

Detección de Ojos Mediante el Algoritmo Viola-Jones

Ezequiel Vijande, Lucero Guadalupe Fernandez

Resumen—En este trabajo se utilizó el algoritmo Viola-Jones para reconocimiento de caras y ojos en tiempo real para la posterior colocación de una imagen sobre los ojos. Se analizaron además los efectos de la iluminación sobre el algoritmo y el reconocimiento de features.

Index Terms—Viola-Jones, real-time face detection, real-time eye detection

I. INTRODUCCIÓN

El siguiente trabajo busca explicar y analizar una implementación del algoritmo de Viola-Jones en detección de ojos con el fin de agregar un efecto visual, superponiendo una imagen en cada ojo.

Se presenta una explicación del algoritmo utilizado, el proceso de superposición de la imagen y un análisis de como se ve afectado el funcionamiento del algoritmo dependiendo de la iluminación.

II. EL ALGORITMO VIOLA-JONES

El algoritmo de reconocimiento de objetos Viola-Jones puede detectar rápidamente y con alto nivel de confianza rostros en tiempo real. Combina los conceptos de *Features Haar*, *Imágenes Integrales*, el *Algoritmo AdaBoost* y *Clasificador en Cascada*. Para entender entonces el funcionamiento del algoritmo, se debe comprender el de cada concepto mencionado. Los mismos se detallan a continuación.

II-A. Features Haar

En primer lugar cabe mencionar que la motivación del uso de features por sobre píxeles se debe a que un sistema basado en features opera más velozmente que uno con base en píxeles. Un feature Haar consiste en regiones oscuras y claras y produce un único valor que resulta de tomar la suma de las intensidades de las regiones claras y sustraerla de la suma de las intensidades de las regiones oscuras. En el caso del algoritmo Viola-Jones los features a utilizar son los que se muestran en la Figura 1, y nos permiten extraer información útil de una imagen como bordes o líneas rectas.

II-B. Imágenes Integrales

Estos features rectangulares pueden computarse rápidamente usando una representación intermedia de la imagen, llamada imagen integral. Donde el valor para el punto (x, y) en la imagen integral es la suma de los píxeles hacia arriba y hacia la izquierda (inclusive) del punto (x, y) de la imagen original, según:

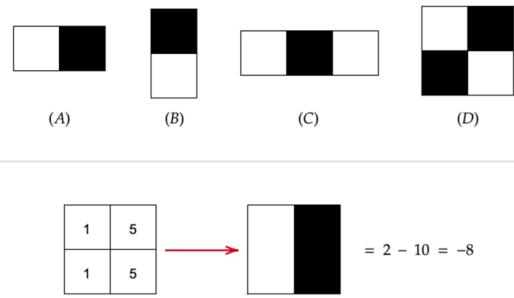


Figura 1: Features Haar (arriba) y cómo calcularlos (abajo).

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

donde $ii(x, y)$ es la imagen integral y $i(x, y)$ es la imagen original.

Ésto se muestra a continuación, en la Figura 2.

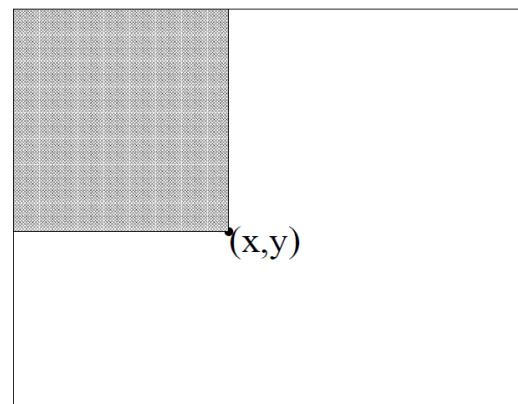


Figura 2: El valor de la imagen integral en (x, y) es la suma de los píxeles hacia arriba y hacia la izquierda.

Esta representación es esencial pues permite un cálculo mucho más rápido de una región rectangular, como ilustra la Figura 3. La región D se calcula entonces como $D = 4 + 1 - (2 + 3)$. Entonces, como la extracción de los features Haar involucra el cálculo de regiones rectangulares claras y oscuras, las imágenes integrales reducen ampliamente el tiempo necesario para cumplir esta tarea.

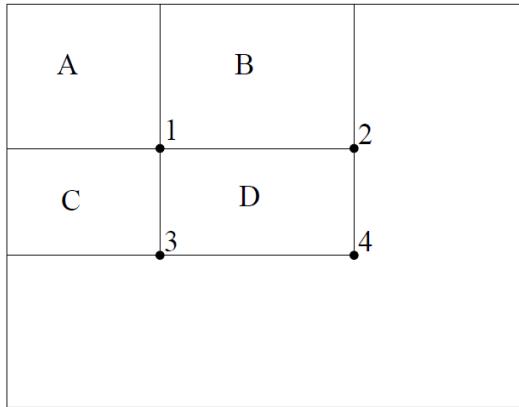


Figura 3: Cálculo para la región rectangular D.

II-C. Algoritmo AdaBoost

Este algoritmo se utiliza para seleccionar el mejor subset de features de entre todos los features disponibles, que son alrededor de 45,000. El output es un clasificador llamado *strong classifier*, y está compuesto de una combinación lineal de *weak classifiers* o clasificadores débiles, que son los mejores features. Se denominan *weak* porque no se espera que incluso los mejores features, individualmente, clasifiquen el set de train de manera correcta.

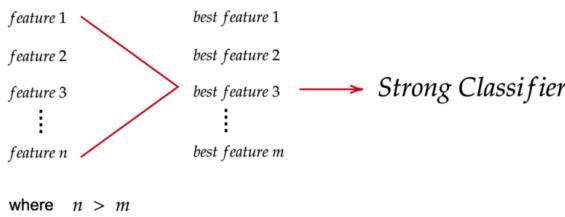


Figura 4: El objetivo del algoritmo AdaBoost es extraer los mejores features para luego, generar por combinación lineal un *strong classifier*.

II-D. Clasificador en Cascada

Un clasificador en cascada es un clasificador multi-etapa que permite una detección rápida y acertada a bajo costo computacional. La idea es que clasificadores más pequeños y eficientes pueden ser utilizados para rechazar gran mayoría de las sub-ventanas negativas mientras detecta gran cantidad de positivas. De esta manera, los clasificadores más simples se usan para rechazar la mayoría de las sub-ventanas antes que los más complejos sean llamados para reducir la tasa de falsos positivos.

La cascada inicia con un clasificador fuerte (*strong classifier*) de dos features como un filtro de cara. A modo de detalle, los dos features utilizados en esta primera etapa se muestran en la Figura 5. En la primera fila se muestran los features y en la fila siguiente se superponen con un rostro. El primer feature mide la diferencia de intensidad entre la región de los ojos y la

parte superior de las mejillas, que deriva de la observación de que la región de los ojos suele ser más oscura que las mejillas. El segundo compara a su vez, la intensidad de la región de los ojos con el puente de la nariz.

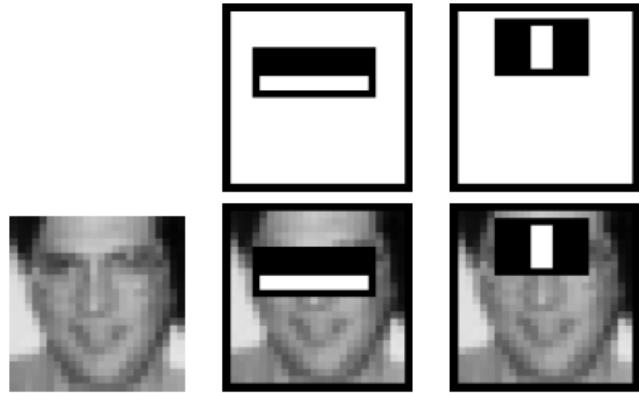


Figura 5: Los dos features utilizados en la primera etapa del clasificador.

Esta primera etapa reduce significativamente la cantidad de sub-ventanas que necesitan más procesamiento con pocas operaciones. En la Figura 6 se observa el diagrama en cascada. Un resultado positivo en la primera etapa triggerearía la evaluación del segundo clasificador. Un resultado positivo en esta etapa resulta en la evaluación de un tercer clasificador y así. Un resultado negativo en cualquier punto de la cascada lleva al rechazo de esa sub-ventana.

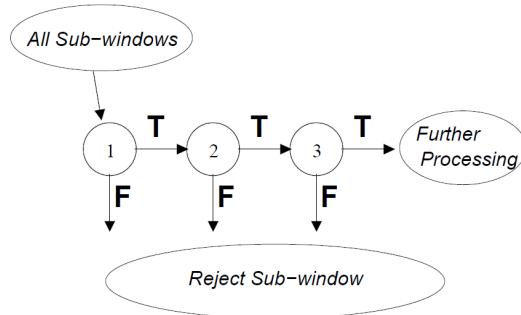


Figura 6: Diagrama del funcionamiento del clasificador en cascada.

Esta estructura refleja el hecho de que dentro de una imagen, hay una gran cantidad de sub-ventanas que son negativas, por lo que intenta rechazar la mayor cantidad posible de ventanas en una etapa temprana.

II-E. Aplicación en tiempo real

Viola y Jones entrenaron el algoritmo y obtuvieron que el clasificador en cascada óptimo tenía 32 capas de clasificadores y un total de 4297 features. Estos features fueron encontrados mediante AdaBoost. La primera etapa usa 2 features y rechaza el 60% de las no-caras mientras que detecta de manera correcta casi el 100% de las caras.

En este proyecto, el objetivo es aplicar Viola-Jones en tiempo real, tomando el feed de la webcam para su posterior procesamiento. Lo que permite este algoritmo, debido a su baja necesidad de cálculo es poder procesar cada frame para detectar si hay una o más caras y la ubicación de sus ojos. Esto se realiza entonces de la siguiente manera:

1. Se divide el frame en sub-ventanas de 24x24 píxeles de tamaño, para matchear el tamaño de la data de entrenamiento.
2. Se convierte en una imagen integral, la representación intermedia previamente mencionada.
3. Se lo alimenta al clasificador en cascada producido durante la fase de entrenamiento.
4. Se detecta una cara si una sub-ventana pasa todas las etapas del clasificador en cascada.

III. COLOCACIÓN DE LA IMAGEN SOBRE LOS OJOS

Una vez detectada la ubicación de los ojos se busca superponer sobre ambas regiones una imagen con fondo transparente, que se muestra a continuación:



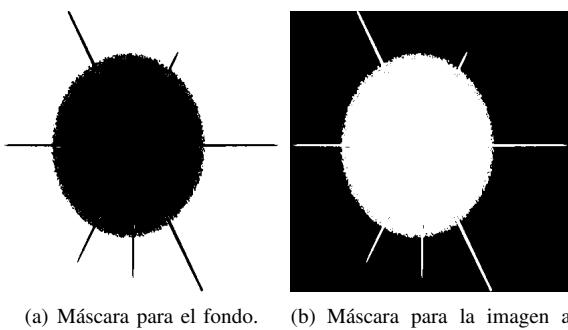
Figura 7: Imagen con fondo transparente utilizada.

En primer lugar, se extrae la región de interés, donde se encuentra el ojo, y donde se superpondrá la imagen mediante el uso de máscaras.

A continuación, se obtienen dos máscaras binarias, donde el '0' es representado por color negro y el '1', por píxeles blancos con valor 255, de la imagen a superponer.

La Figura 8(a) es la máscara a utilizar para el fondo, lo que vendría a ser parte del rostro, mientras que la Figura 8(b) es aquella a utilizar para la imagen transparente que se quiere superponer sobre los ojos.

Lo que resta hacer es un *and bit a bit* de la región de los ojos y su máscara correspondiente, la de la Figura 8(a). Por



(a) Máscara para el fondo.
(b) Máscara para la imagen a superponer.

Figura 8: Máscaras utilizadas.

otro lado, un *and bit a bit* de la imagen a superponer con la máscara 8(b) y sumar las dos, devolviendo la región de interés al frame para su posterior visualización en el feed de la webcam.

IV. EFECTOS DE LA ILUMINACIÓN

Se realizaron pruebas del funcionamiento del algoritmo para distintos escenarios de iluminación posibles, con el fin de analizar el rendimiento y las limitaciones del mismo. Para este análisis se probó el algoritmo en 14 escenarios con iluminación diferente, ya sea por la intensidad de la misma o por la ubicación de la fuente de luz. Estos escenarios pueden verse en la figura 9.

Se calcularon cuatro valores para cada uno de los escenarios, cada uno de estos valores corresponde al promedio de intensidad de todos los pixels dentro de una zona determinada de la imagen en escala de grises. Estos valores son:

1. Bg, utiliza toda la imagen.
2. F, solo utiliza el bounding box donde se encuentra la cara.
3. L, únicamente utiliza la mitad izquierda del bounding box de la cara.
4. R, únicamente utiliza la mitad derecha del bounding box de la cara.

Los valores obtenidos para cada escenario se proporcionan en la tabla I.

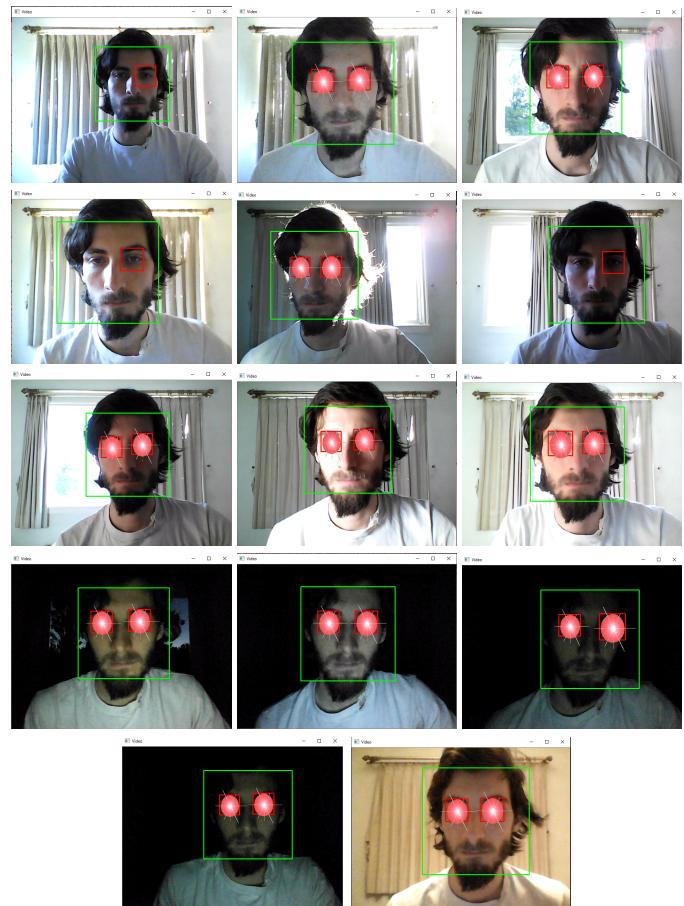


Figura 9: Escenarios de iluminación utilizados

Valores para cada imagen				
Fila, columna	Bg	F	L	R
(1,1)	150	83	82	140
(1,2)	162	100	90	111
(1,3)	152	107	131	83
(2,1)	166	112	141	86
(2,2)	126	83	73	94
(2,3)	139	66	84	48
(3,1)	139	70	93	47
(3,2)	140	121	166	75
(3,3)	182	125	159	91
(4,1)	45	45	40	50
(4,2)	40	45	37	53
(4,3)	17	18	16	20
(5,1)	24	25	22	27
(5,2)	148	104	109	100

Cuadro I: Valores de las imágenes en Fig 9

V. DISCUSIÓN DE LOS RESULTADOS

Con el fin de comparar qué escenarios de iluminación proporcionaron mejores resultados se tuvieron en cuenta dos factores principales. La rapidez y facilidad con la que el algoritmo detectó la cara y los ojos, y por otro lado la estabilidad de la detección. Con la estabilidad de la detección se quiere decir cuánto tiempo el algoritmo puede mantener correctamente la detección de la cara y de los ojos.

Los mejores resultados se obtuvieron para las imágenes de la Figura 10. Las cuatro imágenes están ubicadas de mejor a peor recorriéndolas empezando desde arriba a la izquierda y moviéndose hacia la derecha y luego hacia abajo. Los valores correspondientes a estas imágenes se muestran en la tabla II.

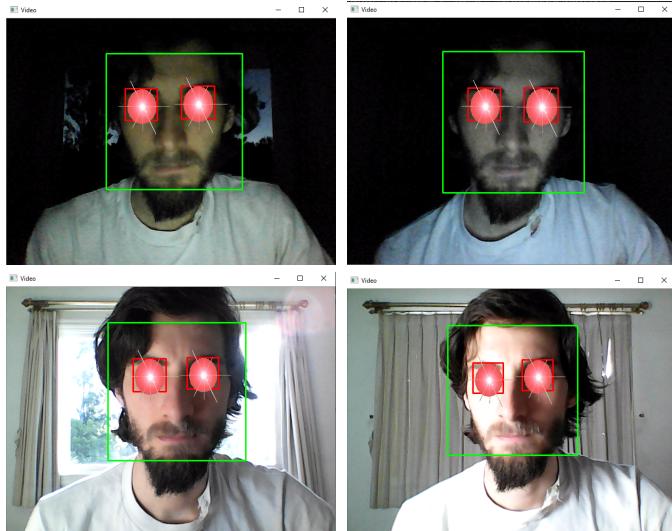


Figura 10: Escenarios con los mejores resultados

Valores para cada imagen					
Fila, columna	Bg	F	L	R	$ L - R $
(1,1)	45	45	40	50	10
(1,2)	40	45	37	53	16
(2,1)	152	107	131	83	48
(2,2)	140	121	166	75	91

Cuadro II: Valores de las imágenes en Fig 10

Las cuatro imágenes de la figura 10 tienen una rapidez y facilidad muy similar para encontrar y marcar la cara y los ojos correctamente. Las diferencias entre ellas fueron en la habilidad para mantener la detección correctamente. Las últimas dos solían confundir zonas incorrectas con los ojos ocasionalmente.

El primer patrón que se puede encontrar entre estas imágenes es que mientras menor es la diferencia entre iluminación entre ambas mitades de la cara, mejores son los resultados, como puede observarse en la última columna de la tabla II.

Además de esto, parece que el valor de la iluminación del fondo y de la cara no son tan importantes por sí solos, ya que dentro de las cuatro imágenes se tienen dos con iluminación muy baja y otras dos con iluminación alta. Lo que parece más relevante es el contraste entre la iluminación del fondo en relación a la iluminación de la cara. Las dos mejores imágenes tienen una iluminación de la cara mucho más alta que la del fondo. Mientras que en las dos peores, aunque la iluminación en general es más alta, el fondo es más brillante que la zona donde se encuentra la cara.

En cuanto a los peores resultados obtenidos, las cuatro imágenes correspondientes a los peores escenarios se pueden ver en la Figura 11. Las cuatro imágenes están ubicadas de peor a mejor recorriéndolas empezando desde arriba a la izquierda y moviéndose hacia la derecha y luego hacia abajo. Los valores correspondientes a estas imágenes se muestran en la tabla III.

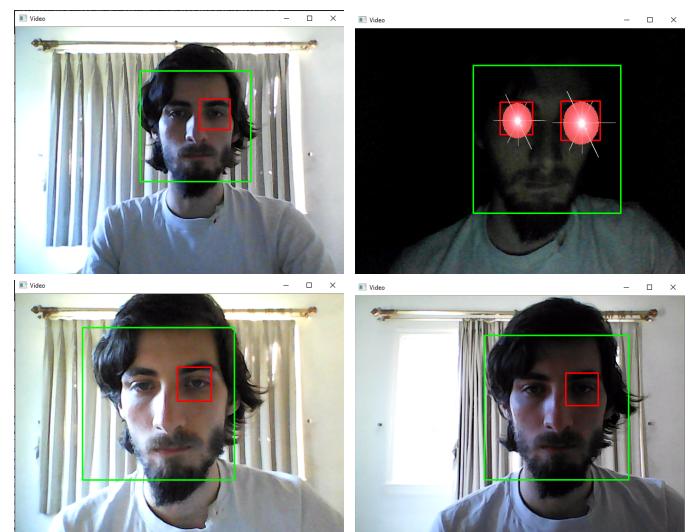


Figura 11: Escenarios con los peores resultados

Valores para cada imagen					
Fila, columna	Bg	F	L	R	$ L - R $
(1,1)	150	83	82	140	58
(1,2)	17	18	16	20	4
(2,1)	166	112	141	86	55
(2,2)	139	66	84	48	36

Cuadro III: Valores de las imágenes en Fig 11

Estas cuatro imágenes fueron categorizadas como las peores principalmente porque presentaron una gran dificultad en

siquiera detectar la cara y aun más en detectar correctamente la ubicación de ambos ojos. Además, las que pudieron llegar a detectar la cara y ambos ojos correctamente no fueron capaces de mantener la detección por mucho tiempo.

Omitiendo la imagen que se encuentra en la primera fila y en la segunda columna de la figura 11. Las tres imágenes restantes son muy similares, principalmente porque la iluminación del fondo es notablemente mas fuerte que la de la cara. Si bien la diferencia de iluminación entre la mitad izquierda y la derecha es apreciable en las tres imágenes, parece que el hecho de que el fondo esté más iluminado que la cara tiene mayor influencia, ya que hay imágenes con diferencias similares entre ambas mitades que dieron mucho mejores resultados.

También se puede destacar que el ojo que mas le cuesta detectar es el que esta a la izquierda en la imagen, que siempre es el que esta mas iluminado. Esto se debe a que el algoritmo de Viola-Jones funciona detectando las diferencias de iluminación entre el área superior de la mejilla y el ojo, al tener toda esa región con una gran iluminación se reduce notablemente el contraste y el algoritmo falla.

En cuanto a la imagen más oscura, este escenario presentó mucha dificultad para siquiera detectar la cara, esto se debe a que es el escenario con menor iluminación de todos, aunque la iluminación en sí no parece ser el factor con más influencia, al tener una iluminación tan baja se pasa un límite en el que el algoritmo ya casi no puede distinguir la cara en la imagen.

VI. CONCLUSIONES

Para concluir, se pudo verificar el correcto funcionamiento del efecto agregado en los ojos utilizando el algoritmo de Viola-Jones para la ubicación de los mismos.

Adicionalmente, se analizó el rendimiento del algoritmo para distintos escenarios de iluminación. Los dos factores que mas repercutieron en la habilidad del algoritmo de detectar la cara y los ojos fueron el contraste entre la iluminación del fondo contra la de la cara, y las diferencias de iluminación dentro de la cara. Particularmente, si la iluminación de una mitad de la cara es muy fuerte, el algoritmo tiene problemas en detectar el ojo en esa mitad porque el feature de Viola-Jones depende del contraste entre la mejilla y el ojo. También si se tiene una iluminación demasiado baja de la cara, el algoritmo es incapaz de detectarla. Sin embargo, no se pudo apreciar una pérdida de calidad progresiva al aumentar la intensidad de la iluminación en la cara, solo se debe prever estar por encima del límite mínimo necesario.

REFERENCIAS

- [1] P. Viola and M. Jones, Robust Real-time Object Detection in Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling, July 2001.