Implementación y análisis de un detector de manos basado en visión artificial

Lucía Cheung
Carlos A. Medina C.
Facultad de Ingeniería Eléctrica
Universidad Tecnológica de Panamá
lucia.cheung@utp.ac.pa
carlos.medina@utp.ac.pa

Resumen— *Este artículo documenta el desarrollo y análisis de un detector de manos en imágenes y vídeo basado en visión artificial. Se combinaron ideas y aportes de diversos autores, determinando que una implementación factible del detector de manos es utilizando clasificadores en cascada con "boosting", junto con descriptores basados en gradientes, que capturan efectivamente las características de las manos. Para ello, se investigó e implementó cada una de las etapas involucradas en este detector, incluyendo la preparación de la base de datos de imágenes, el entrenamiento del sistema, utilizando algoritmos de aprendizaje automático y la implementación del algoritmo de detección. Se evalúan y comparan diferentes implementaciones con ligeras variaciones entre sí, para determinar aquella con mejor desempeño en términos de precisión y tiempo de procesamiento. Para mejorar este último parámetro, también se investigan aspectos relativos a la optimización del código utilizado en el detector.*

Palabras claves— *árboles de decisión con boosting, clasificadores en cascada, descriptores basados en gradientes, detección de manos, histogramas de gradientes orientados.*

Abstract— *This article documents the development and analysis of a computer vision-based hand detector for images and video. Ideas and contributions from several authors were combined, and it was determined that a feasible implementation of the hand detector can be achieved by using a cascade of boosted classifiers, along with gradient based descriptors, which effectively capture the characteristics of the hands. For this purpose, we investigated and carried out each of the stages involved in the development of this detector, including the preparation of the database of images, the training of the system using machine learning algorithms, and the implementation of the detection algorithm. Different implementations with slight variations were assessed and compared in terms of their accuracy and processing time. Aspects relative to the optimization of the code used in the detector were also investigated.*

Keywords— *boosted decision trees, cascaded classifiers, gradient based descriptors, hand detection, histograms of oriented gradients.*

Tipo de artículo: *original*
Fecha de recepción: *30 de julio de 2012*
Fecha de aceptación: *5 de febrero de 2013*

1. Introducción

El avance de la tecnología, particularmente el desarrollo de nuevos dispositivos y aplicaciones, ha impulsado el interés en explorar nuevas modalidades de interacción, que permitan al usuario aprovechar al máximo las capacidades y funcionalidades de estas nuevas tecnologías. En este sentido, existe un creciente interés, en especial, por las "interfaces de usuario naturales", que son capaces de interpretar expresiones naturales del ser humano como los gestos y las palabras, dependiendo poco así de la manipulación de dispositivos hardware por parte del usuario.

Este trabajo es importante porque las interfaces no obstructivas basadas en visión son una alternativa atractiva, pero aún se encuentran con muchas limitaciones para ser consideradas como una alternativa seria a los dispositivos de interacción tradicionales. El principal reto es lograr el balance entre la eficiencia computacional (para garantizar la respuesta en tiempo real) y la robustez del sistema (para operar bajo condiciones no controladas).

En particular, el reconocimiento de gestos de la mano no es una tarea trivial, ya que la mano es altamente deformable, posee muchos grados de libertad, y puede adoptar muchas poses diferentes [1]. Estas características de las manos se suman a las dificultades que de por sí tiene cualquier tarea de visión artificial: lidiar con condiciones de iluminación variables, ambientes visualmente complejos y dinámicos, sombras irregulares, cambios de perspectiva, oclusión, ruido y demás.

El problema de detección de manos y reconocimiento de gestos para la interacción hombre-computador es abordado con métodos muy diversos, ya que no existe un único método para el reconocimiento automático que sea adecuado para todas las aplicaciones. Cada uno depende del dominio de la aplicación, el entorno físico en el cual será utilizado, e incluso el tipo de usuario final que lo empleará [2].

En los primeros sistemas de reconocimiento de gestos, se emplearon diferentes tipos de marcadores, como cintas adhesivas de colores o guantes especiales, para facilitar la detección y el seguimiento de la mano. Sin embargo, en la mayoría de los trabajos posteriores se ha buscado eliminar todo tipo de marcas auxiliares, ya que no se alinea con el objetivo de lograr interfaces no obstructivas.

La mayoría de los métodos o enfoques se pueden clasificar como basados en movimiento, en profundidad, en color, forma, apariencia y "multipistas" (que consideran varias características de manera combinada para tomar la decisión final) [2]. En [2] y [3], se hace una extensa comparación de los diferentes enfoques utilizados.

En general, se puede decir que los métodos basados únicamente en color, movimiento o forma (silueta o geometría) son sencillos y rápidos, pero hacen suposiciones fuertes acerca del entorno (por ejemplo, [4]). Los métodos basados en color, generalmente restringen los colores que pueden presentarse en la imagen; los métodos basados en forma, asumen que el entorno es simple para hacer la segmentación adecuadamente; y, los métodos basados en movimiento para la detección funcionan principalmente bajo entornos estáticos. En algunos casos, los clasificadores consisten en reglas codificadas manualmente basándose en el conocimiento específico del dominio.

Por otro lado, los métodos basados en apariencia toman en cuenta la intensidad de los pixeles y características que surgen de la relación entre los valores de los pixeles, tales como el contraste entre regiones, los bordes, los gradientes, la textura, entre otras. Se extraen múltiples características *(features)* de una base de datos de imágenes, y se entrenan clasificadores utilizando aprendizaje automático. Entre los métodos basados en apariencia más populares para el reconocimiento de objetos están los *Haar-like features* [5] (utilizados en [6] y [7]) y los histogramas de orientación de gradientes (utilizados en [8]).

El objetivo de este trabajo es diseñar e implementar un detector de manos, basado en visión artificial. Para ello, se utiliza una cámara web sencilla para capturar los cuadros de vídeo, que luego se procesan por medio de *software* para identificar la posición y el tamaño de las manos.

Para este proyecto, era prioridad que el detector fuese robusto y estable ante una gran variedad de condiciones. Por lo tanto, se hizo la menor cantidad posible de suposiciones acerca del usuario y el entorno en el cual se utilizará el sistema. Entre las características deseadas del sistema están:

- La capacidad de operar bajo condiciones de iluminación y fondo variables, cambiantes e impredecibles, incluyendo la posibilidad que aparezcan otras partes del cuerpo dentro de la imagen.
- La flexibilidad que el usuario pueda utilizar cualquier vestimenta. Esto es porque en algunos sistemas se exige al usuario utilizar camisas con mangas largas, para facilitar la segmentación de las manos.
- La capacidad de detectar la posición y el tamaño de la mano; es decir, que la mano no necesariamente estará en el centro de la imagen y pueden haber otros elementos dentro de ésta.
- Funcionar sin requerir de guantes o marcas especiales en las manos.

Además, como se espera aplicar este sistema como un método de interacción, este debe tener buen tiempo de respuesta, ya sea en tiempo real (15~30 fps) o tiempo casi real (NRT –*near real-time*) (5~15 fps).

A continuación se documenta el desarrollo general del detector, especificando el método propuesto, la evaluación de diversos aspectos y características del mismo; se presentan las conclusiones del diseño y análisis realizados, y se indican algunos trabajos futuros con base en este proyecto.

Figura 1. Ejemplo de vocabulario de gestos [7].

2. Plataformas y herramientas

El desarrollo de este proyecto no requirió de equipo especializado, y la programación se basó en librerías gratuitas de código abierto (open source). El desarrollo completo del proyecto, las implementaciones y las pruebas se efectuaron en una computadora portátil con procesador Pentium Dual-Core CPU T4500 @ 2.30 GHz, 2 GB de memoria RAM y sistema operativo Windows 7 de 64 bits; para la captura de las imágenes y vídeo, una cámara web normal de un solo lente *Logitech QuickCam Pro for Notebooks*, que captura imágenes en tres canales de color; es decir, que no se emplearon cámaras estereoscópicas ni con sensores de profundidad.

En cuanto al *software* para procesamiento de imágenes y visión artificial, se utilizó la librería OpenCV en sus versiones 2.2.0 y 2.3.1, y la librería IntelTBB para paralelizar la ejecución de ciertos bucles dentro del algoritmo de detección.

3. Métodos utilizados

El planteamiento inicial del proyecto contemplaba el reconocimiento de diferentes gestos estáticos de un vocabulario predefinido, como los de la Figura 1, donde cada gesto representaría un comando dentro de una aplicación. Se procedió a investigar los diferentes enfoques utilizados para el reconocimiento multigestos, pero se encontró que muchos funcionaban únicamente bajo condiciones muy restringidas. Entonces se decidió utilizar la fusión de dos métodos basados en apariencia *(Haar Cascade Classifier* y *Histograms of Oriented Gradients)*, que son robustos y pueden utilizarse bajo condiciones variadas, pero que están formulados para el caso binario (dos clases). Esto implicó el uso de un solo gesto, ya que una de las clases representaría dicho gesto, y la otra clase, correspondería "al resto".

Los métodos investigados se pueden extender al caso multiclase, pero esto conlleva sus propias complejidades técnicas. Por esta razón, se decidió desarrollar el sistema para reconocimiento de un solo gesto. Se eligió el gesto de "mano abierta", que es el más fácil de realizar y más cómodo para el usuario. El hecho de utilizar un solo gesto puede ser una ventaja desde el punto de vista del usuario, ya que no tiene que memorizarse varios gestos diferentes y el significado de cada uno.

Para compensar la pérdida de expresividad por el hecho de emplear un solo gesto (figura 2a), se dedicó mayor esfuerzo a lograr que el sistema pudiera reconocer la mano abierta en movimiento y en diferentes perspectivas (figura 2b).

En cuanto al marco *(framework)* de detección, a pesar de que la cascada con *boosting* y *Haar-like features* ha tenido mucho éxito en la detección de rostros y de algunos otros tipos de objetos, no ha resultado ser tan adecuada para la detección de manos. La principal razón se debe al hecho que, a diferencia de los rostros, las manos tienen poca estructura interna [9] (ej. en los rostros: ojos, nariz, boca). En [10] también se menciona el hecho de que los *Haar-like features* no funcionan tan bien cuando la característica más distintiva del objeto es su contorno, como sucede con las manos. Por este motivo, al igual que en [9], se optó por utilizar la técnica de Histogramas de Gradientes Orientados (HOG–*Histograms of Oriented Gradients*). Estos features capturan mejor las propiedades de las manos, como el contorno, los dedos, así como algunos gradientes sutiles en la palma.

Los métodos utilizados para el desarrollo de este trabajo incluyen:

- Características basadas en gradientes, específicamente *HOG features*, ya que codifican mejor las propiedades de las manos.
- Bloques descriptores de diferentes tamaños y proporciones como en [11], ya que tienen mayor potencial para capturar patrones útiles del objeto.
- Histogramas integrales [5], para calcular los *features* de manera eficiente.
- Un clasificador dispuesto en cascada, que permite enfocar la atención del detector en regiones "prometedoras" de la imagen, y descartar patrones negativos fáciles de manera eficiente.

4. Implementación

Una vez establecidos los métodos a utilizar, el tipo de gesto a reconocer y las limitaciones, el siguiente paso fue preparar una base de datos con muestras positivas (manos) y muestras negativas (otros objetos o patrones).

Se implementó el algoritmo para calcular los histogramas de orientación de gradientes, y un algoritmo de aprendizaje para construir el clasificador en cascada. Además, se compararon

Figura 2. Muestras del conjunto de entrenamiento: (a) con poco movimiento y rotación, (b) con mayor movimiento y rotación.

distintas implementaciones en donde se variaron aspectos relacionados con el cálculo de los gradientes y los histogramas.

A continuación se presentan los detalles de la implementación del detector.

4.1. Preparación de la base de datos

Para entrenar el sistema se preparó una base de datos de muestras positivas que consiste en 7000 imágenes de manos (izquierda y derecha), tomadas con diferentes fondos y con iluminación variada. La mayoría de las imágenes fueron capturadas con la misma cámara que se utilizó para hacer el resto de las pruebas. Estas muestras eran cuadradas (1:1).

Para las muestras negativas se utilizaron 5410 imágenes que no contienen manos. Una parte se extrajo de Internet, y otra parte fue capturada con la cámara web. A diferencia de las muestras positivas, estas imágenes eran de cualquier tamaño y proporción. De estas imágenes se extraían subregiones cuadradas seleccionadas al azar para utilizar como muestras negativas. Se trató que las imágenes fueran lo más variadas posible. Algunos ejemplos de muestras positivas y negativas se muestran en la Figura 3.

4.2. Partición de la base de datos

Para evaluar la capacidad de generalización del clasificador, es necesario separar previamente las muestras en dos grupos: el conjunto de entrenamiento y el conjunto de prueba. El conjunto de entrenamiento es utilizado por el algoritmo de aprendizaje automático para aprender los parámetros del modelo. La capacidad de generalización de este modelo es posteriormente evaluado utilizando el conjunto de prueba, que consiste en muestras completamente nuevas, que el modelo no ha visto anteriormente.

Para el entrenamiento de cada etapa se utilizaron 6000 muestras positivas y 6000 muestras negativas. Cada vez que se entrena una etapa nueva y se añade al clasificador, una fracción de las muestras es rechazada por el clasificador, por lo que se debe reponer con muestras nuevas para mantener las 6000 muestras de cada tipo. Para las muestras positivas, es necesario reponer aproximadamente el 0.5 %, y para las muestras negativas, cerca del 50 %.

4.3. Caracterización de la mano utilizando histogramas de orientación de gradientes

Como se indicó, en este proyecto se decidió utilizar los HOG *features*, ya que en trabajos como los de [8] se ha demostrado que los histogramas de gradientes son más adecuados para codificar las características de las manos que los contrastes entre regiones *(Haar-like features)*.

El gradiente de la imagen puede calcularse de diferentes maneras. Una de ellas es convirtiendo primero, la imagen a una imagen en escala de grises, para luego calcular el gradiente. Otra manera es calculando directamente el gradiente de la imagen a

Figura 3. Muestras (a) positivas (b) negativas.

color tomando en cuenta los tres canales de color y utilizando el enfoque de gradiente de color basado en vector o gradiente vectorial propuesto por [12]. Este último método ha demostrado una mejor relación señal a ruido que el gradiente escalar, que está definido como la norma L2 de los gradientes escalares de cada canal. Debido a que el gradiente vectorial se plantea como una opción más resistente al ruido, pero de mayor demanda computacional, surgió un problema interesante a examinar, que es el *trade-off* entre la eficiencia computacional y la calidad del *feature*. En la sección de resultados se presenta la comparación del desempeño de las dos implementaciones de gradiente. Aunque existen otros métodos, solo se probaron los dos mencionados, para luego elegir el que presentara mejor desempeño.

4.4. Construcción de los histogramas de orientación y resolución de los mismos

Primero, la imagen se divide en cuadrículas o celdas. Los histogramas de orientación se calculan de manera local en cada una de estas celdas. Luego, se construyen los bloques descriptores concatenando y normalizando los histogramas de cuatro celdas adyacentes. Para construir el histograma de una celda, cada pixel dentro de la celda contribuye a la magnitud de su gradiente al intervalo *(bin)* correspondiente en el histograma.

La propuesta original de Dalal y Triggs [13] utiliza histogramas de nueve intervalos para la detección de peatones. Las manos tienen características diferentes, por lo que no necesariamente se dará un desempeño óptimo utilizando la misma cantidad de intervalos. Por esto, se realizaron pruebas con histogramas de nueve intervalos y, en un esfuerzo por mejorar la eficiencia computacional, también con histogramas de seis intervalos. A simple vista parece que esta reducción drástica en la resolución del histograma puede impactar negativamente de manera significativa el desempeño. Sin embargo, el resultado no es necesariamente obvio, ya que el tener demasiados intervalos puede también ser contraproducente. Los resultados de las evaluaciones se presentan en la sección de resultados, y se compara el desempeño en términos de la tasa de falsa alarma y de detección, y también se compara el tiempo de procesamiento.

4.5. Bloques descriptores

Un bloque descriptor se forma al concatenar el histograma de gradientes de cada una de las cuatro celdas que componen el bloque como se ilustra a continuación en la figura 4. Cada bloque descriptor, con los cuatro histogramas, se normaliza independientemente de los otros bloques. Según los resultados en [13], la mejor forma de normalizar es utilizando la norma L2-Hys, pero por eficiencia computacional, en este proyecto se utilizó la norma L2 (distancia euclideana).

Como se mencionó anteriormente, en este proyecto se adoptó la idea de [11] de utilizar bloques de diferentes tamaños y proporciones. En la figura 5 se muestran los diferentes tamaños de bloques que se utilizaron. Las dimensiones están dadas en fracciones con respecto al tamaño de la

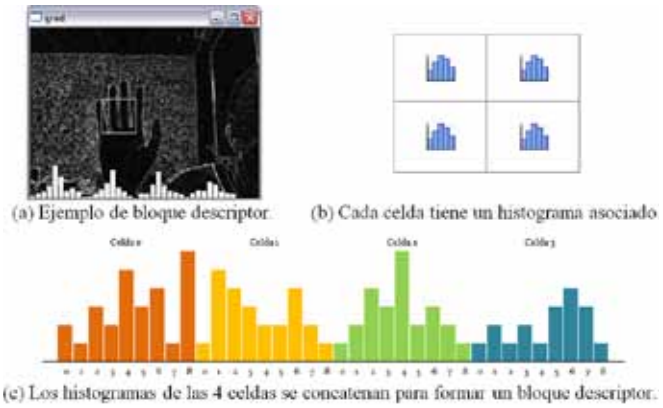

Figura 4. Bloque descriptor.

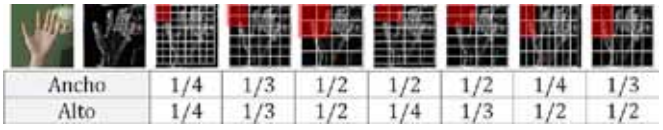

Figura 5. Bloques descriptores.

ventana (la cual mantiene siempre la relación de aspecto 1:1). El traslape es de medio bloque en todos los casos. En total resultaron 155 bloques.

Dependiendo de la cantidad de intervalos que se utilice para el histograma, pueden haber 5580 *features* (155 bloques × 4 celdas/bloque × 9 *features*/celda) o 3720 *features* (155 bloques × 4 celdas/bloque × 6 *features*/celda), para 9 y 6 intervalos respectivamente.

4.6. Histograma integral de gradientes

El algoritmo no es eficiente si para calcular un bloque descriptor se necesita realizar la suma de los gradientes de todos los pixeles en cada celda. Por ello, se utiliza una imagen integral para cada intervalo del histograma, para obtener de manera eficiente la suma de las magnitudes en cada una de las celdas. La imagen integral permite calcular las sumas en regiones rectangulares con pocos accesos a memoria, lo que permite reducir de manera significativa el tiempo de procesamiento, ya que no sólo se emplea la imagen integral dentro de la ventana de detección para calcular rápidamente los bloques, sino que se emplea a nivel de la imagen completa donde se están buscando las manos. De esta manera, una vez calculado el histograma integral de la imagen, éste se puede utilizar para calcular eficientemente todos los bloques de todas las ventanas de detección dentro de la imagen.

4.7. Entrenamiento del clasificador

A continuación se describe la estructura del clasificador utilizado y los componentes que lo conforman.

Estructura del clasificador

La estructura del clasificador consiste en una cascada, cuyas etapas están dispuestas en orden de complejidad creciente. Por lo general, en las imágenes predominan las ventanas negativas; la estructura en cascada permite rechazar, de manera eficiente, estas ventanas que no son de interés y enfocar los recursos en las ventanas positivas.

Cada etapa de la cascada consiste en un clasificador con *boosting*, que a su vez está compuesto de varios clasificadores más simples, los clasificadores débiles. Para este proyecto se utilizaron árboles de decisión de un solo nivel como clasificador débil, debido a su simplicidad.

Como se explicó anteriormente, el método utilizado en este proyecto es similar al de [11], que modifica el algoritmo desarrollado en [13]. Una modificación adicional que se introdujo en este proyecto, es la utilización de árboles de decisión con *boosting*, en lugar de clasificadores tipo SVM. En este proyecto no se realizaron pruebas para comparar el desempeño de ambos métodos. Sin embargo, para muchas tareas, los clasificadores con *boosting* basados en árboles de decisión han demostrado ser suficientemente buenos a pesar de su simplicidad [14].

Entrenamiento de la cascada de clasificadores

Las etapas de la cascada se entrenan de manera secuencial. Los ejemplos que se utilizan para entrenar cada etapa son los ejemplos que logran pasar las etapas anteriores. Esto significa que al finalizar el entrenamiento de cada etapa, las muestras deben ser evaluadas nuevamente por la cascada incluyendo la última etapa entrenada. Como cada etapa está entrenada para rechazar el 50 % de las muestras negativas, hay que reponer el conjunto de muestras negativas con nuevas muestras extraídas de la base de datos. Para el entrenamiento de las etapas se utilizó el código ya implementado por *OpenCV*.

4.8. Detección

La preparación de la base de datos y el entrenamiento del clasificador constituyen la parte *"offline"* de la implementación del proyecto. La otra parte consiste en el propio proceso de detección de manos dentro de un vídeo.

El programa de detección que se implementó analiza cada cuadro del vídeo de manera independiente. Cada cuadro de 320×240 es escaneado y analizado en su totalidad para

determinar la presencia de manos. Entre los parámetros que se deben elegir están: el tamaño de las ventanas de detección y el paso entre ventanas.

El algoritmo es capaz de detectar manos de cualquier tamaño; sin embargo, para usos prácticos del detector se fijó un rango que va de 60×60 hasta 173×173 pixeles (ver figura 6), que corresponde a una distancia de aproximadamente 1 metro hasta 30 centímetros entre la mano y la cámara (puede variar según el tamaño de la mano y las condiciones de fondo e iluminación).

La orientación de la mano debe ser vertical, pero se tolera una ligera rotación tanto en el plano como fuera del plano de ±30° aproximadamente.

El detector puede operar bajo una gran variedad de condiciones de entorno e iluminación, y no impone restricciones en cuanto a la vestimenta del usuario. Además, no es necesario ningún tipo de calibración inicial. El ajuste de brillo y contraste, enfoque, exposición y balance de blancos se deja al software de la cámara, ya que son funciones que realizan bastante bien las cámaras modernas.

Post-procesamiento de ventanas

Al momento de escanear la imagen se producen múltiples detecciones positivas sobre el mismo objeto, por lo que es necesario agrupar dichas detecciones. Dos ventanas se consideran del mismo grupo si la distancia y la diferencia de tamaño se encuentra dentro de un rango establecido.

Los grupos que están compuestos por menos de tres ventanas al final son eliminados. De esta manera se consigue reducir las falsas alarmas sin afectar de manera apreciable la tasa de detección.

4.9. Optimización

Debido a las restricciones de tiempo, es muy importante la forma como se implementa el algoritmo de detección. A pesar de que diferentes implementaciones del código permiten obtener el mismo resultado, puede haber una diferencia muy significativa en el tiempo de procesamiento. Dos factores que permitieron reducir el tiempo de procesamiento fueron, el uso de una tabla de búsqueda para guardar los factores de normalización que son compartidos por todos los intervalos de los histogramas de un mismo bloque, y la paralelización del código.

En cada cuadro de vídeo el clasificador debe evaluar miles de ventanas para determinar si corresponde o no a una mano. Esta es la parte que toma más tiempo y de la cual depende principalmente el *frame rate* del video. Para

Figura 6. Rango de tamaño detectable.

reducir este tiempo, se utiliza la librería Intel TBB para paralelizar el proceso de escaneo, tal que en lugar de procesar, ventana por ventana, de manera secuencial, se procesan varias ventanas simultáneamente, utilizando múltiples hilos (*threads*).

También se utiliza esta librería para paralelizar el cálculo del gradiente de la imagen, ya que el gradiente de cada pixel puede ser calculado de manera independiente.

5. Evaluación del desempeño del sistema

Se evaluó y comparó el desempeño del sistema bajo diferentes implementaciones del cálculo del gradiente y diferentes resoluciones del histograma. Para el cálculo del gradiente, se probaron dos opciones: el gradiente de la imagen en escala de grises, y el gradiente vectorial, para la imagen a color. Por otro lado, se probó el histograma con seis intervalos y con nueve intervalos para cada tipo de gradiente.

Para medir el desempeño se utilizó el conjunto de prueba, que consiste en muestras positivas y negativas en el mismo formato que el conjunto de entrenamiento. Las muestras son evaluadas por el clasificador en cascada, y de ahí se determina la tasa de acierto (número de verdaderos positivos / número de muestras positivas) y la tasa de falsa alarma del sistema (número de falsos positivos / número de muestras negativas).

Para comparar el desempeño de diferentes implementaciones, se utilizó la curva ROC *(Receiver Operating Characteristics)*, que grafica la tasa de acierto contra la tasa de falsa alarma (figura 7). Mientras más se acerque la curva a la esquina superior izquierda, mejor, ya que representa una mejor tasa de detección para una determinada tasa de falsa alarma. Para construir la curva ROC se evaluó el clasificador añadiendo una etapa a la vez, hasta tener la cascada completa.

Otro aspecto importante que se debe considerar es el tiempo de procesamiento, por lo

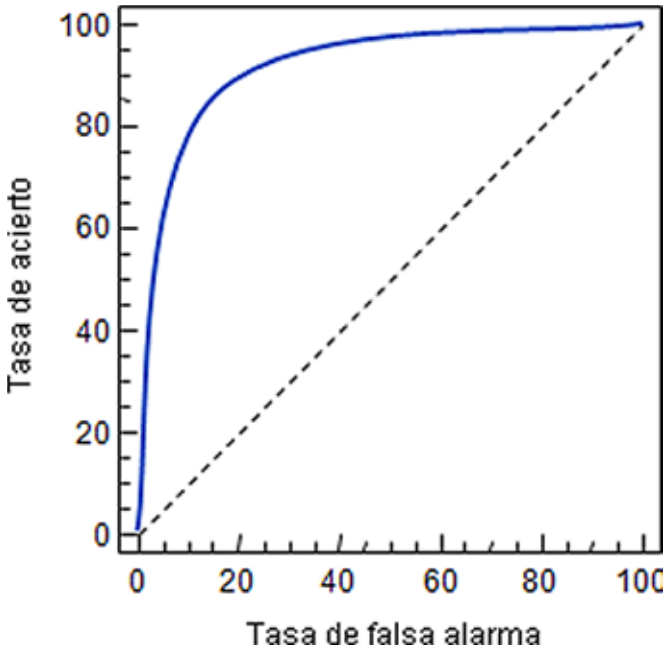

Figura 7. Curva ROC.

que las distintas implementaciones también fueron puestas a prueba con una secuencia de video. De las cuatro implementaciones diferentes, se eligió la de mejor desempeño para comparar el tiempo de procesamiento con y sin paralelización.

6. Resultados y discusión

En esta sección se analizan las características y el desempeño de los clasificadores entrenados. Se comparan las diferentes implementaciones en términos del tiempo de entrenamiento, el número de clasificadores débiles, la tasa de acierto y de falsa alarma, y el tiempo de procesamiento.

6.1. Resultados del entrenamiento

El tiempo de entrenamiento de un clasificador en cascada de 20 etapas, utilizando 6000 muestras positivas y 6000 negativas por etapa, resultó entre 9 y 14 horas. El proceso que tomaba más tiempo no era el entrenamiento de los clasificadores débiles en sí, ya que el espacio de búsqueda era relativamente reducido (5580 *features* para histogramas de

nueve intervalos y 3720 para histogramas de seis intervalos). Antes de entrenar una etapa, el algoritmo dedica una gran parte del tiempo al proceso de reponer las muestras negativas rechazadas por una etapa anterior.

El tiempo de entrenamiento requerido y el número de clasificadores débiles total se presentan en la tabla 1. Como puede observarse, las implementaciones con histogramas de seis intervalos requirieron menor tiempo de entrenamiento.

Muchos de los *features* son reutilizados ya sea dentro de una misma etapa o en etapas diferentes, por lo que la cantidad de clasificadores débiles no representa la cantidad de features diferentes utilizados.

A continuación se examinan los clasificadores débiles seleccionados por el algoritmo de aprendizaje. Debido a la gran cantidad de datos y la cantidad de dimensiones que se maneja, es complicado, pero a la vez importante, poder visualizar y verificar los cálculos y los resultados de los algoritmos. A pesar de que no es posible verificar cada uno de los clasificadores débiles de cada una de las etapas, la visualización permite comprobar, a grandes rasgos, que el resultado del entrenamiento tiene sentido.

En la figura 8 se muestran los clasificadores de la primera etapa de la cascada (para la implementación con gradiente vectorial de color e histogramas de seis intervalos). El cuadro blanco indica la posición del bloque descriptor, mientras que las líneas rojas y celestes muestran la posición y orientación del gradiente evaluado. El color rojo indica que la muestra recibe un "voto positivo" por parte del clasificador débil si el gradiente en esa posición es *mayor* que el umbral. Por otro lado, el color celeste indica lo contrario, que la muestra recibe un "voto positivo" si el gradiente es *menor* que el umbral. Los mismos clasificadores funcionan para detectar tanto la mano derecha como la mano izquierda.

Como muestran estas figuras, los bloques descriptores más discriminantes son de diferentes tamaños y proporciones, lo que verifica lo señalado en [11], que utilizando bloques de diferentes tamaños y proporciones —en lugar de bloques cuadrados de un solo tamaño— es posible capturar patrones más útiles. Es interesante notar, a parte de la posición del gradiente, el área que abarca el bloque, ya que los gradientes son normalizados localmente, por bloque. Muchos de los bloques son relativamente grandes que capturan la diferencia en la cantidad y la magnitud de los gradientes dentro de la palma de la mano con respecto al contorno y los dedos.

Tabla 1. Tiempo de entrenamiento.

Gradiente	Intervalos del Histograma	Tiempo (horas)	Número de clasificadores débiles
Gris	6	9:22	3565
Gris	9	14:23	3169
Color	6	11:11	2980
Color	9	12:16	2736

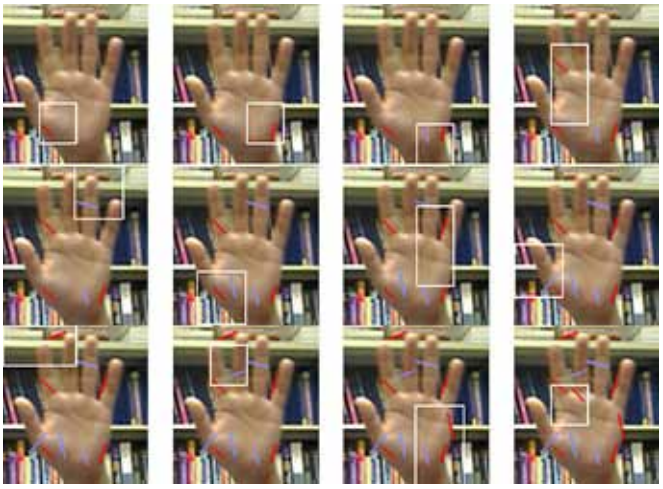

Figura 8. Clasificadores débiles de la primera etapa (gradiente de color, 6 intervalos).

En forma similar se analizaron los clasificadores débiles de las demás etapas. Como era de esperarse, los gradientes rojos se concentran principalmente en el contorno de la mano (que significa que deben haber gradientes fuertes en la orientación señalada), mientras que los celestes se concentran más en la palma (es decir, que no deben haber gradientes fuertes en la orientación señalada).

6.2. Evaluación del desempeño de los clasificadores

En esta sección se compara el desempeño en términos de la tasa de acierto y de falsa alarma de las distintas implementaciones, en las cuales se variaron el tipo de gradiente y el número de intervalos para los histogramas.

Número de intervalos del histograma

Como se explicó anteriormente, el objetivo de reducir el número de intervalos en los histogramas, es reducir el procesamiento requerido y hacer más eficiente la evaluación de los clasificadores. Sin embargo, es importante verificar primero que esto no deteriore el desempeño del sistema. En las gráficas de la Figuras 9 y 10, se puede observar que un histograma de seis intervalos proporciona un desempeño ligeramente superior que uno de nueve intervalos, particularmente para tasas de falsa alarma bajas.

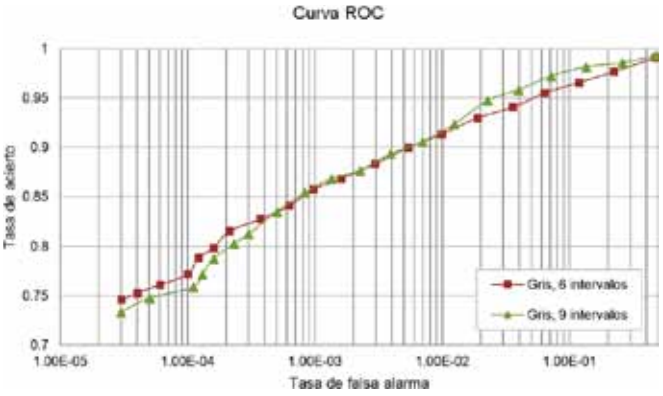

Figura 10. Comparación 6 intervalos vs 9 intervalos, gradiente escalar gris.

Para tasas de falsa alarma elevadas, el histograma de nueve intervalos tiene mejor desempeño, pero al ir añadiendo etapas, llegado un punto, el histograma de seis intervalos empieza a ser mejor. Esto es cierto tanto para el gradiente vectorial de color, como para el gradiente escalar gris.

Tipo de gradiente

En cuanto al tipo de gradiente, el gradiente vectorial de color presentó mejor desempeño cuando se utilizaban seis intervalos, mientras que utilizando nueve intervalos, ninguno presentó una ventaja significativa. Estos resultados se muestran a continuación.

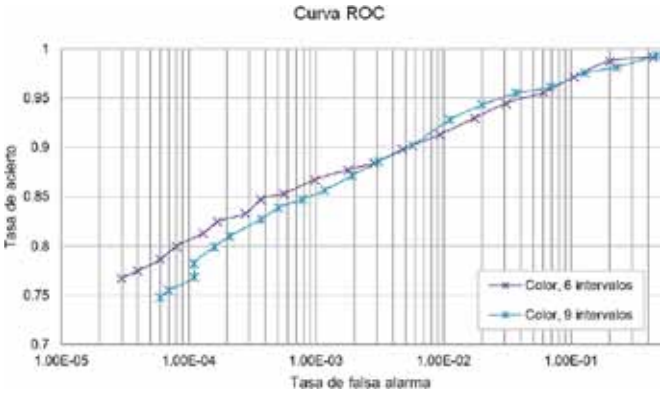

Figura 9. Comparación 6 intervalos vs 9 intervalos, gradiente vectorial de color.

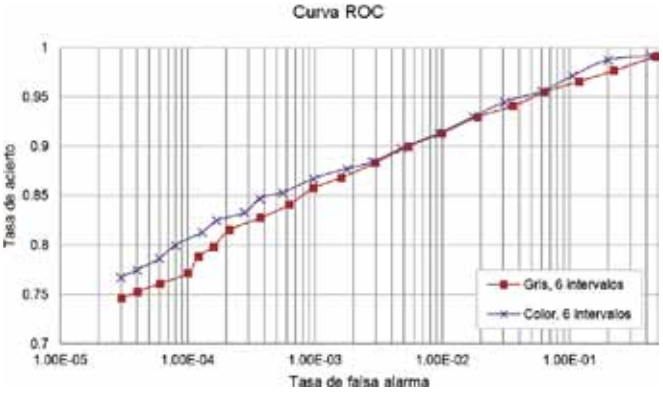

Figura 11. Comparación gradiente vectorial de color vs. gradiente escalar gris, 6 intervalos.

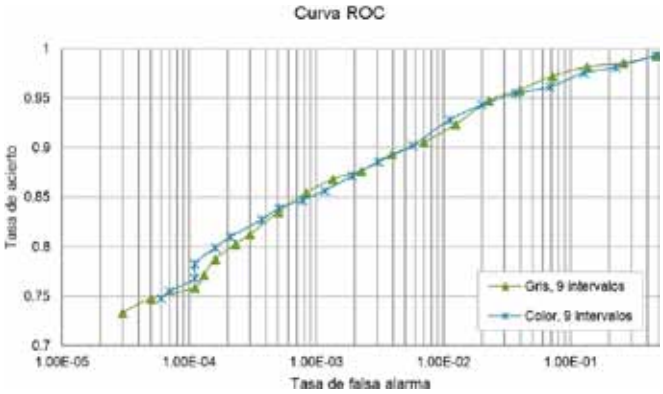

Figura 12. Comparación gradiente vectorial de color vs. gradiente escalar gris, 9 intervalos.

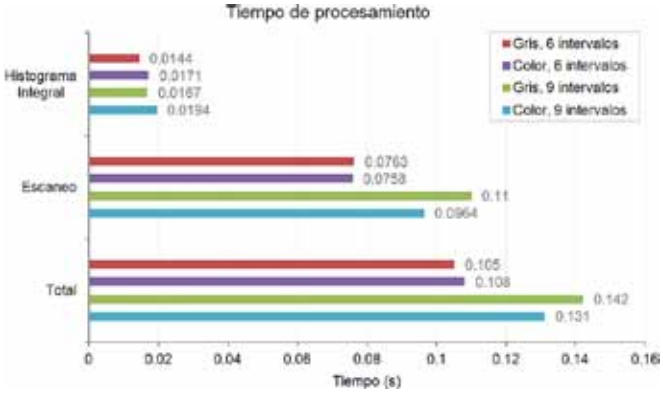

Figura 14. Tiempo promedio de procesamiento.

En general, la implementación que tuvo mejor desempeño fue utilizando gradiente vectorial para imagen a color e histogramas de seis intervalos. Con el clasificador completo, de 20 etapas, se obtuvo una tasa de acierto de 0.77 y una tasa de falsa alarma de 3×10^{-5}.

Comparación del tiempo de procesamiento

Para comparar el tiempo de procesamiento de las diferentes implementaciones, se utilizó un vídeo de prueba, que es un vídeo grabado, de 640×480 pixeles. El programa de prueba realiza el mismo procedimiento que cuando se trata de una captura en tiempo real: reduce la resolución a 320×240 y lo refleja, antes de iniciar la detección. Se analizaron los dos procesos que consumen mayor tiempo, que son la construcción del histograma integral y la clasificación de cada una de las ventanas de detección (5273 en total), los cuales representan aproximadamente el 85-89 % del tiempo total de procesamiento por cuadro de vídeo. Para ello se utilizó paralelización en el código de detección, cuyos resultados se muestran a continuación.

Estos resultados demuestran que fue acertado hacer la prueba de reducir el número de intervalos en los histogramas, ya que es considerablemente más eficiente. En cuanto a la tasa de acierto y de falsa alarma, ya se demostró anteriormente que utilizando menos intervalos no sólo no se deteriora el desempeño, sino que se mejora ligeramente. En la siguiente tabla se resumen las velocidades obtenidas para cada una de las implementaciones, evaluadas sobre el vídeo de prueba.

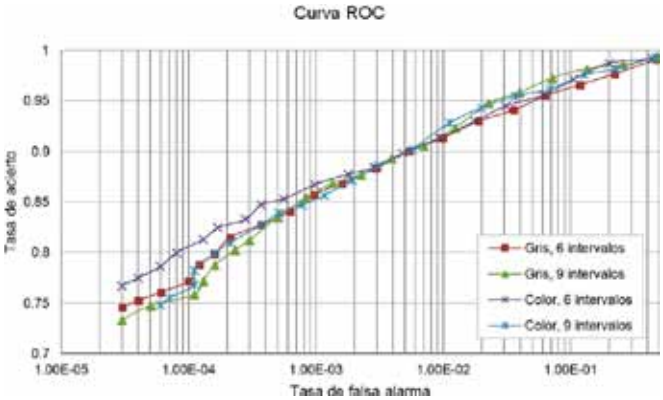

Figura 13. Curvas ROC de las 4 implementaciones.

Tabla 2. Velocidad de cuadro promedio.

Tipo	Velocidad de cuadro (fps)
Gris, 6 intervalos	3565
Color, 6 intervalos	3169
Gris, 9 intervalos	2980
Color, 9 intervalos	2736

Realizando estas pruebas se extrajo una lección importante, y es que los resultados no son tan fácilmente predecibles. Al principio se pensaba que utilizando el gradiente escalar gris iba a resultar más eficiente por ser más sencillo que el gradiente vectorial de color. Efectivamente, se redujo el tiempo para la construcción del histograma integral que incluye el cálculo del gradiente, sin embargo, se aumentó el tiempo de escaneo y clasificación, que realmente depende de cuántos clasificadores hay por etapa y cuántas etapas en promedio se deben evaluar para rechazar las muestras negativas. Se puede decir que el gradiente escalar gris produjo clasificadores menos eficientes.

6.3. Optimización con Intel TBB

También se realizaron pruebas para comparar el tiempo de procesamiento con y sin paralelización utilizando Intel TBB, para verificar que la ganancia en velocidad compensa el overhead que implica la propia paralelización. Como ya se ha explicado, el cálculo del histograma integral y el escaneo de las ventanas son dos buenos candidatos a ser paralelizados, ya que son los que determinan el tiempo total de procesamiento, y además presentan la propiedad de independencia de datos, que los hace elegibles a ser paralelizados.

Para la evaluación se utilizó el mismo vídeo que en las pruebas anteriores, con la implementación que utiliza gradiente vectorial de color e histogramas de seis intervalos. La siguiente tabla muestra los resultados.

Tabla 3. Tiempo de entrenamiento.

Proceso	Sin TBB	Con TBB
Histograma integral de orientación gradientes	25 ms	17 ms
Escaneo y clasificación de ventanas	136 ms	76 ms
Otros	15 ms	15 ms
Total	176 ms	108 ms
***Frame Rate* promedio**	**5.7 fps**	**9.3 fps**

La diferencia es significativa, sobre todo en el escaneo, en donde el tiempo se reduce en un 45% aproximadamente.

Intel TBB escala la aplicación a través de los múltiples núcleos disponibles de manera automática. Por ejemplo, en la computadora de doble núcleo que se utilizó en este proyecto, la aplicación solamente puede aprovechar el 50% de la capacidad del CPU (100 % de un solo núcleo) si no se utiliza *multithreading* con la librería de Intel TBB. En cambio, si se utiliza, puede aprovecharse la capacidad de ambos núcleos.

7. Conclusiones

La creación de interfaces naturales que utilizan visión artificial para la interpretación de gestos de las manos no es una idea nueva, sin embargo, su realización aún se enfrenta con diversos retos, principalmente el de lograr un buen rendimiento y ser robusto a la vez.

En este trabajo se desarrolló un sistema capaz de detectar las manos dentro de un vídeo, con un desempeño en tiempo casi real. Se inició con una investigación extensa de diferentes métodos utilizados para el reconocimiento de objetos y se llevó a cabo el proceso completo de desarrollo, desde la creación de las bases de datos de muestras hasta el entrenamiento del sistema y las pruebas del mismo.

Se combinaron ideas y aportes de diversos autores, determinando que una implementación factible del detector de manos es utilizando clasificadores en cascada con *boosting*, con descriptores basados en gradientes, que capturan efectivamente las características de las manos. Se evaluó el desempeño del sistema desarrollado, en términos de su precisión y tiempo de procesamiento, y se realizaron múltiples pruebas y comparaciones de variaciones de la implementación. A este respecto, se encontró que el mejor desempeño (entre las implementaciones evaluadas) se da cuando se utilizan gradientes vectoriales de color e histogramas de seis intervalos.

Debido a lo importante que es lograr un buen tiempo de respuesta, también se tomó muy en cuenta el aspecto de optimización, haciendo lo posible para elegir estructuras de datos adecuadas y utilizando librerías para paralelizar el código.

8. Trabajos futuros

Algo en lo que se está trabajando es en la evaluación de diferentes entornos. A pesar de que se ha determinado la tasa de acierto y de falsa alarma utilizando el conjunto de prueba, el desempeño del sistema durante una ejecución determinada en realidad puede variar mucho según las condiciones del entorno. En otras palabras, el desempeño reportado es para el conjunto de prueba en general, que contiene muestras muy variadas, tomadas en diferentes entornos; sin embargo, siempre habrá condiciones en las que el desempeño del sistema es bueno, y otras en las que no es tan bueno. Por esto, para entender mejor cómo responde el sistema bajo diferentes condiciones, también es necesario realizar un análisis cualitativo del comportamiento del sistema en diferentes entornos, considerando variaciones de iluminación, color, elementos de fondo y contraste.

Sin duda, a partir del trabajo realizado, los estudios futuros pueden tomar una variedad de direcciones diferentes, enfocándose en diferentes aspectos que se pueden mejorar y extender. Se pueden considerar modificaciones a la implementación actual, por ejemplo, utilizando bloques descriptores de otros tamaños y proporciones, cambiando el esquema de normalización de los bloques, utilizando otro tipo de gradiente, etc. También se pueden considerar cambios más significativos, como por ejemplo, probar con otros tipos de clasificadores.

Por otro lado, para una aplicación real, lo más probable es que sea necesario integrar un algoritmo de seguimiento *(tracking)*, que ayude a estabilizar las detecciones, reducir el error, y posiblemente también para mejorar la eficiencia, identificando las regiones con mayor probabilidad de que aparezca la mano.

Otro posible estudio es el desarrollo y aplicación de clasificadores multiclase, posibilitando la interacción a través de un vocabulario más extenso.

REFERENCIAS

[1] L. Bréthes, P. Menezes, F. Lerasle, and J. Hayet, "Face Tracking and Hand Gesture Recognition for Human-Robot Interaction," Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'04), vol. 1, pp. 1901-1906, 2004.

[2] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan, "Vision-Based Hand-Gesture Applications," *Communications of the ACM*, vol. 54, n.o2, pp. 60-71, febrero 2011.

[3] X. Zabulis, H. Baltzakist, and A. Argyros, "Vision-based Hand Gesture Recognition for Human-Computer Interaction," in *The Universal Access Handbook (Human Factors and Ergonomics)*, Constantine Stephanidis, Ed.: CRC Press, 2009, ch. 34, pp. 34.1-34.30.

[4] Thiago R. Trigo and Sergio R. M. Pellegrino, "An Analysis of Features for Hand Gesture Classification," 17th International Conference on Systems, Signals and Image Processing (IWSSIP'10), pp. 412-415, 2010.

[5] Paul Viola and Michael Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01), pp. 511-518, 2001.

[6] Mathias Kölsch and Matthew Turk, "Robust Hand Detection," 6th IEEE International Conference on Automatic Face and Gesture Recognition (FGR'04), pp. 614-619, 2004.

[7] Hardy Francke, Javier Ruiz-del-Solar, and Rodrigo Verschae, "Real-time Hand Gesture Detection and Recognition using Boosted Classifiers and Active Learning," *Electrical Engineering*, Universidad de Chile, 2007.

[8] Jorn A. Zondag, "Empirical Analysis of Histogram of Oriented Gradient Features, Weak Classifiers and Machine Learning Algorithms for Hand Detection," Department of Electrical Engineering, Eindhoven University of Technology, Master Graduation paper 2009.

[9] J. A. Zondag, T. Gritti, and V. Jeanne, "Practical study on real-time hand detection," 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII'09), pp. 1-8, 2009.

[10] Gary Bradski and Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, Primera edición ed.: O'Reilly Media, Inc., 2008.

[11] Qiang Zhu, Shai Avidan, Mei-chen Yeh, and Kwang-ting Cheng, "Fast human detection using a cascade of histograms of oriented gradients," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), pp. 1491-1498, 2006.

[12] H. C. Lee and D. R. Cok, "Detecting boundaries in a vector field," *IEEE Transactions on Signal Processing*, vol. 39, no. 5, pp. 1181-1194, 1991.

[13] Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2, pp. 886-893, 2005.

[14] OpenCV. (2012) OpenCV v2.4.1 Documentation. [Online]. http://docs.opencv.org/modules/ml/doc/boosting.html