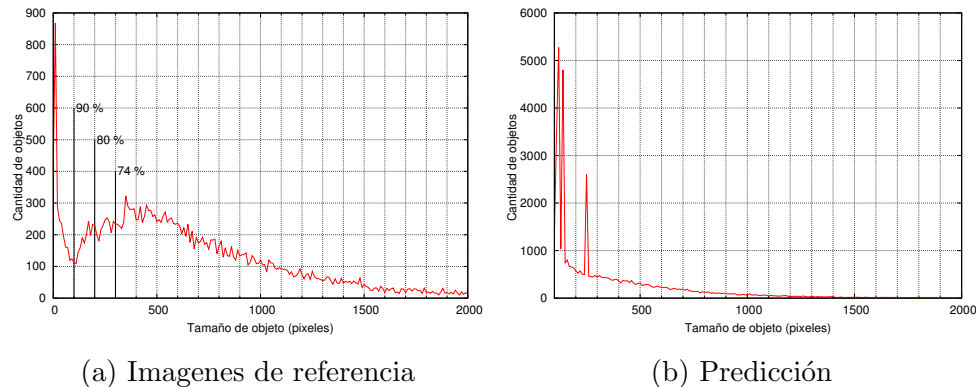


Figura 4.13: Variación de la cantidad de objetos mano de acuerdo con su tamaño

el 90 % presenta un tamaño mayor a 100 píxeles, mientras en el caso de la predicción se

contabiliza un total de  $6,90 \times 10^6$  objetos de los cuales un 99 % presenta un tamaño menor a 100 píxeles y un 95 % presenta un tamaño menor a 10 píxeles.

La figura 4.14 ilustra los resultados para diferentes valores de  $\tau_s$ . En ella los diferentes colores indican los diferentes objetos o clases en que fue separada la imagen. Como se

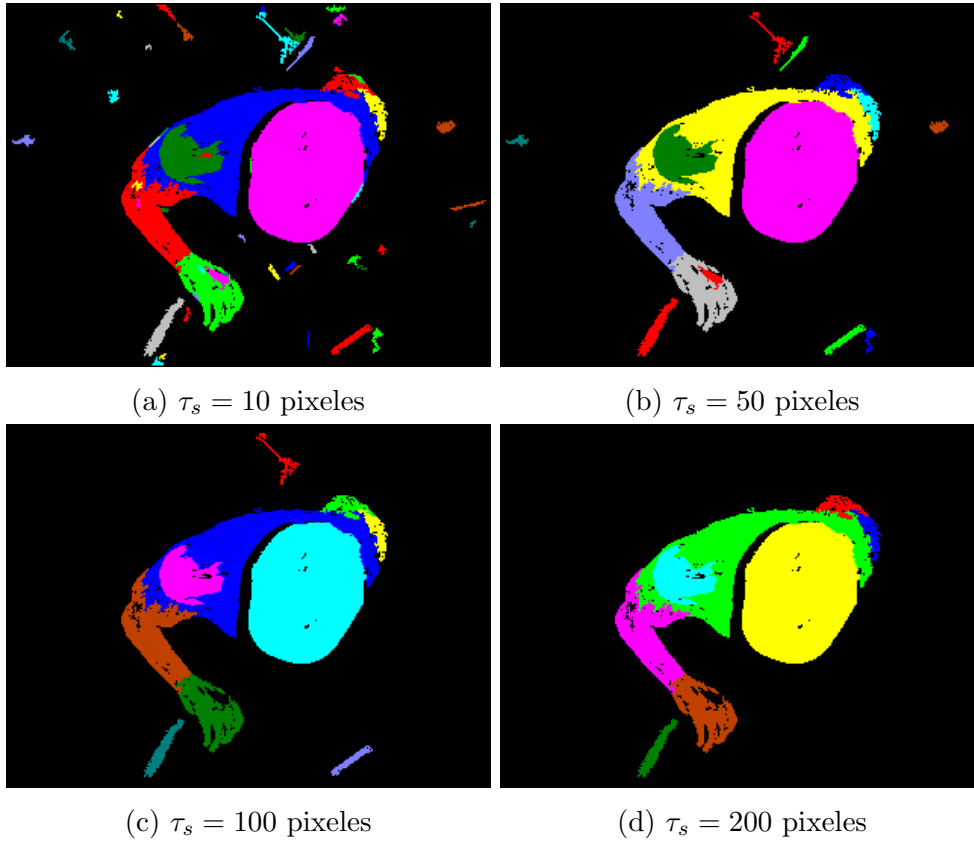
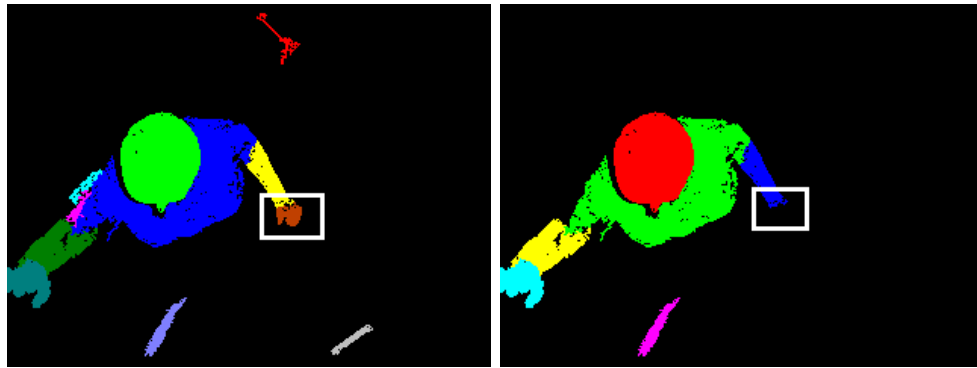


Figura 4.14: Efecto del tamaño de objeto en la separación de clases

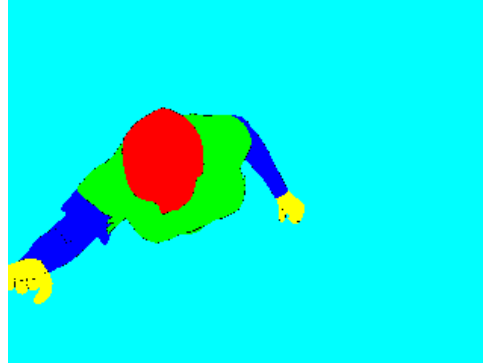
puede ver, con un mayor umbral se consigue eliminar una mayor cantidad de objetos del fondo de la imagen. Sin embargo, al aumentar este umbral se aumenta la probabilidad de eliminar manos reales de la predicción como se puede ver en la figura 4.15.

Utilizando un valor de  $\tau_s = 100$  píxeles se tiene una probabilidad de que el 90 % de los objetos mano está sobre ese umbral, mientras para un valor de  $\tau_s = 200$  píxeles esta probabilidad baja al 80 % lo que se traduce en la eliminación de más manos correctamente clasificadas. Esto se ilustra en la figura 4.15b, como se puede ver, al pasar de un umbral de 100 píxeles a uno de 200 píxeles se elimina una mano correctamente identificada.

La figura 4.16 ilustra los resultados tras utilizar un  $\tau_s = 100$  píxeles para la predicción. Como se observa, por debajo de un umbral de 300 píxeles la predicción presenta mayor número de objetos que la imagen de referencia, sin embargo, un  $\tau_s = 300$  píxeles implica una probabilidad la eliminación de objetos mano del 26 %.



(a) Separación con umbral de 100 px    (b) Separación con umbral de 200 px



(c) Clases teóricas

Figura 4.15: Error producido por el tamaño del umbral

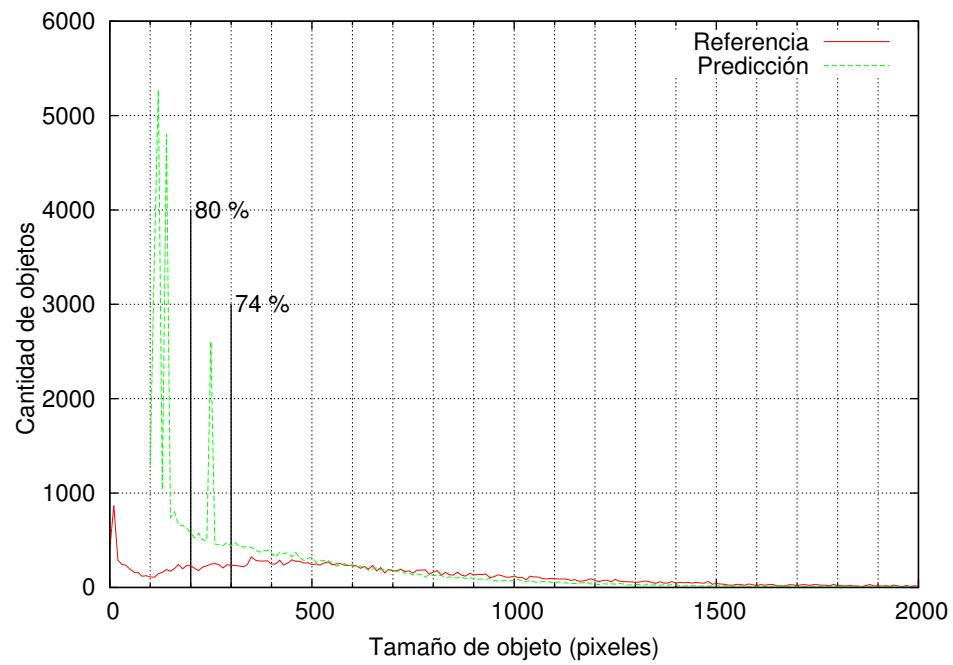


Figura 4.16: Predicción con un  $\tau_s = 100$  píxeles

### 4.3. Búsqueda de candidatos a mano

Una vez que se separan todas las clases mediante componentes conectados y se eliminan todos aquellos grupos de píxeles con una cantidad de píxeles menor a  $\tau_s$  se procede a la clasificación de los objetos restantes. Para ello se analiza cada uno de los objetos (grupos de píxeles de una misma clase) en la imagen dejando solo aquellos que fueron clasificados como mano con lo que se tiene una lista de candidatos. La figura 4.17 ilustra el resultado de este proceso.

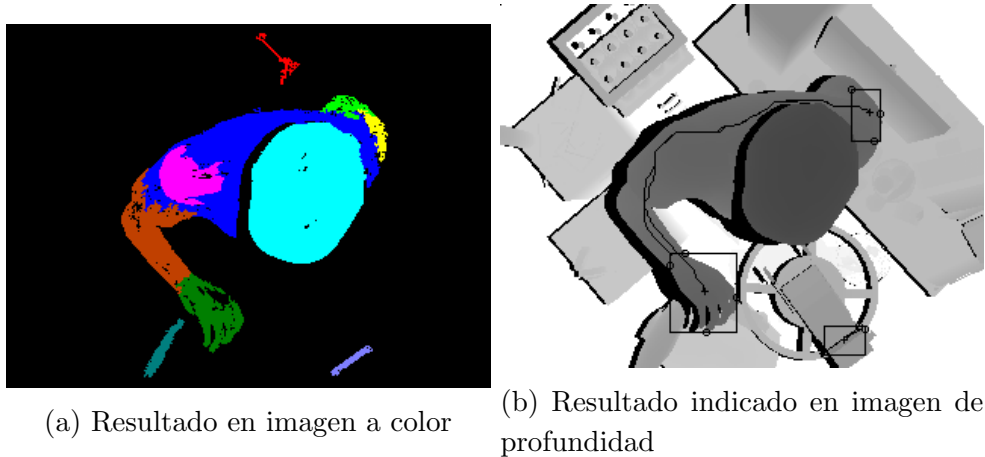


Figura 4.17: Candidatos a mano

En la figura 4.17a se puede ver el resultado tras el análisis de componentes conectados y en la figura 4.17b se puede ver como se marca en la imagen de profundidad aquellos que fueron predichos como mano. Como se observa, hay dos objetos que fueron incorrectamente marcados. Para suprimir estos casos, el siguiente paso en la búsqueda de candidatos consiste en eliminar de la lista todos aquellos objetos que no tengan al menos un objeto vecino diferente de fondo, el cual puede ser de cualquiera de las clases (mano, cabeza, cuerpo, brazo) ya que toda mano real debe tener asociado a si al menos un objeto de otra clase (brazo por ejemplo). La figura 4.18 muestra la imagen presentada en 4.17b tras la eliminación de aquellos objetos sin un vecino. Como se puede ver, quedan únicamente 2 candidatos a mano eliminando aquellos objetos aislados dejando una menor cantidad de candidatos para su final clasificación.

El que un candidato tenga al menos un objeto vecino no garantiza que dicho objeto sea realmente una mano pero el que no tenga un objeto vecino garantiza que no lo es por lo que al eliminar objetos aislados se reduce la cantidad de falsos positivos. Así por ejemplo, en una prueba realizada con un total de 500 imágenes, se encontraron un total de 1123 candidatos a mano de los cuales un 52,6 % corresponde a manos reales. Una vez eliminados los objetos aislados se tiene un total de 770 candidatos eliminándose un 31,4 % del total. De los objetos restantes, 71,7 % corresponde a manos reales mientras del 31,4 % eliminado, este porcentaje corresponde al 11 %, representando un 3,5 % de la lista original de 1123 candidatos. De esta forma, al eliminar los objetos aislados, se elimina el 6,6 % de manos reales lo cual se debe a que en esos casos, por la posición de las manos en la



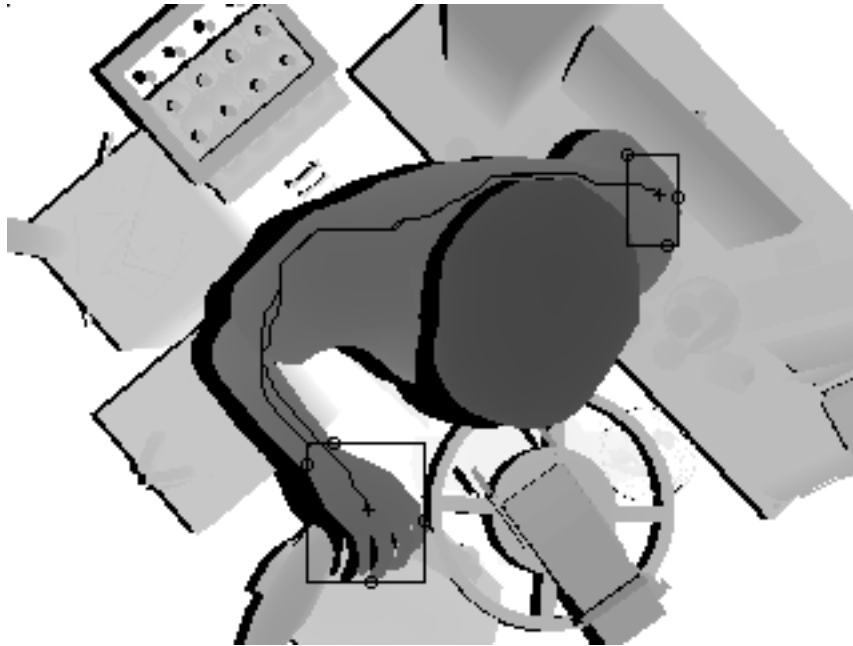


Figura 4.18: Candidatos a mano después de eliminar objetos aislados

imagen, aparecen como un objeto aislado. Esto se ilustra en la figura 4.19 donde se puede ver como la mano aparece en el borde de la imagen como un objeto aislado por lo que es eliminada de la lista de candidatos.

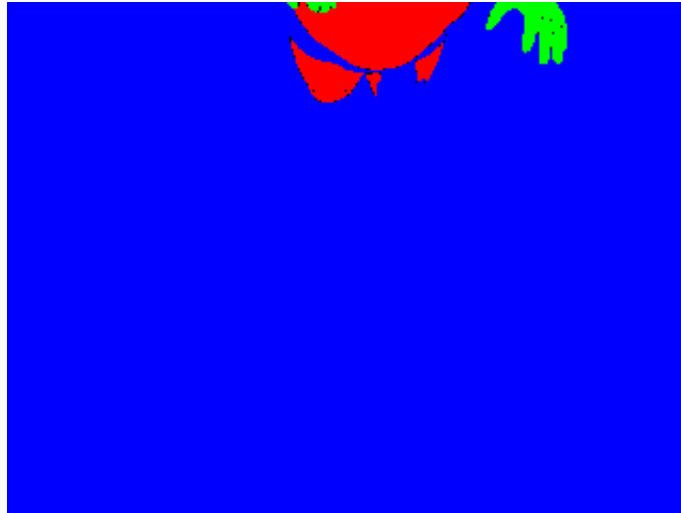


Figura 4.19: Ejemplo de una mano real eliminada

#### 4.4. Extracción de descriptores de ruta

Una vez que se tiene una lista de candidatos a mano se procede a la extracción de descriptores para cada uno de estos candidatos para su posterior clasificación. A continuación se presentan resultados de la obtención de la ruta y del histograma.

#### 4.4.1. Obtención de la ruta más larga

Utilizando distancias geodésicas se procede a la obtención de la ruta sobre la cual se determina la distribución de clases. Para lograr dicho cometido se procede a encontrar la ruta más larga a partir del centro del candidato a mano hasta una longitud máxima ( $L_{max}$ ) determinada. Para determinar el valor  $L_{max}$  óptimo se varía desde 10 cm hasta 150 cm y se mide la sensibilidad y la especificidad obtenidas y como resultado óptimo se escoge aquel que maximiza ambos parámetros. La figura 4.20 muestra los resultados obtenidos. Como se puede ver, conforme aumenta la longitud de la ruta, la sensibilidad de

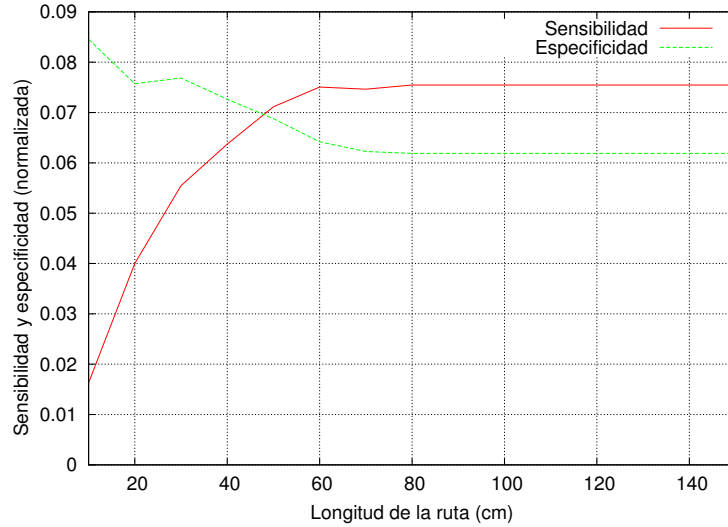


Figura 4.20: Variación de la sensibilidad y especificidad en la detección en función de la longitud de la ruta ( $L_{max}$ )

la detección aumenta, lo cual indica que la cantidad de manos correctamente detectadas aumenta también. De igual manera se ve como la especificidad decrece, indicando un descenso en la cantidad de manos correctamente descartadas por lo que el valor escogido es aquel que maximice ambos parámetros, es decir, que presente la mayor cantidad de manos correctamente detectadas junto con la mayor cantidad correctamente descartadas lo cual se da en la intersección de ambas gráficas con una longitud de la ruta de 50 cm. Una vez obtenida esta ruta se procede a la generación del histograma.

#### 4.4.2. Extracción del histograma

La extracción del histograma consiste en el conteo de la cantidad de píxeles de cada una de las clases que se encuentran a lo largo de la ruta encontrada. La figura 4.21 muestra el histograma obtenido en las dos rutas mostradas en la figura 4.18.

Como se puede ver, en el caso del histograma del objeto no mano, el porcentaje de píxeles pertenecientes a la clase mano a lo largo de la ruta es mucho menor al caso de la mano real, estas diferencias son las aprovechadas por el clasificador utilizado en la etapa final

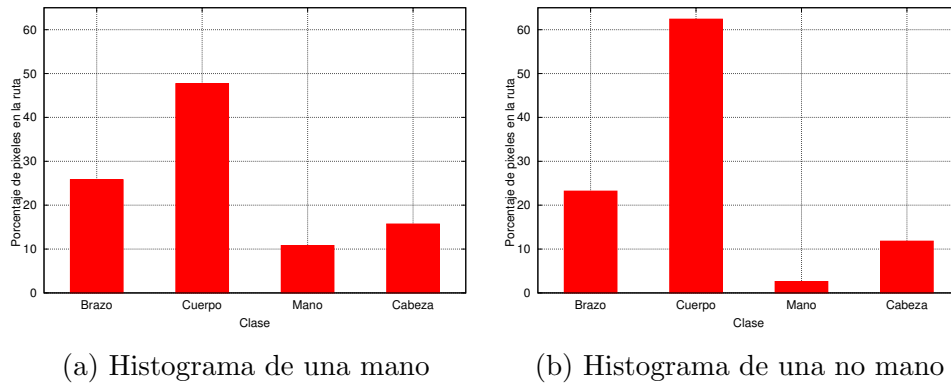


Figura 4.21: Histogramas para las rutas mostradas en la figura 4.18

para filtrar todos aquellos objetos erróneamente marcados como mano.

## 4.5. Clasificación mediante Máquinas de Vectores de Soporte (SVM)

Esta es la etapa final y en ella se eliminan, mediante el uso de máquinas de vectores de soporte, todos aquellos candidatos a mano que fueron marcados erróneamente como tal.

Para este proyecto se hace uso de SVM con kernel gaussiano:

$$k(\underline{\mathbf{x}}, \underline{\mathbf{x}}') = \exp(\gamma ||\underline{\mathbf{x}} - \underline{\mathbf{x}}'||^2) \quad (4.1)$$

$$\gamma = -\frac{1}{2\sigma^2} \quad (4.2)$$

Mediante el uso de kernel gaussianos se realiza un mapeo del espacio de características a un espacio multidimensional en el cual se realiza la separación de clases.

Para esta etapa se procede a determinar la combinación de parámetros que optimiza la separación de manos de las no manos. Para ello se cuenta con un conjunto de entrenamiento de 500 imágenes para las cuales se procede a variar valor de la constante  $C$  (ver (2.24)) y  $\gamma$  para posteriormente variar la cantidad de imágenes usadas en el entrenamiento.

Además se hace uso de otro conjunto de 500 imágenes seleccionadas aleatoriamente de la base de datos para su validación. Para verificar la efectividad de la clasificación se hace el cálculo de la especificidad y la sensibilidad (ver sección 2.5).

### 4.5.1. Entrenamiento de la SVM

Para determinar la combinación de parámetros que produce los mejores resultados en el entrenamiento, utilizando un conjunto de entrenamiento de 500 imágenes, se procede a

variar los parámetros  $C$  y  $\gamma$  del SVM con  $C$  variando de 10 a 100000 y  $\gamma$  variando de 0,1 a 10. Cabe destacar, que aunque el conjunto de entrenamiento inicial es de 500 imágenes, en cada imagen se encuentran al menos un elemento mano y uno no mano por lo que la cantidad de elementos con los que se entrena el SVM es mayor al número de imágenes. Una vez determinada la combinación de  $C$  y  $\gamma$  que produce los mejores resultados se procede a determinar el efecto de la variación de la cantidad de imágenes de entrenamiento en el desempeño del SVM.

Como vector característico se utiliza el histograma generado a partir de la ruta calculada más la longitud de la misma con lo que se tiene un vector de 5 elementos. Como entrada del algoritmo se tiene la lista de las posiciones teóricas de las manos en cada una de las imágenes así como la lista de los candidatos a mano determinado en las secciones anteriores. Para cada una de las imágenes se enseña al SVM que un candidato a mano es una mano real si en la lista de posiciones teóricas de las manos se encuentra una con una distancia no mayor a 5 píxeles al centro de la mano, de lo contrario se enseña al SVM que no es una mano. Además de ello se enseña que todo candidato cuya ruta más larga encontrada no supera los 10 cm no es una mano real ya que se considera que toda mano real debe tener otro objeto asociado (brazo o cuerpo por ejemplo) por lo que la ruta desde la mano debe ser mayor a 10 cm. La figura 4.22 muestra los resultados de la sensibilidad y especificidad.

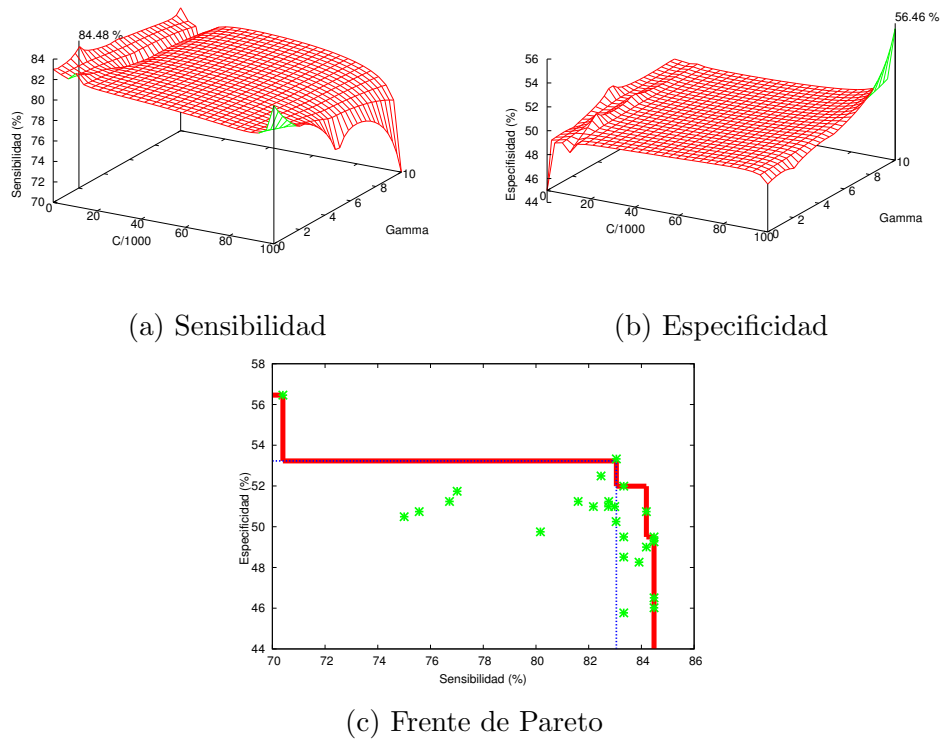


Figura 4.22: Variación de la sensibilidad y la especificidad al variar los valores de  $C$  y  $\gamma$  para el SVM

Como se puede ver en la figura 4.22a, la sensibilidad disminuye al aumentar tanto  $C$  como

$\gamma$  con la máxima sensibilidad obtenida de un 84,48 % para un valor de  $C$  de 100 y un valor de  $\gamma$  de 2. Este valor indica que de la totalidad de manos presentes en las imágenes de prueba se detectaron correctamente un 84,48 %. Por otro lado, tal y como se ve en la figura 4.22b la máxima especificidad obtenida fue de un 56,46 % para un valor de  $C$  de 100000 y un valor de  $\gamma$  de 10. Este valor indica que de la totalidad de objetos no mano presentes en las imágenes de prueba se descartaron correctamente un 56,46 %.

Estos resultados se pueden explicar debido a que un aumento en el valor de  $C$  ocasiona una disminución en el margen del SVM, además, un aumento en el parámetro  $\gamma$  implica una disminución en el valor de  $\sigma$  del kernel, lo cual ocasiona que el clasificador esté más cercano a un clasificador del vecino más cercano, el cual clasifica basado en la cercanía de los datos a los valores de entrenamiento, ocasionando una disminución en la generalización. Sumando estos factores ocasiona que la sensibilidad de la predicción disminuya tras el aumento de estos parámetros.

Como se puede ver, los valores máximos de sensibilidad y especificidad se presentan para valores de  $C$  y  $\gamma$  distintos. La figura 4.22c muestra el diagrama de Pareto para los resultados obtenidos mostrándose con una línea roja el frente de Pareto que es la zona en la que se encuentran los resultados no dominados y por lo tanto los candidatos a ser elegidos como los resultados óptimos. Con el objetivo de mantener un equilibrio entre la sensibilidad y la especificidad es que se elige  $C = 100$  con  $\gamma = 0,2$  (línea punteada) ya que si por ejemplo, se cuenta con una alta sensibilidad pero una baja especificidad se tiene un clasificador que identifica la gran mayoría de candidatos como mano. En este punto se obtiene una sensibilidad de 83,05 %, lo que se traduce en un total de 289 manos detectadas dentro de un total de 348 y una especificidad de 53,23 % que se traduce en un total de 289 manos descartadas en un total de 402 no manos.

Una vez que se determina la combinación de parámetros que produce la mejor clasificación, utilizando estos valores, se procede a determinar el efecto de la variación de la cantidad de imágenes utilizadas en el entrenamiento en el resultado de la clasificación. La figura 4.23 muestra la sensibilidad y especificidad de la clasificación tras variar la cantidad de imágenes de entrenamiento.

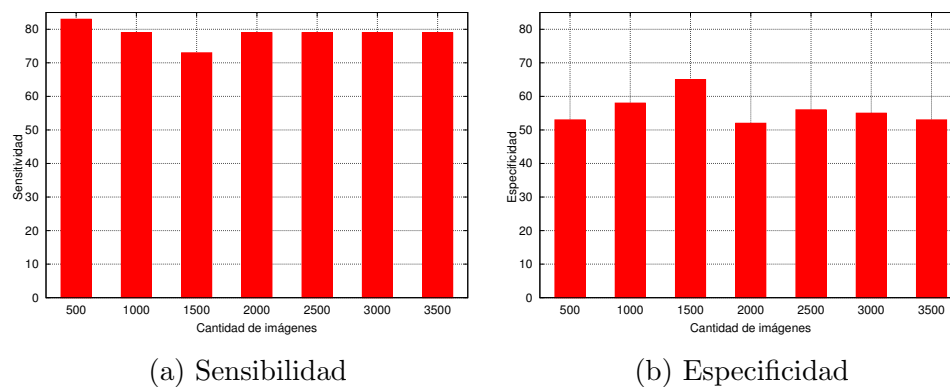


Figura 4.23: Variación de la sensibilidad y la especificidad al variar la cantidad de imágenes de entrenamiento del SVM para un  $C = 100$  y  $\gamma = 0,2$

Como se puede ver, aumentar la cantidad de imágenes utilizadas para el entrenamiento no produce mejores resultados, presentándose la mayor sensibilidad (83,05 %) para un total de 500 imágenes y la mayor especificidad (65,67 %) para un total de 1500 imágenes. Sin embargo, para este último caso también se presenta la menor sensibilidad (73,27 %) con lo que se selecciona como el mejor resultado el obtenido utilizando 500 imágenes de entrenamiento.

Estos resultados se pueden explicar debido a que al aumentar la cantidad de imágenes utilizadas en el entrenamiento se pierde la generalidad en la clasificación por lo que el SVM aprende a clasificar el grupo con el cual fue entrenado presentando peores resultados al validar con un grupo diferente. De aquí que la cantidad de imágenes de entrenamiento con la que se consigue el resultado que mejor describe a la población de de 500.

#### 4.5.2. Clasificación

Una vez determinados los parámetros que realizan de mejor manera la separación de los objetos mano de los no mano se procede a su evaluación.

Se detectan varios factores que afectan la sensibilidad y especificidad de la detección. El primero de ellos es las zonas en el cuerpo que a pesar de no ser manos son clasificadas como tal y por su tamaño no son filtradas al aplicar componentes conectados. Esto se ilustra en la figura 4.24.

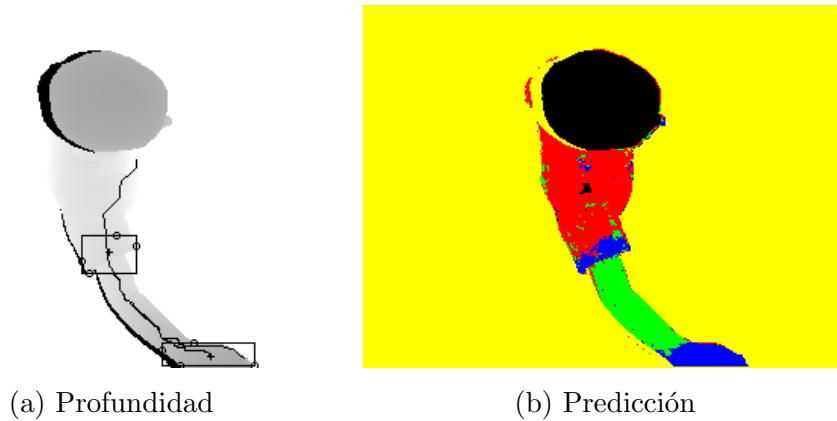


Figura 4.24: Ejemplo de un falso positivo debido a una zona del cuerpo erróneamente clasificada como mano

Como se puede ver, la zona de la manga fue clasificada como mano y al realizar el histograma sobre la ruta obtenida se obtiene un resultado consistente con el de una mano. Esto sumado a su tamaño hace que sea clasificado como mano lo cual aumenta la tasa de falsos positivos afectando la especificidad de la detección. Este problema se puede evitar utilizando un descriptor que considere la geometría del candidato a mano.

Otro factor que afecta la clasificación es los casos en que por la posición de la mano no es posible encontrar una ruta sobre la superficie del cuerpo cuya longitud sea mayor a 10

cm lo cual hace que sean consideradas como no manos por el clasificador aumentando la tasa de falsos negativos y afectando la sensibilidad de la clasificación. Esto se ilustra en la figura 4.25.

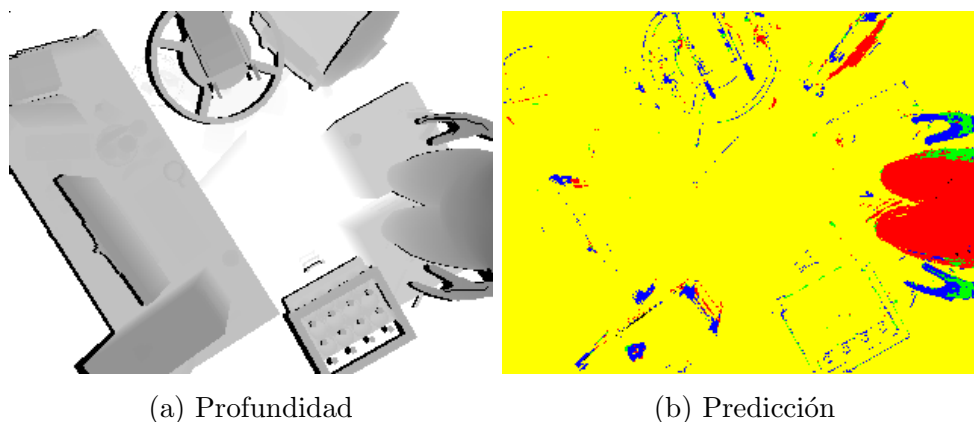


Figura 4.25: Ejemplo de un falso positivo debido a la posición de la mano

Como se puede ver, las dos manos presentes en la figura fueron descartadas debido a la longitud de la ruta encontrada sobre ambas. Debido a la posición en que estas se encuentran en la imagen no es posible encontrar una ruta que cumpla con los criterios definidos para ser considerada como una mano.

Finalmente, un factor que afecta en la clasificación es el valor de  $\tau_s$  utilizado para filtrar candidatos a mano. La figura 4.26 muestra la curva ROC (Receiver operating characteristic por sus siglas en inglés). En ella se presentan los resultados obtenidos tras variar el valor de  $\tau_s$  utilizado para filtrar los candidatos a mano.

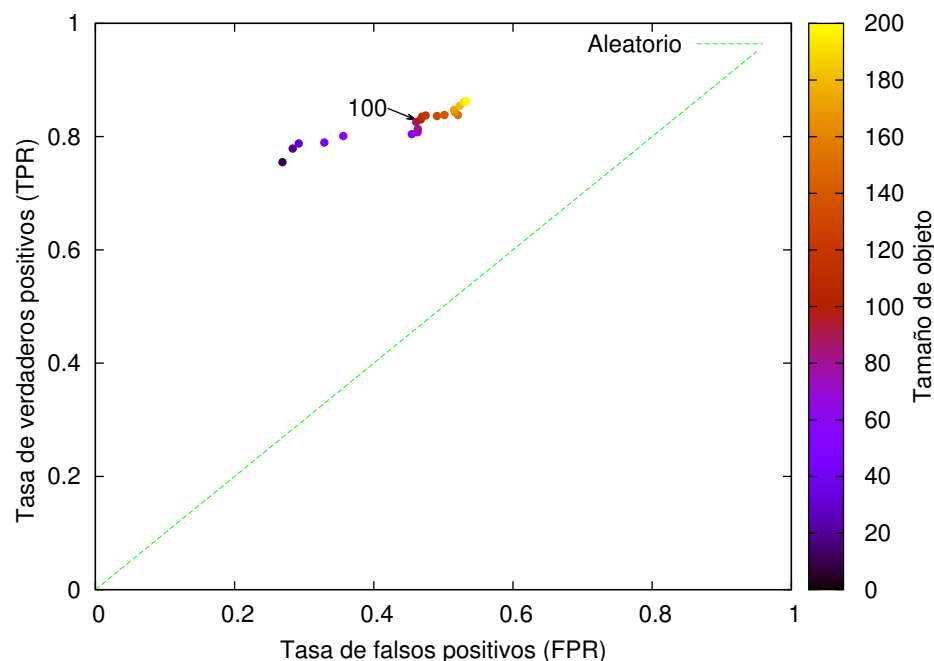


Figura 4.26: Espacio ROC para una variación del valor de  $\tau_s$

Como se puede ver, ante un aumento en el tamaño del objeto ( $\tau_s$ ), se presenta una tasa de falsos positivos mayor y una tasa de verdaderos positivos mayor. En ella se identifica el valor de  $\tau_s$  determinado de forma experimental en una etapa anterior. Ante tamaños de objetos menores se presenta una menor tasa de falsos positivos pero a su vez se presenta un menor tasa de verdaderos positivos y a tamaños de objeto mayores se presenta una tasa de verdaderos positivos mayor pero la tasa de falsos positivos también aumenta por lo que un valor intermedio es la mejor elección. Además, cabe destacar que ante un  $\tau_s$  menor se tiene una mayor cantidad de objetos por clasificar y la cantidad de falsos positivos aumenta. Esto se ilustra en la figura 4.27.

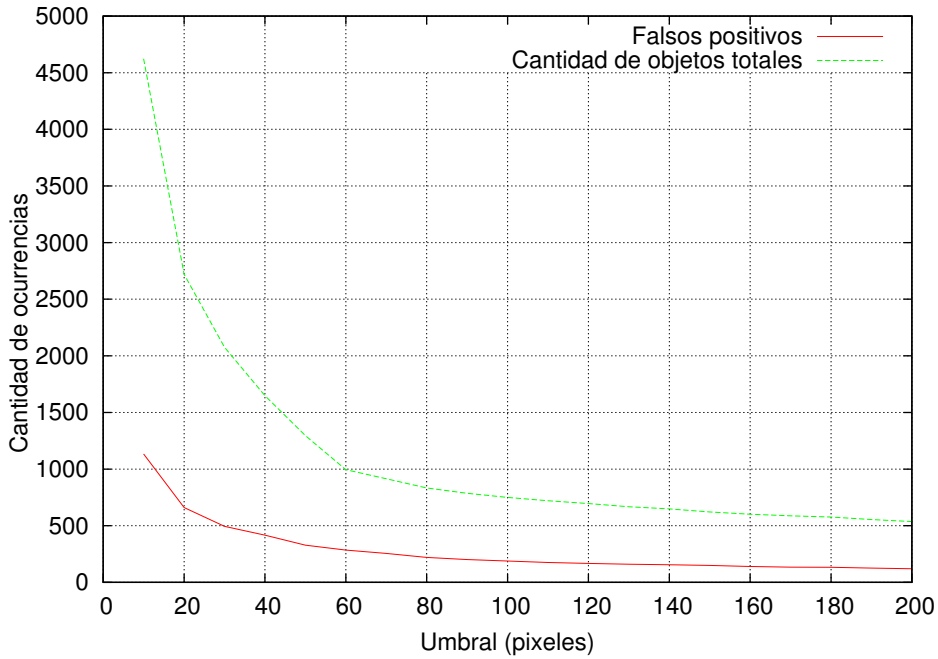


Figura 4.27: Efecto de  $\tau_s$  en la cantidad de falsos positivos de la clasificación

Como se puede ver, para un  $\tau_s$  menor a 100 píxeles la variación en la cantidad de falsos positivos tiene un comportamiento exponencial. Esto se debe a que, como se puede ver, ante un umbral menor la cantidad de objetos por clasificar es mayor, lo cual aun manteniendo la tasa de falsos positivos, hace que la cantidad de estos aumente.

## 4.6. Comportamiento del clasificador ante imágenes con ruido

En esta sección se evalúa el comportamiento del sistema ante imágenes con ruido. Para ello se hace uso de la misma base de datos pero se agrega ruido gaussiano en  $Z$  a cada una de las imágenes según se describe en [7]. Se utiliza un bosque con 10 árboles, 15 niveles por árbol y un  $BD : RS$  de 10 pxm : 5 pxm ya que estos son los valores que se determina producen los mejores resultados.



Además, se utiliza un valor de  $\tau_s = 100$  píxeles y se entrena el SVM con un valor de  $C = 100$  y un  $\gamma = 0,2$ . Se procede a entrenar el RDF utilizando imágenes con ruido según se describe en [7] para luego variar el valor de  $\sigma$ .

Para evaluar el desempeño del sistema se entrena el bosque con imágenes con ruido utilizando un valor de  $\sigma = 1$  (nivel de ruido del Kinect) para posteriormente evaluar utilizando valores de  $\sigma$  de 0.1 a 25. La figura 4.28 ilustra el resultado de la predicción utilizando imágenes con ruido para valores de  $\sigma = 1, 10$ , y 25.

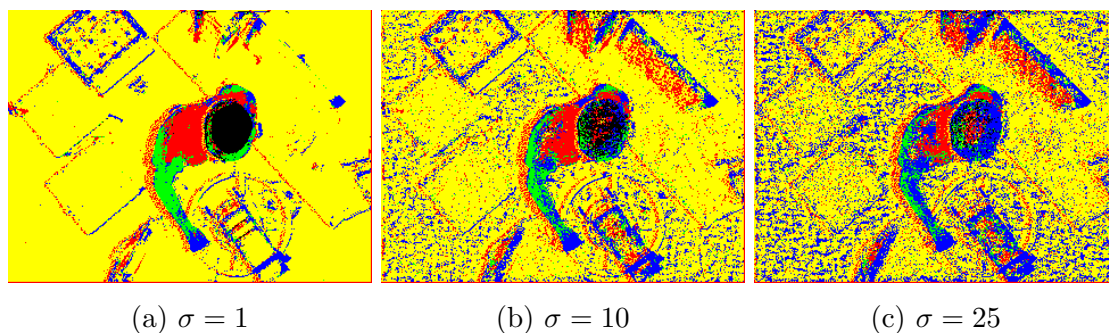


Figura 4.28: Ejemplos de predicciones con ruido ante la variación de  $\sigma$

Como se puede ver, al aumentar el valor de  $\sigma$ , es notoria la cantidad de falsos positivos en la predicción los cuales se representan como los puntos azules dispersos por toda la imagen. Como lo muestra la figura 4.29, gran cantidad de este ruido es filtrado al aplicar componentes conectados. Aunque la gran mayoría del ruido es filtrado se observa como

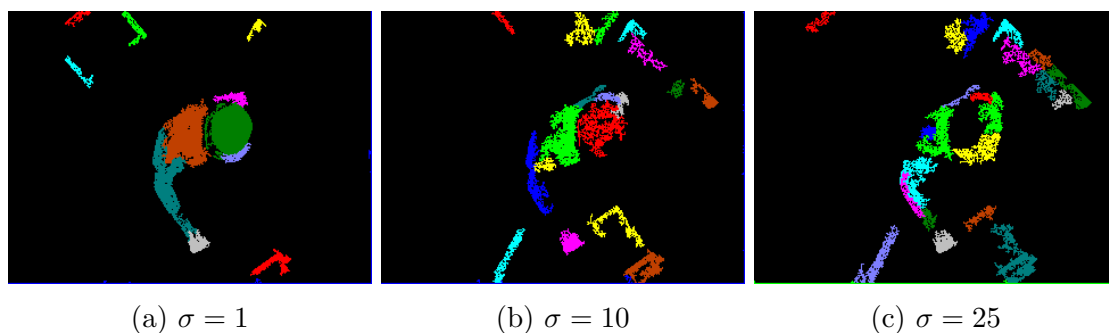


Figura 4.29: Ejemplos de predicciones con ruido después de aplicar componentes conectados ante la variación de  $\sigma$

los objetos detectados se ven degradados con el aumento en la cantidad de ruido. Esto se debe a que la cantidad de ruido introducida en la imagen de profundidad ocasiona que el predictor produzca grupos de píxeles menores para regiones del cuerpo mayores como lo son los brazos o el cuerpo.

La figura 4.30 muestra el resultado de la clasificación para los ejemplos mostrados en la figura 4.29. En ella se observa como en el caso de  $\sigma = 1$  se detectó correctamente la mano y se produjo un falso positivo. Además se puede ver como al aumentar el valor de  $\sigma$  aumenta la cantidad de falsos positivos en la clasificación, esto debido a que al ruido introducido

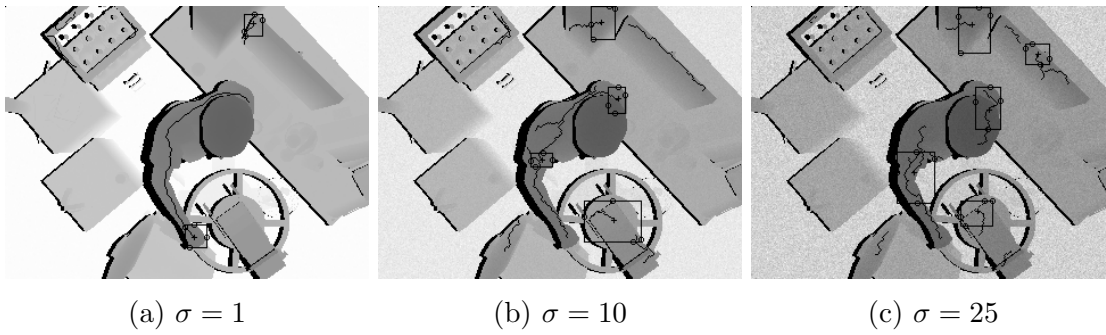


Figura 4.30: Ejemplos de predicciones con ruido después de la clasificación ante la variación de  $\sigma$

produce grupos de píxeles que no son filtrados al aplicar componente conectados sobre los cuales es posible encontrar una ruta cuyo histograma coincide con el encontrado en una mano real aumentando así la tasa de falsos positivos. Por otro lado, al mismo tiempo, el ruido introducido en las regiones correctamente detectadas como mano, brazo o cuerpo ocasiona que el histograma obtenido a partir de la ruta encontrada desde el centro del candidato a mano no coincida con la esperada por el SVM provocando que se clasifique como no mano y aumentando así la tasa de falsos negativos.

La figura 4.31 ilustra el comportamiento de la predicción ante la variación en el valor de  $\sigma$ . Todo punto que se localice cerca de la línea punteada central se considera como una

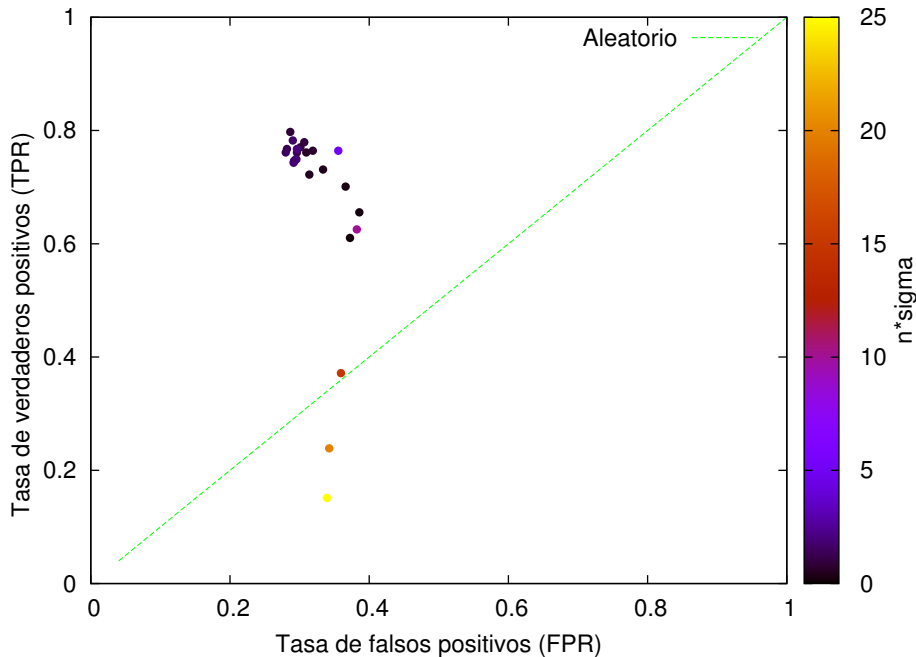


Figura 4.31: Curva ROC para una variación de la cantidad de ruido

predicción aleatoria. Cuanto más arriba de la línea se encuentre una predicción mejor se considera, de igual manera, cuanto más abajo se encuentre se considera como una peor predicción. Se puede apreciar claramente como al aumentar el valor de  $\sigma$  los puntos se

localizan más abajo en la gráfica lo que concuerda con los resultados mostrados, presentándose los mejores resultados para los valores más bajos de ruido. Por otro lado, se puede ver como la tasa de falsos positivos (FPR) tiende a ser constante lo cual se debe a que al aumentar el nivel de ruido aumenta tanto la cantidad de objetos por clasificar como la cantidad de falsos positivos (como se puede ver en la figura 4.32), con la cantidad de manos reales constante, lo que se traduce en el comportamiento mostrado en la figura 4.31.

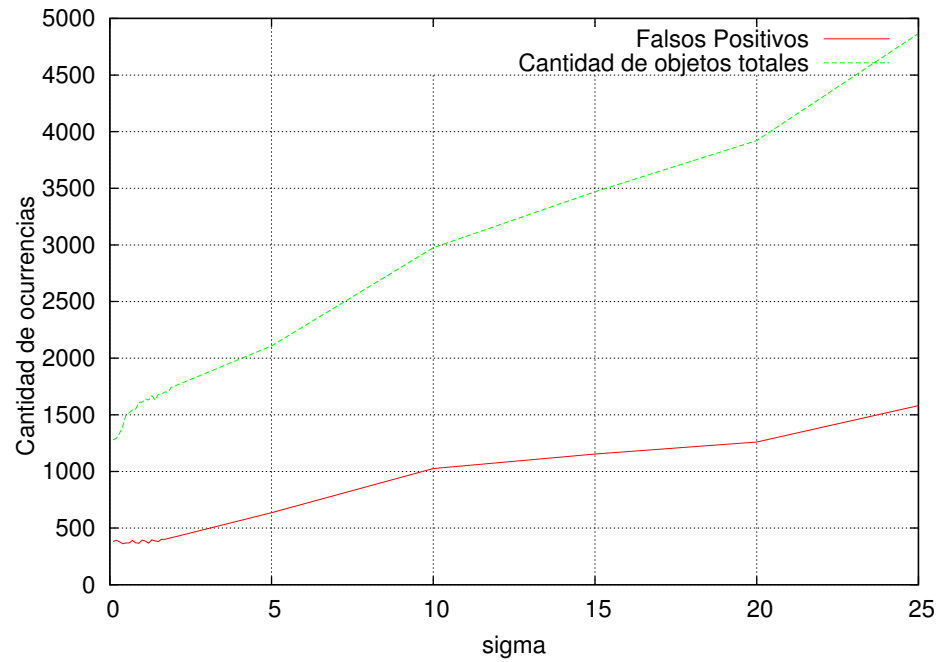


Figura 4.32: Efecto de la variación de  $\sigma$  en la cantidad de falsos positivos de la clasificación



# Capítulo 5

## Conclusiones

Se ha presentado un sistema para la detección de manos en imágenes de profundidad. Mediante la utilización de clasificadores basados en bosques de decisión aleatoria se puede realizar una segmentación de una escena en distintas clases definidas por el usuario. Sin embargo el ruido presente en las imágenes por clasificar tiene un alto impacto en la sensibilidad y precisión de la clasificación afectándolo negativamente. El desplazamiento máximo de caja y tamaño máximo de región utilizados son determinantes en la precisión de la clasificación. Además, se puede determinar cómo tanto un aumento en la cantidad de niveles como en la cantidad de árboles en el bosque mejora la sensibilidad y precisión de la clasificación.

Mediante la utilización del algoritmo de componentes conectados es posible realizar el filtrado de elementos incorrectamente clasificados al utilizar el predictor. Sin embargo, la definición de un umbral adecuado es determinante ya que por ejemplo, al pasar de un valor de 100 pxm a uno de 200 pxm se presenta una probabilidad del 20 % de eliminar elementos correctamente clasificados.

Además, mediante la utilización del algoritmo de Dijkstra se calculan distancias geodésicas en una imagen de profundidad y utilizando la ruta encontrada mediante la determinación de distancias geodésicas, se elabora un histograma que se utiliza para la clasificación de elementos mano o no mano.

Por otra parte, aunque la utilización del histograma como descriptor para la clasificación presenta una forma práctica de clasificar manos, se determinó que se ve limitado por casos en los que no es posible encontrar una ruta que describa correctamente una mano.

También se determinó cómo las máquinas de vectores de soporte son una opción válida para la clasificación de manos y que estas, sin necesidad de una gran cantidad de imágenes de entrenamiento, describen adecuadamente la población.

Finalmente, se recomienda para trabajos futuros, el complementar los histogramas utilizados para la clasificación final de las manos con otro tipo de descriptor que permita describir la geometría de la mano. Esto porque aunque los histogramas presentan una

forma práctica de describir una posible mano, se ve limitada en ocasiones en las que, por la posición de la mano, no es posible elaborar un histograma con las características esperadas para una ruta sobre una mano real.

# Bibliografía

- [1] De Barcelona. Real-time Hand Pose Recognition using Depth Sensors combined with Spherical Shape Model Descriptor Oscar Lopes Advisor : Sérgio Escalera. URL <http://www.maia.ub.es/~sergio/linked/masteroscarlopes2012.pdf>.
- [2] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. *Methods in molecular biology (Clifton, N.J.)*, 609:223–239, 2010.
- [3] André Brandao, Leandro A. F Fernandes, and Esteban Clua. M5AIE : A Method for Body Part Detection , Tracking and Pose Classification using RGB-D Images. *International Conference on Computer Vision Theory and Applications used*, pages 1–8, 2014. URL [http://www2.ic.uff.br/~laffernandes/projects/tracking/2014\\_VISAPP/brandao\\_et\\_al\\_VISAPP\\_2014.pdf](http://www2.ic.uff.br/~laffernandes/projects/tracking/2014_VISAPP/brandao_et_al_VISAPP_2014.pdf).
- [4] Leo Breiman. Random Forests. *Machine Learning*, pages 1–33, 2001.
- [5] Christopher J C Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. URL <http://www.springerlink.com/index/Q87856173126771Q.pdf><http://research.microsoft.com/pubs/67119/svmtutorial.pdf>.
- [6] Chia-Ping Chen, Yu-Ting Chen, Ping-Han Lee, Yu-Pao Tsai, and Shawmin Lei. Real-time hand tracking on depth images. *2011 Visual Communications and Image Processing (VCIP)*, (1):1–4, November 2011. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6115983>[http://www.eyetap.org/~raymondlo84/tmp/hand\\_tracking\\_kinect.pdf](http://www.eyetap.org/~raymondlo84/tmp/hand_tracking_kinect.pdf).
- [7] Benjamin Choo, Michael Landau, Michael DeVore, and Peter Beling. Statistical Analysis-Based Error Models for the Microsoft KinectTM Depth Sensor. *Sensors*, 14:17430–17450, 2014. URL <http://www.mdpi.com/1424-8220/14/9/17430/>.
- [8] A Criminisi, J Shotton, and E Konukoglu. Decision Forests for Classification , Regression , Density Estimation , Manifold Learning and Semi-Supervised Learning. *Learning*, 7:81–227, 2011. URL <http://www.nowpublishers.com/product.aspx?product=CGV&doi=0600000035>.
- [9] Edsger Wybe Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1(1 959):269–271, 1959.

- [10] Accelerating Random Forests. Accelerating Random Forests on CPUs and GPUs for Object-Class Image Segmentation. 2013. URL [http://www.ais.uni-bonn.de/theses/Benedikt\\_Waldvogel\\_Master\\_Thesis\\_07\\_2013.pdf](http://www.ais.uni-bonn.de/theses/Benedikt_Waldvogel_Master_Thesis_07_2013.pdf).
- [11] Matheus Giovanni and Soares Beleboni. A brief overview of Microsoft Kinect and its applications. 2014.
- [12] A. Hernandez-Vela, N. Zlateva, A. Marinov, M. Reyes, P. Radeva, D. Dimov, and S. Escalera. Graph cuts optimization for multi-limb human segmentation in depth maps. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 726–732, June 2012. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247742>.
- [13] Vitaliy Konovalov, Albert Clapés, and Sergio Escalera. Automatic hand detection in rgb-depth data sequences. *Frontiers in Artificial Intelligence and Applications*, 256:91–100, 2013. URL <http://www.maia.ub.es/~sergio/linked/vitaliyccia2013.pdf>.
- [14] Jeff Kramer, Matt Parker, Hc Daniel, Florian Echtler, and Nicolas Burrus. *Hacking the Kinect*. 2012. URL <http://link.springer.com/content/pdf/10.1007/978-1-4302-3868-3.pdf>.
- [15] A. Kurakin, Z. Zhang, and Z. Liu. A real time system for dynamic hand gesture recognition with a depth sensor. *20th European Signal Processing Conference (EUSIPCO 2012)*, (Eusipco):1975–1979, 2012. URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=6333871](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6333871).
- [16] Stephen McKeague, Jindong Liu, and Guang-Zhong Yang. Hand and body association in crowded environments for human-robot interaction. *2013 IEEE International Conference on Robotics and Automation*, pages 2161–2168, May 2013. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6630867><http://hamlyn.doc.ic.ac.uk/eo/gesturedataset/paper.pdf>.
- [17] Klaus Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001. URL <http://media.cs.tsinghua.edu.cn/~taopin/ML2005/intro2Kernel-basedLearn-TNN-2001.pdf>.
- [18] I. Oikonomidis, N. Kyriazis, and a. a. Argyros. Tracking the articulated motion of two strongly interacting hands. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1862–1869, June 2012. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247885>[http://www.ics.forth.gr/\\_publications/2012\\_06\\_cvpr\\_twohands.pdf](http://www.ics.forth.gr/_publications/2012_06_cvpr_twohands.pdf).
- [19] Christian Plagemann, Varun Ganapathi, Daphne Koller, and Sebastian Thrun. Real-time identification and localization of body parts from depth images. *2010 IEEE International Conference on Robotics and Automation*, pages 3108–3113,



- May 2010. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5509559>.
- [20] Akmal Rakhmadi, Nur Zuraifah Syazrah Othman, Abdullah Bade, Mohd Shafry Mohd Rahim, and Ismail Mat Amin. No Title. *Journal of Computer Science*, 6(10):1096–1104, 2010.
- [21] Siddharth S. Rautaray and Anupam Agrawal. Real Time Multiple Hand Gesture Recognition System for Human Computer Interaction. *International Journal of Intelligent Systems and Applications*, 4(5):56–64, May 2012. URL <http://www.mecspress.org/ijisa/ijisa-v4-n5/v4n5-8.html><http://www.mecspress.org/ijisa/ijisa-v4-n5/IJISA-V4-N5-8.pdf>.
- [22] Gerhard X Ritter and Joseph N Wilson. *Computer Vision Algorithms in Image Algebra*. 2001.
- [23] G.X. Ritter and J.N. Wilson. *Handbook of computer vision algorithms in image algebra*. 2001.
- [24] Jong Hyun Ryu, Sujin Kim, and Hong Wan. Pareto front approximation with adaptive weighted sum method in multiobjective simulation optimization. *Proceedings - Winter Simulation Conference*, (x):623–633, 2009.
- [25] Hannes Schulz, Benedikt Waldvogel, Rasha Sheikh, and Sven Behnke. CURFIL : Random Forests for Image Labeling on GPU. 2015.
- [26] Atid Shamaie and Alistair Sutherland. A Dynamic Model for Real-Time Tracking of Hands in Bimanual Movements 2 Tracking Hands in Bimanual Movements. pages 172–179, 2004.
- [27] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Iza-di. Accurate, Robust, and Flexible Real-time Hand Tracking. *Microsoft Research*, 2015.
- [28] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. *Cvpr 2011*, pages 1297–1304, June 2011. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5995316>.
- [29] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, and Alex Kipman. Efficient human pose estimation from single depth images. *Decision Forests for Computer Vision and Medical Image Analysis*, 35(12):175–192, 2013. URL [http://link.springer.com/chapter/10.1007/978-1-4471-4929-3\\_13\\$\\delimiter"026E30F\\$npapers3://publication/uuid/B0D3C6EC-3C89-4BF8-B1E4-CC0A8DD03567](http://link.springer.com/chapter/10.1007/978-1-4471-4929-3_13$\\delimiter).

- [30] Justin Solomon and Adrian Butscher. Discrete Differential Geometry. In *Discrete Differential Geometry*, chapter Lecture 10. 2013.
- [31] Lin Song, Ruimin Hu, Yulian Xiao, and Liyu Gong. Real-Time 3D Hand Tracking from Depth Images. *Proceedings of the 2nd International Conference On Systems Engineering and Modeling*, pages 1119–1122, 2013. URL <http://www.atlantis-press.com/php/paper-details.php?id=5732>.
- [32] Jonathan Tompson, Murphy Stein, Yann Lecun, Ken Perlin, and Offline Database. Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. In *SIGGRAPH*, 2014.
- [33] Michael Van den Bergh and Luc Van Gool. Combining RGB and ToF cameras for real-time 3D hand gesture interaction. *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 66–72, January 2011. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5711485>.
- [34] David a Van Veldhuizen and Gary B Lamont. Evolutionary Computation and Convergence to a Pareto Front. *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228, 1998. URL <http://www.lania.mx/~ccoello/EM00/vanvel2.ps.gz>.
- [35] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:I–511–I–518, 2001. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=990517>.
- [36] Paul Viola and Mj Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [37] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28:1, 2009. URL <http://portal.acm.org/citation.cfm?doid=1576246.1531369><http://people.csail.mit.edu/rywang/handtracking/s09-hand-tracking.pdf><http://people.csail.mit.edu/rywang/handtracking/>.
- [38] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE Multimedia*, 19:4–10, 2012.

# Índice alfabético

Clasificación SVM, 58  
Componentes conectados, 8  
  
Descriptor ruta, 53  
Dijkstra, 11  
Distancias geodésicas, 11  
  
Entrenamiento SVM, 55  
Espacio ROC, 21  
Especificidad, 21  
Estadística, 18  
Estado del arte, 23  
  
Histograma, 54  
  
Máquinas de vectores de soporte, 15  
Matriz de Confusión, 18  
  
Objetivos, 3  
  
Precisión, 21  
  
RDF, 5  
Resultados, 39  
Ruido, 60  
Ruta más larga, 54  
  
Sensibilidad, 20  
Solución, 29  
SVM, 55  
  
Teoría, 5  
  
Validación regiones mano, 48  
Variación cantidad de árboles, 46  
Variación cantidad de niveles, 46  
Variación desplazamiento máximo de caja,  
41  
Variación tamaño de región, 41