



UNIVERSIDAD SIMÓN BOLÍVAR  
Decanato de Estudios de Postgrado  
Maestría en Ciencias de la Computación

TRABAJO DE GRADO

**DETECCIÓN DEL CUERPO HUMANO EN  
SITUACIONES DE BÚSQUEDA Y RESCATE  
UTILIZANDO VISIÓN POR COMPUTADOR**

por

**Carlos D. Castillo**

Valle de Sartenejas, Junio de 2005



UNIVERSIDAD SIMÓN BOLÍVAR  
Decanato de Estudios de Postgrado  
Maestría en Ciencias de la Computación

# **DETECCIÓN DEL CUERPO HUMANO EN SITUACIONES DE BÚSQUEDA Y RESCATE UTILIZANDO VISIÓN POR COMPUTADOR**

Trabajo de Grado presentado a la Universidad Simón Bolívar

por  
Carlos D. Castillo

Como requisito parcial para optar al grado de  
Magister en Ciencias de la Computación

Realizado con la tutoría de la Profesora Carolina Chang

Valle de Sartenejas, Junio de 2005



UNIVERSIDAD SIMÓN BOLÍVAR  
Decanato de Estudios de Postgrado  
Maestría en Ciencias de la Computación

## DETECCIÓN DEL CUERPO HUMANO EN SITUACIONES DE BÚSQUEDA Y RESCATE UTILIZANDO VISIÓN POR COMPUTADOR

Este Trabajo de Grado ha sido aprobado en nombre de la Universidad  
Simón Bolívar por el siguiente jurado examinador:

Prof. Juan Carlos Grieco  
Presidente

Prof. Claudio Rocco  
Universidad Central de Venezuela  
Miembro Externo

Prof. Carolina Chang  
Miembro Principal-Tutor

Fecha: 12 Enero 2006.

# RESUMEN

Se presenta un trabajo que aborda el problema de detección del cuerpo humano en situaciones de búsqueda y rescate utilizando visión por computador. La detección de objetos es un problema fundamental en visión por computador. Existen diversas técnicas para atacar este problema. Trabajo previo muestra que el ajuste de plantillas utilizando correlaciones sencillas es efectivo para detectar objetos semi-rígidos en poses varias. Se propone estudiar enfoques basados en ajuste de plantillas para detección de objetos. La aplicación principal del método será la detección del cuerpo humano en situaciones de búsqueda y rescate a bordo de un robot móvil. Se desarrolla el método para detectar objetos semi-rígidos (posibles víctimas) en una escena compleja a tiempo real a partir de unas pocas fotos del objeto de interés. También se presenta un método semi-automático para extraer los contornos del objeto de interés a partir de fotos. Se estudia cómo evaluar la similitud entre dos figuras como paso para aplicar métodos de conglomerados clásicos. Estos conglomerados se utilizan para evaluar sistemáticamente la variedad de poses que puede capturar un conjunto dado de plantillas. Finalmente se compara con otro método del estado del arte y se evalúa el funcionamiento de la técnica aquí presentada en una aplicación robótica.

**Palabras clave:** visión por computador, detección de objetos, ajuste de plantillas, transformación de distancia, chamfering

# AGRADECIMIENTOS

Agradezco a mi tutora y mi amiga Carolina Chang por todo el apoyo técnico/científico, filosófico, infraestructural y financiero durante la realización de este trabajo.

Agradezco a mis amigos Miguel Castro, Eduardo Ruiz (Hawaii), Carolina Martínez, David Ojeda, Ezequiel Zamora, Luis Useche, Yolifé Arvelo, Jorge Guerra y Johanna Figueroa por compartir conmigo los pocos ratos de ocio.

Agradezco a las varias personas me dieron comentarios sobre la forma y fondo de este trabajo: Iliana Chollett, Eduardo Ruiz, Eduardo Klein y Miguel Castro.

Agradezco a los evaluadores de este trabajo los profesores Claudio Rocco y Juan Carlos Grieco por sus valiosos comentarios y por su disposición para proceder a la presentación de defensa tan rápidamente.

Agradezco a los varios modelos de peatones y víctimas para los varios videos y fotos: Juanito, Eduardo Ruiz, Miguel Castro, David Ojeda, Jorge Guerra, Ezequiel Zamora, Luis Useche, Yolifé Arvelo y Julio Castillo.

El apoyo financiero que permitió la realización de este trabajo vino de parte del Departamento de Computación y Tecnología de la Información de la Universidad Simón Bolívar, quien me contrató como ayudante docente durante los dos años que estuve haciendo la maestría. Agradezco a Jesús Ravelo (Jefe del Departamento de Computación y Tecnología de la Información durante ese tiempo) quien me apoyó en todo lo que necesité como ayudante docente de este departamento.

Los equipos que utilicé para este trabajo son propiedad de Carolina Chang y de su organización filantrópica que consta de una sola persona (ella). Agradezco el apoyo incondicional de Carolina Chang y de su organización.

Agradezco a la profesora Mariela Curiel (la coordinadora del Postgrado en Ciencias de la Computación durante ese tiempo) quien me apoyó en todo lo que llegué a necesitar como estudiante de este programa.

Finalmente, agradezco a mis padres Carlos Alberto y Fátima Katiuska, y mis hermanos Julio y Simón por su apoyo incondicional. A ellos cuatro dedico este trabajo.

# ÍNDICE GENERAL

. Agradecimientos	v
<b>1. Introducción</b>	<b>1</b>
1.1. Antecedentes . . . . .	2
1.2. Objetivos . . . . .	4
1.2.1. Objetivos Específicos . . . . .	5
<b>2. Transformación de Distancia</b>	<b>7</b>
2.1. Definiciones Importantes . . . . .	8
2.2. Eficiencia y Precisión de la Transformación de Distancia . . . . .	9
2.2.1. Chamfering . . . . .	10
2.3. Uso de la Transformación de Distancia . . . . .	11
<b>3. Una Aplicación Robótica para Detección de Peatones</b>	<b>13</b>
3.1. Sistema de Detección de Peatones . . . . .	14
3.2. Aplicación Robótica . . . . .	19

3.3.	<i>Bootstrapping</i> y Pruebas de Permutación . . . . .	21
3.3.1.	El Método de Monte Carlo para Bootstrap . . . . .	25
3.4.	Implementación y Puesta en Marcha . . . . .	26
3.5.	Conjuntos de Datos Utilizados y Procesamiento . . . . .	27
3.5.1.	TTK: Sistema de Detección Implementado . . . . .	29
3.5.2.	Detección de Piel . . . . .	29
3.5.3.	Saliency, Esquinas y Tracking . . . . .	31
3.5.4.	Puesta en Marcha en el Robot . . . . .	33
<b>4.</b>	<b>Detección de Víctimas en Situación de Búsqueda y Rescate</b>	<b>34</b>
4.1.	Sistema de Extracción de <i>Templates</i> . . . . .	35
4.1.1.	Conglomerados y Momentos de Hu . . . . .	35
4.2.	Sistema de Detección de Víctimas . . . . .	38
<b>5.</b>	<b>Experimentación y Resultados</b>	<b>42</b>
5.1.	Experimentos con el Sistema de Detección de Peatones . . . . .	42
5.1.1.	Comparación con el Método de Viola y Jones . . . . .	42
5.1.2.	A Bordo del Robot . . . . .	47
5.2.	Experimentos con el Sistema de Detección de Víctimas . . . . .	49
5.2.1.	El Sistema sobre las Fotos Obtenidas . . . . .	50
5.2.2.	El Sistema a Bordo del Robot . . . . .	50
<b>6.</b>	<b>Conclusiones</b>	<b>54</b>



6.1. Direcciones Futuras . . . . .	56
------------------------------------	----

# ÍNDICE DE FIGURAS

3.1. Ejemplo de la transformación de distancia. (a) es la imagen original, (b) es el resultado de la detección de contornos sobre la imagen original utilizando el método de Canny y (c) es el resultado de realizar una EDT sobre la imagen de contornos.	16
3.2. Las 12 plantillas utilizados por la versión inicial del sistema . . . . .	17
3.3. Ejemplos de la salida producida por el sistema sobre varias imágenes fuera de línea. Los rectángulos marcados sobre la imagen son los torsos detectados por el sistema. . . . .	18
3.4. Selección de caraterísticas vía varios tamaños de tablero de ajedrez. Cuadrados blancos representan píxeles seleccionados y cuadrados negros representan píxeles no seleccionados. . . . .	19
3.5. Curva ROC del Clasificador SVM. Arriba: Curva ROC del clasificador SVM con un kernel RBF (función de base radial) y $C=5$ . Ambos clasificadores son de similar complejidad. Observar el pobre desempeño de los kernels lineales y cuadráticos. Abajo: Curva ROC del clasificador SVM con un kernel RBF y distintas selecciones de características vía tablero de ajedrez. La pérdida de precisión se puede observar en la Tabla 3.1. . . . .	20
3.6. Valores de $D(T, I)$ para una imagen. Arriba: Imagen de prueba. Abajo: gráfico de contorno de los valores de $D(T, I)$ . . . . .	22

3.7. Valores de $D(T, I)$ para una imagen. Arriba: Imagen de prueba. Abajo: gráfico de contorno de los valores de $D(T, I)$ . . . . .	23
3.8. Una simplificación del código del algoritmo de detección implementado. La función <b>corrlista</b> calcula el error de alineación. . . . .	30
3.9. Algoritmo detección de piel basado en YCbCr . . . . .	32
4.1. Diagrama del sistema de funcionamiento del sistema de detección de víctimas. . .	39
4.2. Ejemplos de extracción de templates: (a) es la imagen original, (b) es la imagen de contornos en escala de grises y (c) es una imagen de la componente fuertemente conexa más grande del grafo de vecindad resultante de la aplicación de un umbral de (b). . . . .	40
4.3. Detección de una víctima alrededor de escombros. (a) escombros, (b) víctima ocluida por escombros y (c) el sistema detectando una víctima alrededor de escombros.	40
4.4. Detección de víctimas a varias distancias. (a) imagen original, arriba: detección más cercana, abajo: detección más lejana, (b) imagen marcada . . . . .	41
4.5. Resistencia a oclusión alrededor de características parciales no críticas. (a) una vista no muy ocluida, (b) una vista moderadamente ocluida y (c) una vista muy ocluida. . . . .	41
5.1. Las cuatro características de estilo de Haar utilizadas por el sistema de Viola y Jones . . . . .	44
5.2. Curva ROC del sistema TTK sobre las imágenes rotuladas. . . . .	47
5.3. Curva ROC del sistema CHLF sobre las imágenes rotuladas. . . . .	48

# ÍNDICE DE CUADROS

3.1. Selección de características vía tablero de ajedrez: selección hecha y media y desviación standard de la tasa de clasificación haciendo validación de 5-partes del clasificador final (de confirmación) de torsos. . . . .	17
3.2. Ejemplo de detección de piel. La imagen YCbCr tiene la banda Y en R, Cb en G y Cr en R. . . . .	31
4.1. Clusterización de los <i>templates</i> de peatones . . . . .	38
4.2. Clusterización de los <i>templates</i> de víctimas . . . . .	39
5.1. Ejemplo de una tabla de contingencia o matriz de confusión . . . . .	45
5.2. Tabla de contingencia para TTK sobre los ejemplos rotulados de prueba . . . . .	45
5.3. Tabla de contingencia para CHLF sobre los ejemplos rotulados de prueba . . . . .	45
5.4. El sistema funcionando a bordo del robot, en un pasillo en un ambiente de oficinas.	52
5.5. El sistema funcionando a bordo del robot, en un pasillo en un ambiente de oficinas.	53

# LISTA DE ABREVIATURAS

<b>AUC</b>	Área bajo la curva ROC
<b>CHLF</b>	Classifier using Haar Like Features
<b>DT</b>	Transformación de Distancia
<b>EDT</b>	Transformación de Distancia Euclidiana
<b>FP</b>	Falsos Positivos
<b>FN</b>	Falsos Negativos
<b>LIBSVM</b>	Librería de SVMs en C++, con enlaces para ser usado desde Python
<b>PIL</b>	Python Imaging Library
<b>RBF</b>	Función de Base Radial (proviene de su traducción en inglés, Radial Basis Function)
<b>RGB</b>	Red-Green-Blue
<b>ROC</b>	Receiver Operating Characteristic
<b>SVM</b>	Support Vector Machine
<b>TTK</b>	Template Toolkit
<b>VN</b>	Verdaderos Negativos
<b>VP</b>	Verdaderos Positivos

## Capítulo 1

# INTRODUCCIÓN

“Las verdades que revela la ciencia  
superan siempre a los sueños que  
destruye”

---

Ernest Renan

La capacidad visual de los seres humanos es sorprendente. Una persona puede detectar y reconocer a un objeto inmediatamente con tan solo una mirada. Los computadores, sin importar qué tan poderosos sean, todavía no son buenos en este tipo de actividades. La visión por computador se ocupa de la detección, segmentación, localización y reconocimiento de objetos en imágenes [18, 20, 48, 49].

La detección de objetos es un problema fundamental de visión por computador. Aunque por sí solo es de interés desde un punto de vista computacional, también ocurre que la mayoría de los objetivos de robots inteligentes requieren de un entendimiento de la situación de su entorno y la visión (y la capacidad de detectar objetos utilizando visión) es un sentido de suma importancia para poder lograr esto [27, 52].

Una de las actividades fundamentales de la robótica es facilitar la realización de tareas que serían muy peligrosas para una persona. Este tipo de tareas típicamente ha sido desactivación de minas antipersonales y bombas, y exploración de sitios peligrosos (volcanes, superficie de planetas

lejanos) entre otros. Más recientemente, se ha incluido actividades de búsqueda y rescate en desastres urbanos [7]. Dentro de este tipo de actividades, se encuentra la detección automática de víctimas.

Los retos en la detección visual de objetos son:

- **Pose:** Debido a que el cuerpo humano es un objeto semi-rígido que puede tener una gran cantidad de poses, su detección automática es complicada.
- **Componentes Estructurales:** Los componentes estructurales (como la ropa) varían de persona a persona, lo que hace que haya gran variabilidad entre patrones de la misma pose.
- **Oclusión:** Los objetos pueden estar parcialmente ocluidos por otros objetos.
- **Orientación:** El cuerpo humano varía en las diferentes orientaciones con respecto al eje de la cámara.
- **Condiciones de la imagen:** Condiciones como iluminación y características de la cámara afectan la apariencia del cuerpo humano.

En la investigación presente se plantea un método para detección automática de víctimas en situaciones de búsqueda rescate hechas por un robot, basada en una técnica llamada *template matching*, que se explicará en detalle en las siguientes secciones.

## 1.1. Antecedentes

Recientemente para detectar objetos en escenas se han utilizado muchos métodos: vecino más cercano después de extraer componente principales [40], redes neurales convolucionales [34], máquina de vector de soporte sobre coeficientes de wavelet [41,45], stereo-visión y redes neurales [35].

Los enfoques utilizados hasta ahora para resolver problema de detección automática de personas incluyen alguna de las siguientes suposiciones:

- El sistema no tiene que funcionar a tiempo real.
- El sistema de captura de video está estático en la escena por lo que restando cuadros consecutivos se sabe qué se debe revisar [45,56]. Para este tipo de aplicaciones la literatura relevante se encuentra en el área de vigilancia automática. Esto no es cierto en este caso porque el software va a estar ejecutándose en un robot en un ambiente complejo y no estructurado. No se dispone de un modelo explícito de la escena.
- Utiliza dos o más cámaras especialmente calibradas (stereo) y hace revisiones en planos de igual profundidad y paralelos a la cámara [35].
- Es factible utilizar una cámara de temperatura y se debe reportar como acierto cualquier temperatura por encima de la media ambiental. La literatura relevante de esta área viene de la robótica para búsqueda y rescate. Esto no es cierto debido a que una de las posibles situaciones a evaluar podría ser un incendio (o cualquier desastre que aumente la temperatura media ambiental). También es deseable hacerlo desde el punto de vista de visión por computador porque los sensores de cámara son cada vez más económicos.

Estas suposiciones en general no son válidas, y están relacionadas con los requerimientos de cada aplicación para la cual se desarrolla el sistema.

Gavrila et al. [25,26] utilizaron *template matching* sobre un mapa de distancia para detectar peatones en poses libres (como parte de un proyecto para evitar accidentes viales contra peatones) que son confirmados utilizando un clasificador RBF (función de base radial) a bordo de un automóvil. Se intentó obtener una versión del sistema Chamfer de Gavrila, pero el sistema y los datos relacionados (base de datos de imágenes, *templates*, etc) son propiedad de DaimlerChrysler, donde trabaja Gavrila<sup>1</sup>, y desafortunadamente no se pudieron obtener. Según lo descrito en su trabajo, el sistema funciona muy bien, pero por limitaciones de propiedad intelectual no se pudo comparar en detalle con el sistema aquí presentado.

Konolige [3] y su equipo de investigación en SRI (Stanford Research Institute) tienen un sistema basado en correlaciones (no *template matching*) y visión estereoscópica llamado Small

---

<sup>1</sup>Dariu Gavrila, comunicación personal, 2004.



Vision System (SVS). En demostraciones (en video, descargables de la página web de Konolige) se observa que el sistema detecta efectivamente vistas frontales de peatones en un ambiente de oficina. Desafortunadamente la licencia SVS cuesta \$250 y se necesita un par de cámaras calibradas especialmente que cuestan \$1600 cada una, que vende la compañía de Konolige.

Viola y Jones [55] recientemente presentaron un sistema de detección de objetos que utiliza características sencillas basadas en suma y resta de valores de píxeles de regiones cercanas en lugar del valor de los píxeles directamente. Estas características sencillas codifican conocimiento del dominio que es difícil de aprender utilizando datos de entrenamiento. El sistema de Viola y Jones utiliza un algoritmo similar a AdaBoost [22] para entrenar clasificadores.

Más recientemente Bertozzi et al. [1] utilizaron figuras parametrizadas y simetría para detectar peatones y luego extendieron su aplicación utilizando filtros de Kalman (Broggi [8]) para detectar efectivamente dónde están los peatones. Los resultados de estos trabajos son fuertes, pero luego estos autores definieron una metodología para evaluar el desempeño de algoritmos de detección de peatones [2]. Desafortunadamente el financiamiento privado que obtiene el grupo de investigación no le permite “publicar el software ni detalles de los algoritmos”<sup>2</sup>.

## 1.2. Objetivos

El objetivo del presente trabajo es mostrar cómo técnicas integradas basadas en *template matching* de detección visual de objetos son competitivas con otras técnicas de estado del arte [33, 45, 55] tanto en desempeño como en precisión. Esto puede ser visto como un halago a las técnicas basadas en *template matching* o una profunda ofensa a técnicas de detección visual de objetos semirígidos basadas en técnicas más elaboradas.

Se eligió utilizar técnicas basadas en *template matching* debido a su simplicidad y su viabilidad para capturar clases de objetos semirígidos sin una textura característica a partir de un descriptor sencillo como lo es el borde de una imagen.

---

<sup>2</sup>Alberto Broggi, comunicación personal, 2004.

El enfoque que aquí se presenta utiliza técnicas clásicas de *template matching* sobre la transformación de distancia similar al enfoque descrito por Borgefors [5] en 1988 y Gavrilu et al [23] en 2000. El método también es similar a lo presentado por Brunnelli y Poggio [9] en 1995. Más recientemente Thayananthan et al [54] presentaron una serie de extensiones al método para funcionar mejor en escenas *cluttered* basados en un histograma de figuras llamados *shape context*.

En este trabajo se presentan modernizaciones a los métodos de ajuste de plantillas sobre transformación de distancia. Las modernizaciones están basadas en métodos probabilísticos, estadísticos y de simulación.

En el trabajo se presentan dos casos de estudio distintos como caso de prueba de los métodos implementados:

- **Detección de Peatones:** Caso de prueba inicial, debido a que los conjuntos de datos son más manejables y en variedad de poses es casi tan rico como el dominio de víctimas. Además existen varias aplicaciones o sistemas que involucran detección de peatones lo cual hace factible compararse con otros sistemas del estado del arte.
- **Detección de Víctimas:** La aplicación de detección de víctimas en situación de búsqueda y rescate en desastres urbanos es el objetivo principal de este trabajo. Este trabajo se enmarca en un proyecto de diseño, construcción y desarrollo de robots para búsqueda y rescate [7].

### 1.2.1. Objetivos Específicos

Los objetivos específicos de este estudio son:

1. Desarrollo de una herramienta para probar métodos de *template matching* y/o clasificación posiblemente para detección de objetos de interés sobre imágenes arbitrarias.
2. Experimentación con esta herramienta y conjuntos de datos públicos de peatones y caras, para evaluar la viabilidad del método.

3. Creación de un conjunto de datos de víctimas en situación de búsqueda y rescate en ambientes urbanos. Este conjunto de datos puede servir de referencia en investigaciones futuras en esta área.
4. Desarrollo de una herramienta que permita obtener de una manera semi-automática los *templates* de un conjunto de imágenes del objeto de interés.
5. Implementación e implantación de una versión en tiempo real del método de detección sobre una plataforma robótica.
6. Puesta a tono del sistema de detección en tiempo real sobre la plataforma robótica, sobre todo en cuanto a cómo afecta la operación autónoma del robot y cómo mejorar el método de navegación.
7. Evaluación experimental en una simulación de desastre urbano.

El trabajo se organiza de la siguiente manera: en el próximo capítulo se presenta una introducción a la teoría de la transformación de distancia. Después se presenta una aplicación robótica desarrollada en este trabajo que utiliza transformación de distancia para realizar la detección de peatones en imágenes arbitrarias. En el Capítulo 4 se presenta un método para detectar víctimas en situación de búsqueda y rescate que utiliza ajuste de plantillas sobre múltiples características. Luego, en el Capítulo 5 se presentan los resultados de la experimentación realizada con los métodos presentados en los dos capítulos anteriores. Y finalmente, se dan las conclusiones de este trabajo y las recomendaciones para investigación futura.

## Capítulo 2

# TRANSFORMACIÓN DE DISTANCIA

“No toda distancia es ausencia, ni todo  
silencio olvido“

---

Vox Populi

El objeto principal de una transformación de distancia es calcular la distancia de un punto a un objeto (un conjunto de píxeles). Esto es, la distancia de un punto  $p$  al punto más cercano  $q$  perteneciente al objeto.

Cuando se necesita conocer esto para varios puntos  $p$ , es más rápido consultarlo en un mapa de distancia, lo cual es una imagen donde la distancia ha sido pre-calculada en todos los puntos. La transformación de distancia (DT, por sus siglas en inglés) es la operación que calcula un mapa de distancia dada una imagen binaria que representa al objeto.

Hay varias maneras de calcular la distancia entre dos puntos en una cuadrícula discreta: distancia Euclidiana, distancia Manhattan, y otras.

La idea de utilizar la transformación de distancia en detección visual de patrones no es reciente (Rosenfeld, Borgefors [5]), pero su simplicidad es sorprendente y logra compararse muy bien con otros métodos del estado del arte como por ejemplo: Nayar et al [40], Papageorgiu et al. [45], LeCun et al. [34] y Viola y Jones [55].

## 2.1. Definiciones Importantes

Sea  $I$  una imagen binaria de un objeto  $O$  (y un no-objeto  $O'$ ). Una **transformación de distancia** construye un mapa de distancia  $D$  en el que cualquier píxel es la distancia al objeto  $O$ . Esto es:

$$D(p) = \min\{\text{dist}_M(p, q), q \in O\} \quad (2.1)$$

donde  $\text{dist}_M(p, q)$  es la distancia entre  $p$  y  $q$  usando la métrica  $M$ . Se considerarán las siguientes métricas:

$$\text{dist}_4(p, q) = |p_x - q_x| + |p_y - q_y| \quad (2.2)$$

$$\text{dist}_8(p, q) = \max\{|p_x - q_x|, |p_y - q_y|\} \quad (2.3)$$

$$\text{dist}_{(A:B)}(p, q) = A \max\{|p_x - q_x|, |p_y - q_y|\} - (B - A) \min\{|p_x - q_x|, |p_y - q_y|\} \quad (2.4)$$

$$\text{dist}_e(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad (2.5)$$

$$\text{dist}_E(p, q) = (p_x - q_x)^2 + (p_y - q_y)^2 \quad (2.6)$$

Estas distancias se llaman: Manhattan (2.2), tablero de ajedrez (2.3), chamfer (2.4), Euclidiana (2.5), Euclidiana al cuadrado (2.6).

Un concepto importante relacionado es el de diagrama de Voronoi [52]. El diagrama de Voronoi divide el plano en regiones que bordean al píxel de interés.  $VP(p)$  es el polígono de Voronoi alrededor del píxel  $p$  del objeto. Para cada píxel  $p \in O$ ,  $VP(p)$  es la parte de la imagen

$I$  que satisface:

$$VP(p) = \{q \in I | \forall r \in O : \text{dist}_M(p, q) \leq \text{dist}_M(q, r)\} \quad (2.7)$$

El conjunto de vecinos de un punto  $p$  se llama  $N(p)$ :

$$N(p) = \{q = p + n | n \in N\} \quad (2.8)$$

donde  $N = N((0, 0))$  se llama vecindad. La vecindad es una bola con respecto a la métrica  $M$ :

$$N = B_d = \{n | \text{dist}_M(n, (0, 0)) < d\} \quad (2.9)$$

con alguna métrica  $M$ . Son de interés las 4-vecindades y las 8-vecindades:

$$N_4 = \{n | \text{dist}_4(n, (0, 0)) = 1\} \quad (2.10)$$

$$= \{(1, 0), (-1, 0), (0, 1), (0, -1)\} \quad (2.11)$$

$$N_8 = \{n | \text{dist}_8(n, (0, 0)) = 1\} \quad (2.12)$$

$$= N_4 \cup \{(1, 1), (1, -1), (-1, 1), (-1, -1)\} \quad (2.13)$$

## 2.2. Eficiencia y Precisión de la Transformación de Distancia

La transformación de distancia es una transformación global. Una aplicación directa de la definición implicaría considerar todos los píxeles del objeto  $O$  para calcular la  $DT$  de cualquier píxel no-objeto  $O'$ .

En procesamiento de imágenes se trata de aproximar la  $DT$  utilizando transformaciones locales<sup>1</sup>. La forma más sencilla de hacer esto es aplicar la definición de la ecuación (2.1) suponiendo que cada píxel puede ser deducido a partir de los píxeles de su vecindad. Esto se puede demostrar

---

<sup>1</sup>Esto es, que el valor en el mapa de distancia de un píxel solo dependa de los píxeles de su vecindad.

para las métricas de Manhattan, tablero de ajedrez y chamfer pero no para distancia Euclidiana. Esto da pie a una familia de algoritmos que Borgefors [5] llamó *chamfering*. Desafortunadamente la métrica de chamfer (para la cual se puede calcular la transformación de distancia de una manera muy rápida) da aproximaciones burdas a la distancia Euclidiana. Hay enfoques que explotan la rapidez de la transformación de distancia para una métrica chamfer y después reparan algunos píxeles para convertirlo en cuasi-Euclidiano [5].

### 2.2.1. Chamfering

Las transformaciones de distancia chamfer explotan la suposición de que es posible deducir el valor de un píxel a partir de sus vecinos. Esta suposición es correcta para métricas regulares: Una métrica es regular si para todo  $p, q$  tal que  $\text{dist}_M(p, q) \leq 2$ , existe un  $r$  diferente de  $p$  y  $q$  tal que  $\text{dist}_M(p, q) = \text{dist}_M(p, r) + \text{dist}_M(r, q)$ .

Esta propiedad es cierta para Manhattan, tablero de ajedrez y chamfer, entre otras, pero no para distancia Euclidiana cuando los píxeles  $p, q, r$  están restringidos a localizaciones en la cuadrícula entera.

Para estudiar la diferencia máxima de la distancia Euclidiana se debe considerar la distancia chamfer:

$$\text{dist}_{(d_1:d_2)}(p, q) = m_2 d_2 + (m_1 - m_2) d_1 \quad (2.14)$$

donde  $m_1 = |q_x - p_x|$  y  $m_2 = |q_y - p_y|$ , y  $m_1 > m_2$ , la diferencia entre distancia Euclidiana y chamfer es:

$$\sqrt{m_1^2 + m_2^2} - m_2 d_2 - (m_1 - m_2) d_1 \quad (2.15)$$

Si se fija  $d_1 = 1$  y  $d_2 = d$  (con  $d < 2$ ), el  $d$  óptimo se puede conseguir minimizando el máximo de (2.15). El máximo ocurre cuando  $m_2 = 0$ ,  $m_2 = m_1$  o cuando  $m_2$  tiene el valor que hace a la derivada con respecto a  $m_2$  es cero. Para  $m_2 = 0$ , ocurre que (2.15) es cero. Para  $m_2 = m_1$

(2.15) se convierte en:

$$(\sqrt{2} - d)M \quad (2.16)$$

donde  $M = m_1 = m_2$ . La derivada de (2.15) con respecto a  $m_2$  es cero cuando  $m_2 = ((d - 1)/\sqrt{2d - d^2})$ . Substituyendo este valor en (2.15) se obtiene:

$$(\sqrt{2d - d^2} - 1)M \quad (2.17)$$

donde  $M = m_1$ . El máximo de (2.16) y (2.17) ocurre cuando se cruzan:

$$d = 1/\sqrt{2} + \sqrt{\sqrt{2} - 1} = 1.351 \quad (2.18)$$

Por lo tanto el límite superior para (2.15) es:

$$(1/\sqrt{2} - \sqrt{\sqrt{2} - 1})M = 0.06M \quad (2.19)$$

Por motivos de eficiencia computacional no son deseables operaciones de punto flotante. Para resolver esto, Borgefors [5] propone una aproximación entera subóptima  $d_1 = 3$  y  $d_2 = 4$ , y después dividir (enteramente) la transformación de distancia resultante por 3 en cada punto. En este caso la cota superior de la diferencia entre distancia Euclidiana y chamfer es  $\sqrt{2} - 4/3 = 0.08M$ . Esta cota es mucho mejor que los límites para métricas Manhattan (en este caso la cota es  $((\sqrt{2} - 2)M = -0.59M)$  y tablero de ajedrez (en el que la cota es  $(\sqrt{2} - 1)M = 0.41M$ )

### 2.3. Uso de la Transformación de Distancia

En este trabajo el uso que se le da a la transformación de distancia es para calcular la alineación de una silueta a una imagen. Estas imágenes pueden tener variada información como por ejemplo bordes de una imagen visible, bordes de las posiciones donde se detecta presencia de piel, bordes de una imagen en infrarrojo cercano, entre otros. Con la transformación de distancia de una imagen de bordes se obtiene un mapa de distancia a esos bordes. Este mapa



de distancia guía una familia de algoritmos conocidos como *template matching*, estos algoritmos detectan patrones visuales mediante la alineación (o similitud) entre una plantilla y una imagen de borde. No habría problema si la plantilla fuera idéntica a los bordes que aparecen en la imagen de bordes, pero esto en general no es cierto y la transformación de distancia permite compensar estas disimilitudes entre la imagen y la plantilla. En general, las plantillas capturan gran variabilidad en poses del objeto de interés y presentan un método exitoso para detección de objetos semirígidos en poses varias.

En este trabajo se calculan alineaciones con múltiples transformaciones de distancia de imágenes de contorno:

- Bordes de la imagen RGB
- Bordes de las posiciones donde se detecta piel en la imagen
- Bordes de una imagen de puntos de interés

Estas características se integran y permiten obtener mayor precisión de clasificación. La transformación de distancia estará en el centro de la mayoría de los métodos presentados en este trabajo. En el siguiente capítulo se muestra como utilizar este tipo de transformaciones para detectar peatones en imágenes arbitrarias (escenas complejas). Después, se presenta cómo usar este tipo de técnicas para detección de víctimas en operaciones de búsqueda y rescate asistidas por robot.

## Capítulo 3

# UNA APLICACIÓN ROBÓTICA PARA DETECCIÓN DE PEATONES

Caminante, son tus huellas  
el camino y nada más;  
caminante, no hay camino,  
se hace camino al andar.  
Al andar se hace camino  
y al volver la vista atrás  
se ve la senda que nunca  
se ha de volver a pisar.  
Caminante, no hay camino  
sino estelas en la mar...

---

Cantares. Antonio Machado

En este capítulo se presenta una versión extendida y detallada del trabajo presentado por Castillo y Chang [13], en el cual se desarrolló un método que integra *template matching*<sup>1</sup> con clasificadores<sup>2</sup> y se probó para detección peatones. Este trabajo se hizo como un caso de prueba inicial, debido a que los conjuntos de datos son más manejables y en variedad de poses es casi tan rico como el dominio de víctimas. Además existen varias aplicaciones o sistemas que involucran detección de peatones lo cual hace factible compararse con otros sistemas del estado del arte.

---

<sup>1</sup>En español, ajuste de plantillas

<sup>2</sup>En la implementación particular se utilizó un SVM, pero podría ser cualquier método de clasificación

### 3.1. Sistema de Detección de Peatones

Como paso inicial se desea detectar en una imagen estática arbitraria dónde están los peatones y marcarlos adecuadamente. Este tipo de sistemas es de interés en aplicaciones de vigilancia automática (control de acceso, vigilancia de estacionamientos), protección de accidentes viales contra peatones, indexamiento por contenido (por ejemplo de juegos deportivos en televisión).

Para esto, se desarrolló un sistema que en su centro utiliza *template matching* realizando una transformación de distancia Euclidiana para evaluar objetos candidatos. Estos objetos candidatos son verificados inmediatamente utilizando un Support Vector Machine [11, 17] (SVM) especializado para esa componente.

El primer paso del procedimiento es el preprocesamiento. Cada imagen de entrada se convierte a escala de grises y se le determinan los contornos utilizando métodos estándares provistos por las librerías gráficas utilizadas (utilizando el método de Canny [12]). Después de eso se convierte la imagen resultante utilizando una transformación de distancia Euclidiana en un mapa de distancia. En la Figura 3.1 se muestra el resultado del preprocesamiento.

Después que la imagen ha sido adecuadamente preprocesada, comienza el paso de ajuste de plantillas. Como lo describe Gavrilu [23], una imagen  $I$  está ajustada<sup>3</sup> a una plantilla  $T$  cuando:

$$D(T, I) \leq \theta \quad (3.1)$$

donde  $\theta$  es un umbral fijado por el usuario, y es la máxima disimilaridad aceptable entre el mapa de distancia y la plantilla.  $D(T, I)$  se puede definir de varias maneras. En el sistema que se presenta, se experimentó con dos definiciones diferentes: error de alineación promedio y error de alineación máximo. En el caso del error de alineación promedio,  $D(T, I)$  está dado por:

$$D(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t) \quad (3.2)$$

---

<sup>3</sup>Haciendo “match”

donde  $|T|$  es el número de características en  $T$  y  $d_I(t)$  es la distancia entre una característica  $t \in T$  y la característica más cercana en  $I$ .

En el caso del error máximo de alineación  $D(T, I)$  está dado por:

$$D(T, I) = \max_{t \in T} d_I(t) \quad (3.3)$$

donde  $d_I(t)$  es la distancia entre una característica  $t \in T$  y la característica más cercana en  $I$  tal como es calculada por la EDT.

La distancia promedio de alineación cuantifica el error de alineación como la distancia promedio de todos los píxeles individuales en la plantilla, mientras que el error de alineación máximo cuantifica el error de alineación del píxel peor alineado de la plantilla.

Si es válido (el clasificador detecta que pertenece a la clase correspondiente), el componente es adecuadamente marcado en la imagen.

La imagen se recorre buscando siluetas que hagan *match*. Se han revisado dos métodos sencillos para recorrer la imagen:

- Haciendo un recorrido exhaustivo. Si una imagen de tamaño  $X \times Y$  con una plantilla de tamaño  $N \times M$ , primero se intenta hacer *match* a la ventana definida por el rectángulo  $(0, 0, N, M)$ ; después al rectángulo definido por  $(1, 0, N + 1, M)$ , y así sucesivamente hasta que se alcanza el final de la imagen a esa escala.
- Usando muestreo aleatorio. En una imagen de tamaño  $X \times Y$  con una plantilla de tamaño  $N \times M$ , se selecciona un número de muestras proporcional al tamaño de la imagen. Este método de recorrido acelera el proceso con un sacrificio en la precisión.

Se aplicó validación cruzada de 5 piezas<sup>4</sup> del conjunto de entrenamiento y se reporta la media y desviación estándar de la precisión del clasificador en el Cuadro 3.1. Los resultados muestran una precisión relativamente alta precisión en este conjunto de datos tan complejo.

---

<sup>4</sup>5-piece cross-validation

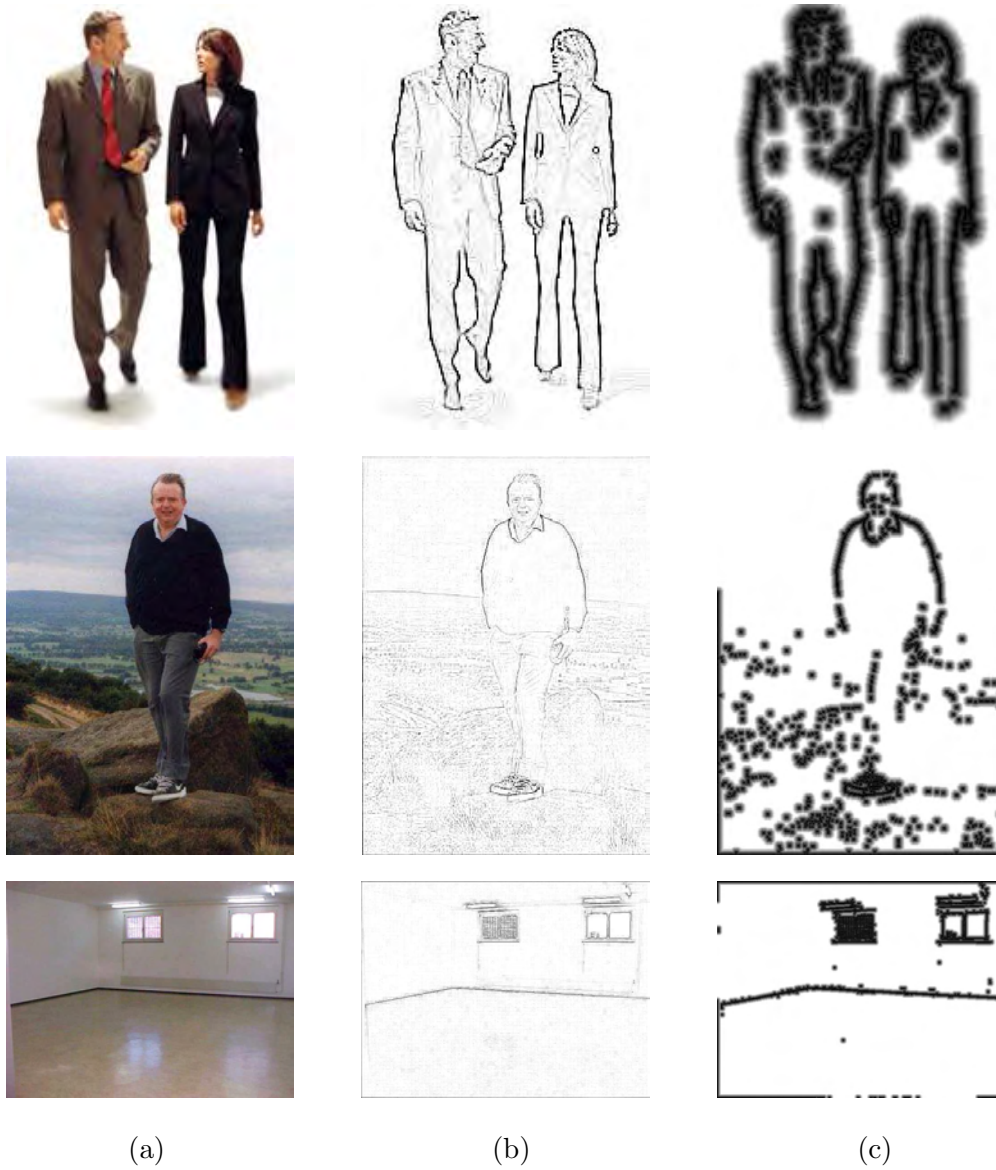


Figura 3.1: Ejemplo de la transformación de distancia. (a) es la imagen original, (b) es el resultado de la detección de contornos sobre la imagen original utilizando el método de Canny y (c) es el resultado de realizar una EDT sobre la imagen de contornos.

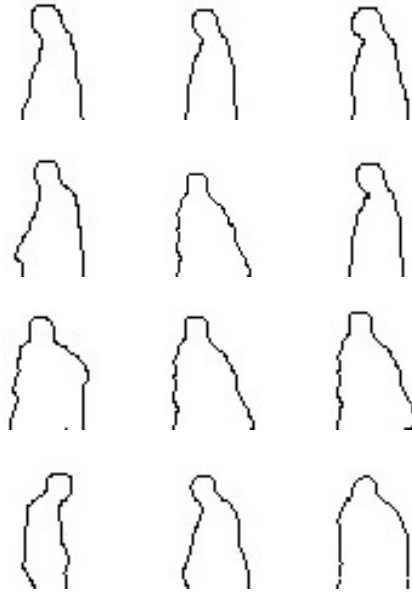


Figura 3.2: Las 12 plantillas utilizados por la versión inicial del sistema

Características	Media $\pm$ Desviación Est.
1x1	89 % $\pm$ 1 %
2x2	86 % $\pm$ 1 %
3x3	86 % $\pm$ 1 %
5x5	87 % $\pm$ 1 %
7x7	83.5 % $\pm$ 1 %

Cuadro 3.1: Selección de características vía tablero de ajedrez: selección hecha y media y desviación standard de la tasa de clasificación haciendo validación de 5-partes del clasificador final (de confirmación) de torsos.

En la Figura 3.5 (izquierda) puede observarse que con un kernel lineal y cuadrático se tiene un desempeño muy pobre en este dominio. Utilizando un kernel cuadrático homogéneo se puede aplicar el método de simplificación de Burges [10], como se reporta en Papageorgiou and Poggio [45] después de resultados reportados por Osuna et al. [42] en otro dominio. Se consideró que la pérdida de precisión hace este enfoque impráctico debido a que el área bajo la curva (AUC) era muy pequeño, el clasificador final era casi tan bueno como adivinar. Más adelante se hará un comentario sobre la relación entre el AUC de una ROC y U-estadísticos para evaluar el desempeño de un clasificador.

En la Figura 3.2 se muestran las plantillas utilizadas para la detección de torsos.

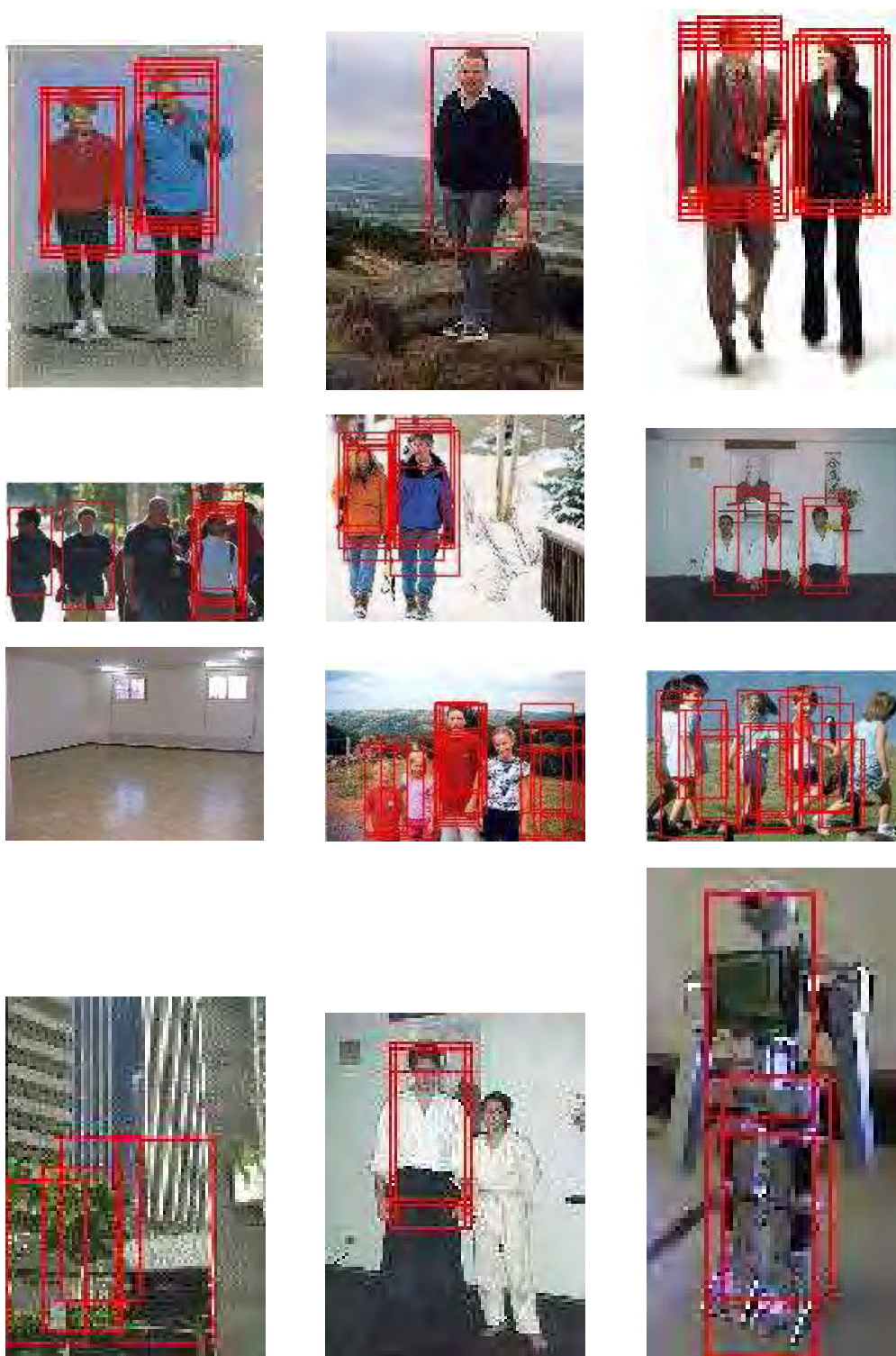


Figura 3.3: Ejemplos de la salida producida por el sistema sobre varias imágenes fuera de línea. Los rectángulos marcados sobre la imagen son los torsos detectados por el sistema.

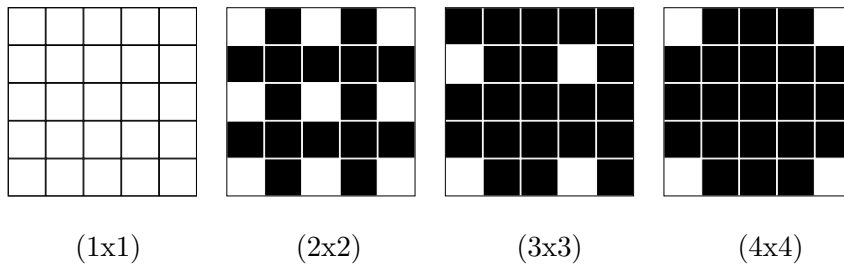


Figura 3.4: Selección de caraterísticas vía varios tamaños de tablero de ajedrez. Cuadrados blancos representan píxeles seleccionados y cuadrados negros representan píxeles no seleccionados.

### 3.2. Aplicación Robótica

El sistema funcionó con buena precisión en un conjunto de pruebas difícil, a pesar de la dificultad natural del dominio de interés. Fue obvio que la fase de clasificación era un cuello de botella inaceptable en cuanto a desempeño en tiempo, sobre todo para aplicaciones de tiempo real de naturaleza *embedded*. En la Figura 3.3 se muestran salidas de prueba sobre varias imágenes.

Se intentaron varios métodos para realizar la clasificación más rápidamente, obteniendo mejoras sustanciales; sin embargo, ninguno hacía factible un recorrido exhaustivo en tiempo real. Como una aplicación robótica generalmente corre en equipos computacionales lentos, se resolvió eliminar la fase de consulta al SVM debido a que era un cuello de botella que dañaba cualquier esfuerzo por tener un sistema de visión a tiempo real.

Finalmente se definió mejor la base de *templates* y se eliminó el clasificador y se obtuvieron resultados alentadores sólo utilizando *template matching*. Se implantó el sistema el mencionado robot ActivMedia Robotics Pioneer II. El procesamiento de imágenes y detección de torsos no se hace a bordo del robot sino en una máquina remota. Se logran procesar 4 imágenes por segundo en una máquina Pentium IV de 2.8 GHz.

Se probó el sistema desarrollado en un robot móvil ActivMedia Robotics Pioneer 2. La versión en línea (a bordo del robot) usa el método de revisión aleatoria previamente descrito.

Es importante resaltar el hecho de que la cámara no está estacionaria y que el fondo de la escena está constantemente variando, técnicas sencillas como resta de fondo no se puede utilizar para obtener los objetos del frente. Se tiene que hacer una revisión multiescala en cada cuadro.



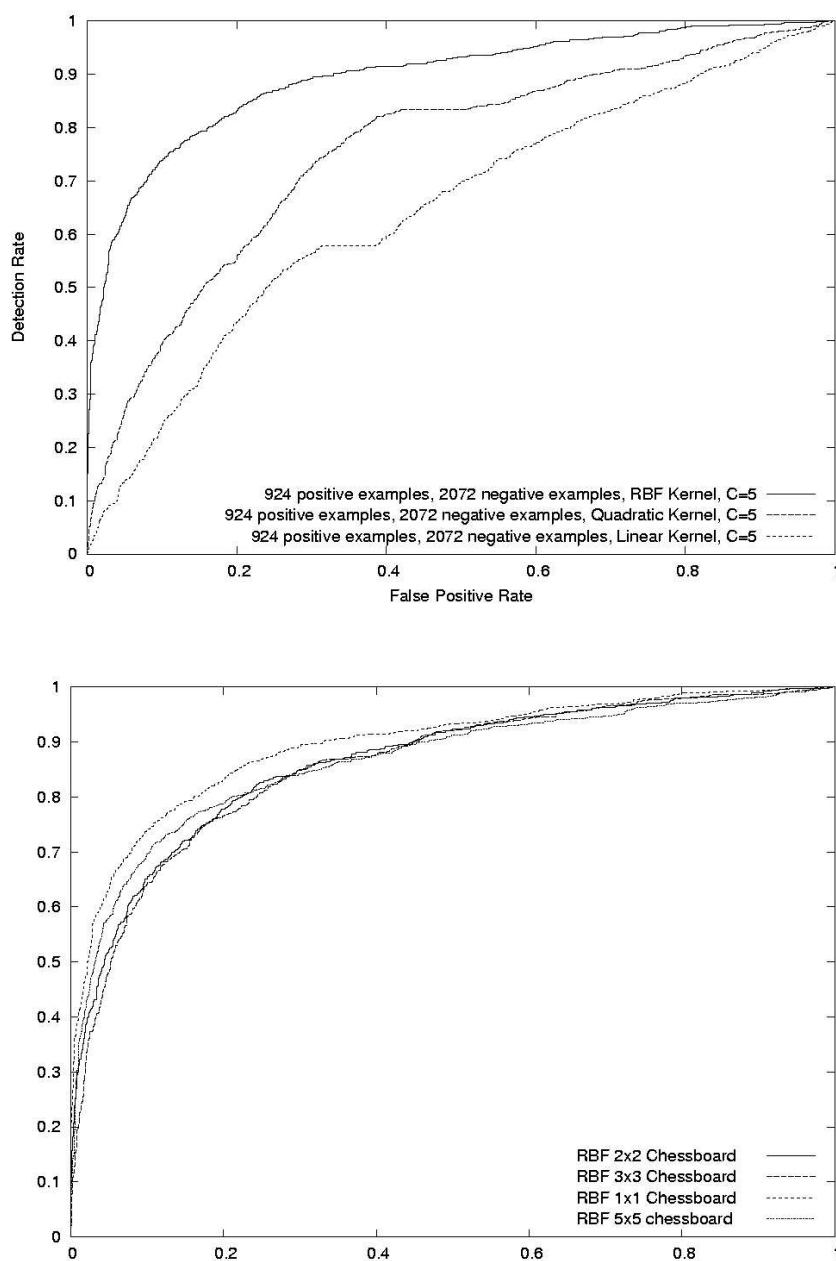


Figura 3.5: Curva ROC del Clasificador SVM. Arriba: Curva ROC del clasificador SVM con un kernel RBF (función de base radial) y  $C=5$ . Ambos clasificadores son de similar complejidad. Observar el pobre desempeño de los kernels lineales y cuadráticos. Abajo: Curva ROC del clasificador SVM con un kernel RBF y distintas selecciones de características vía tablero de ajedrez. La pérdida de precisión se puede observar en la Tabla 3.1.

Finalmente se integró *template matching*, detección de piel [16] y detección de esquinas (saliency) para poder funcionar en tiempo real. El enfoque de este trabajo es mostrar cómo estas técnicas integradas de detección visual de objetos es competitiva con otras técnicas de estado del arte [55] tanto en desempeño como en precisión.

La versión en línea del sistema funciona a 3 Hz.

### 3.3. *Bootstrapping* y Pruebas de Permutación

Una observación importante proviene de estudiar los valores de  $D(T, I)$  para varias imágenes. El valor del umbral  $\theta$  no puede ser constante, sino que tiene que depender de la imagen (de la ventana individual que se está analizando). En las Figuras 3.6 y 3.7 se muestran dos gráficos de contorno de los valores de  $D(T, I)$  para dos imágenes distintas.

Borgefors [5] y más recientemente Gavrilu et al [23] típicamente utilizan un  $\theta$  global para todos los cuadros analizados, pero en realidad en la ecuación 3.1,  $\theta$  depende de  $I$ . Así que una forma más general de la ecuación 3.1 es:

$$D(T, I) \leq \theta(I) \quad (3.4)$$

Este problema puede ser visto como una prueba de hipótesis:

- **H0:** No hay ajuste o alineación. El error de alineación  $\theta - D(T, I)$  es igual a cero
- **H1:** Hay ajuste o alineación. El error de alineación  $\theta - D(T, I)$  es mayor que cero

El método más común para hacer inferencia sobre medias en una sola muestra, pares asociados o dos muestras independientes son las pruebas  $t$ . Todos estos métodos suponen que los datos se distribuyen normalmente, por lo que no es aplicable en este caso (los datos ni siquiera son simétricos).

La inferencia estadística está basada en la distribución de muestra de los estadísticos. La idea fundamental del bootstrap es conseguir una distribución muestral, al menos aproximadamente,

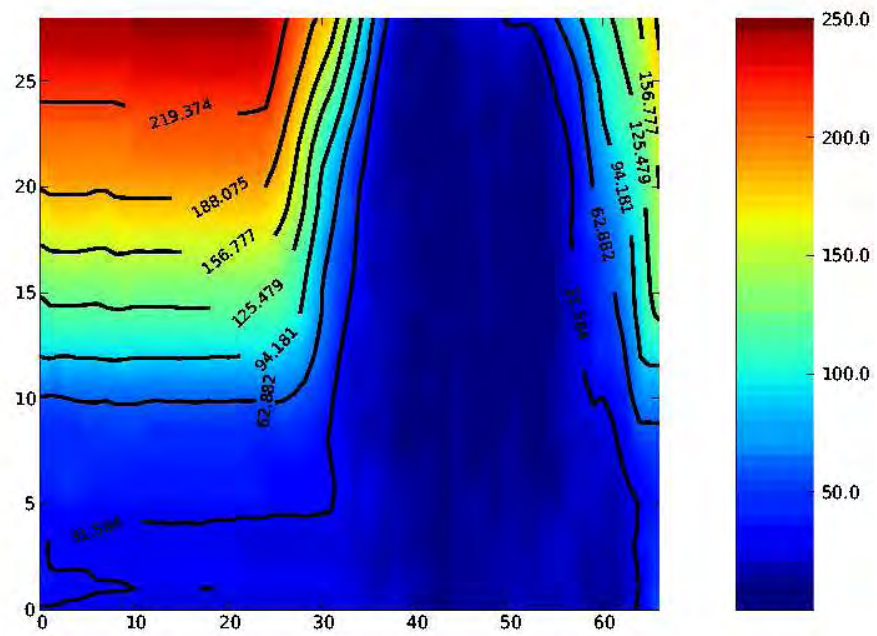


Figura 3.6: Valores de  $D(T, I)$  para una imagen. Arriba: Imagen de prueba. Abajo: gráfico de contorno de los valores de  $D(T, I)$ .

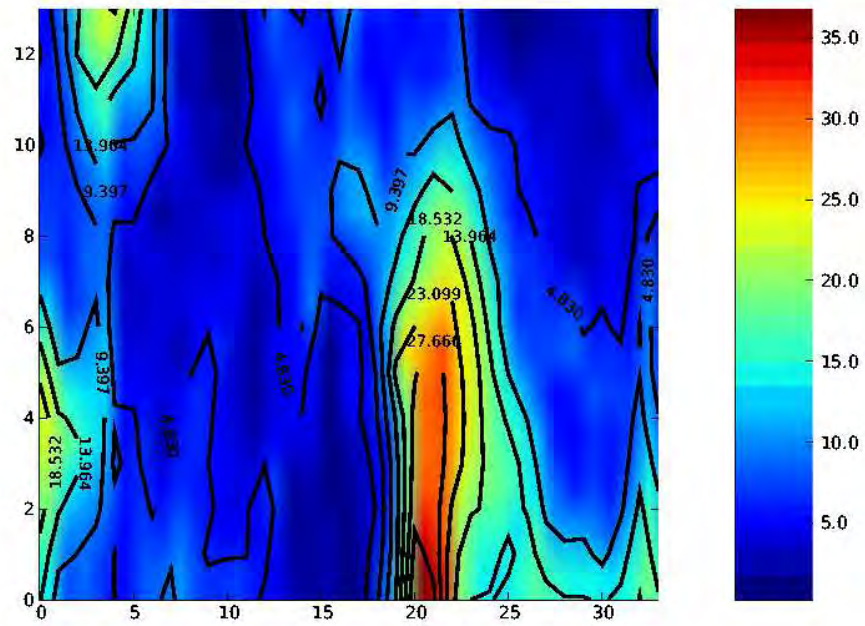


Figura 3.7: Valores de  $D(T, I)$  para una imagen. Arriba: Imagen de prueba. Abajo: gráfico de contorno de los valores de  $D(T, I)$ .

con una sola muestra<sup>5</sup>. Los demás aspectos de inferencia estadística se mantienen sin cambios (p-valor, valor crítico, etc.). Sólo que los nuevos métodos nos liberan de la necesidad de datos con distribución normal o de muestras grandes.

La idea del bootstrap puede crear la impresión que el remuestreo puede crear datos de la nada. Esto luce sospechoso. Pero las observaciones remuestreadas no son utilizadas como si fueran nuevos datos. La distribución bootstrap del remuestreo sólo se utiliza para estimar cómo la media muestral de una muestra grande variaría por efectos del muestreo. Usar los mismos datos para dos propósitos (estimar un parámetro y la variabilidad del estimado) es perfectamente válido. De hecho siempre se hace cuando se estima  $\mu$  utilizando  $\bar{x}$  y después se usa  $s/\sqrt{n}$  para calcular la variabilidad de  $\bar{x}$ .

El método desarrollado es el siguiente:

1. Consistente con la hipótesis nula y el diseño de muestreo. Si el par *template*-imagen fueran distintos ( $H_0$ ) y el efecto no estuviera presente, entonces si se cambiara aleatoriamente el *template*, no habría efecto.
2. Se aleatoriza el vector  $T$ ; esto es, se remuestrea con permutación  $n$  veces. Se calcula el nuevo  $D(T', I)$ , así se genera una distribución de error de alineación de las permutaciones (basándose en  $H_0$ ).
3. Se ubica el valor del estadístico observado (el  $D(T, I)$ ) original en la distribución construida. Un valor en la cola es poco probable que sea al azar.

Ahora el umbral de *matching* depende de la ventana donde hay que hacer *match*. Si el ruido hace buen *match* con la ventana, entonces la silueta tendrá que hacer *match* de una manera considerablemente mejor. Esta construcción produjo mejoras sustanciales al sistema desarrollado en áreas de la imagen donde el detector de bordes detectaba bordes fuertes en regiones muy cercanas (por ejemplo hojas en los árboles).

---

<sup>5</sup>El término bootstrap viene de la imagen imposible de: “pulling yourself up by your own bootstraps”

### 3.3.1. El Método de Monte Carlo para Bootstrap

Sea  $P_n$  un modelo estimado utilizando los datos  $X_1, \dots, X_n$ . El sesgo del bootstrap, varianza y estimadores de distribución son:

$$y = E_*[R_n(X_1^*, \dots, X_m^*, P_n)] \quad (3.5)$$

$$v_{\text{BOOT}} = \text{var}_*[R_n(X_1^*, \dots, X_m^*, P_n)] \quad (3.6)$$

y

$$H_{\text{BOOT}}(x) = P_*\{R_n(X_1^*, \dots, X_m^*, P_n) \leq x\} \quad (3.7)$$

donde  $\{X_1^*, \dots, X_m^*\}$  es una muestra de  $P_n$ ,  $R_n(\cdot)$  es un funcional adecuado y  $E_*$ ,  $\text{var}_*$  y  $P_*$  son la esperanza condicional, varianza y probabilidad, respectivamente, dados  $X_1, \dots, X_n$ . Para aplicar el método Monte Carlo sencillo para calcular los estimadores de bootstrap en las ecuaciones 3.5, 3.6 y 3.7 se comienza con la generación de  $B$  muestras independientes  $\{X_1^{*b}, \dots, X_m^{*b}\}$ ,  $b = 1, \dots, B$ , del modelo estimado  $P_n$ . Y luego se calcula  $R_n^{*b} = R_n(X_1^{*b}, \dots, X_m^{*b}, P_n)$  para  $b = 1, \dots, B$ , y aproximar  $b_{\text{BOOT}}$ ,  $v_{\text{BOOT}}$ , y  $H_{\text{BOOT}}(x)$  con:

$$b_{\text{BOOT}}^{(B)} = \frac{1}{B} \sum_{b=1}^B R_n^{*b} \quad (3.8)$$

$$v_{\text{BOOT}}^{(B)} = \frac{1}{B} \sum_{b=1}^B \left( R_n^{*b} - \frac{1}{B} \sum_{b=1}^B R_n^{*b} \right)^2 \quad (3.9)$$

y

$$H_{\text{BOOT}}^{(B)} = \frac{1}{B} \sum_{b=1}^B I\{R_n^{*b} \leq x\} \quad (3.10)$$

Este método es muy sencillo de programar, sólo hay que tomar muestras de  $P_n$  y calcular adecuadamente el valor de la variable aleatoria  $R_n$ . El cálculo del valor adecuado de  $B$  es un problema de interés en estadística aplicada, y afecta la precisión y el desempeño del sistema implementado. Si se fija un  $B$  muy pequeño, los estimadores no servirán de mucho. Mientras que si se fija muy grande, se perderá mucho tiempo de procesador y mucho procesamiento no ayudará a reducir el error del estimador de bootstrap original.

### 3.4. Implementación y Puesta en Marcha

Como parte de este trabajo se desarrollaron varios programas en C++ [53] que funcionan sobre OpenCV [6]:

- TTK: Template Toolkit es un sistema de visión a tiempo real (que funciona a bordo de plataformas móviles) que utiliza plantillas y clasificadores en cascada. Integra técnicas de bootstrapping basadas en Monte Carlo para calcular los umbrales para *template matching*. Este sistema se probó en dos dominios: torsos frontales de peatones y víctimas en situación de búsqueda y rescate. El sistema integra *matching* sobre la silueta de piel (información de color) y saliency (puntos de interés) para obtener resultados más precisos.
- TemplateExtractor: Un sistema para extracción semi-automática de plantillas de fotografías del objeto de interés. Integra la capacidad para clusterizar *templates* utilizando momentos de Hu [28, 47] y también técnicas de análisis de Procrustes [4, 32] <sup>6</sup>.
- CHLF: (Classifier using Haar Like Features) un sistema que utiliza características basadas en el wavelet de Haar similar al usado en Osuna et al [41] y Papageorgiu y Poggio [45] pero con clasificadores basados en *boosting* [22] tal como lo describen Viola y Jones [55], y en particular Kruppa et al [33]. La implementación hecha se basó en la implementación

---

<sup>6</sup>En la mitología griega, Procrustes es el hijo villano de Poseidón que robaba a viajeros en la vía de Euleis a Atenas. Él ofrecía a los viajeros una habitación para la noche y los hacía entrar en su cama estirándolos si eran muy “cortos” o cortándole las piernas si eran muy “largos”.

de Kruppa et al [33] y se desarrolló tanto para comparar con un sistema de estado del arte como para evaluar la factibilidad de integración con TTK.

De estos tres sistemas desarrollados, el primero será descrito en esta sección, los otros dos serán descritos en los siguientes capítulos.

OpenCV [6] es una librería de algoritmos de visión por computador de libre distribución; la librería es desarrollada por la comunidad de visión por computador y distribuida por Intel. Los prototipos preliminares de la versión final del sistema fueron desarrollados en Python, un lenguaje de programación dinámico muy sencillo con capacidades especiales para desarrollar prototipos utilizando PIL (Python Imaging Library, una librería abierta de procesamiento de imágenes para Python).

Las versiones que incluían un clasificador SVM utilizaban LIBSVM [15], una librería abierta para entrenar SVMs. LIBSVM tiene la peculiaridad de que no requiere software externo para resolver el problema de programación cuadrática que involucra el entrenamiento. LIBSVM está desarrollada en C++ y tiene enlaces para ser utilizado desde Python.

### 3.5. Conjuntos de Datos Utilizados y Procesamiento

Se utilizó el dataset de peatones de MIT CBCL [38, 39, 43, 44, 46] para los peatones. Este conjunto de datos incluye 900 imágenes de cuerpo completo de peatones (en poses frontales y reversas). Las fotografías fueron tomadas a personas caminando en la calle (por lo que los fondos suelen estar “atiborrados”). Las imágenes son de 128x64 píxeles (un total de 8192 píxeles). Los peatones suelen estar centrados en la imagen, pero esto no es buena guía debido a que las características intrínsecas de la persona empiezan a jugar un rol importante, tales como talla y textura.

Para la aplicación que utiliza SVM estos datos fueron cortados (para obtener solo el torso) convertidos a escala de grises e imagen de bordes.



Para la aplicación que utiliza ajuste de plantillas, estos torsos se convirtieron manualmente en una imagen de dato/no-dato del borde del torso. Al final (debido a problemas con la calidad de la imagen) sólo se pudieron obtener alrededor de 50 torsos. Muchos de estos torsos eran muy similares entre sí.

Para escoger un conjunto de plantillas se sabe que el tiempo de ejecución se degrada proporcionalmente al número de plantillas utilizadas. Sin embargo si se usan muy pocas plantillas se tiene una descripción muy mala del objeto de interés. Para esto se escogió un punto de corte en una curva de saturación, tomando en cuenta que los dos factores predominantes son número de plantillas utilizadas contra tiempo de ejecución con ese número de plantillas.

Para el caso de las víctimas se tomaron fotos del objeto de interés. Estas fotos incluían una variedad de poses mayor que la del caso de los peatones (en el que habían 2 poses, de frente y de espalda). En el caso de las víctimas habían 8 tipos de poses (de frente, de espalda, boca abajo y boca arriba de un lado y del otro ( $4 * 2$ )). Estas fotos eran de 320x240. En este caso era infactible el enfoque de clasificación a gran escala (SVM). Por lo que se procedió directamente a extraer las plantillas (mediante extracción de borde y aplicación de umbrales) de las víctimas y se procedió otra vez a estudiar una curva de saturación, con la nueva complejidad de un conjunto de datos gigantesco. También se evaluó la extracción de plantillas hecha de manera automática (que se describe en el capítulo siguiente) en este conjunto de datos y se procedió a comparar con la extracción manual que se hizo.

La curva de saturación muestra como a partir de un punto no se logran mejoras sustanciales agregando nuevas plantillas. Estas fotos tomadas se pusieron a disposición del público (tanto en forma cruda como preprocesada).

En el siguiente capítulo hará una descripción de las extensiones que fueron necesarias a este método para poder funcionar con el conjunto de datos de víctimas.

### 3.5.1. TTK: Sistema de Detección Implementado

El algoritmo implementado hace un recorrido exhaustivo a múltiples escalas. En cada escala evalúa ventanas de tamaño fijo de la imagen resultante de la transformación de distancia. Estas ventanas de tamaño fijo se compara con uno o varios *templates* de ese tamaño.

En la Figura 3.8 se presenta una simplificación del código del algoritmo de detección de objetos implementado. El algoritmo calcula utilizando bootstrap el límite adecuado sobre la condición de *matching*, ver donde dice `t < theta(dm_sample)`, para hacer al umbral de *matching* depender de cada ventana. Esto presenta una mejora sustancial sobre todo sobre escenas de tipo “cluttered”, donde el *template* se puede alinear bien con cualquier tipo de ruido<sup>7</sup>. Adicionalmente sobre la rutina de marcado se implementó un mecanismo de clusterización para eliminar ventanas marcadas repetidamente. Sólo en caso de que en varias pasadas se haya marcado una ventana se considerará sobre el marcado final.

Esta simplificación no presenta la integración de saliency (puntos de interés definidos a través de un criterio genérico) ni información de color (en el caso actual, color de piel). Sin embargo, el *matching* (en esta información) ocurre de manera análoga a la presentada sobre la información de bordes.

La función objetivo se define sobre (en la versión final del sistema) sobre tres alineaciones simultáneas (borde RGB, borde de piel y puntos de interés). Es importante destacar que se necesita una función de ranqueo (objetivo) para poder construir una curva ROC sobre el funcionamiento global del sistema.

### 3.5.2. Detección de Piel

En la Figura 3.9 se presenta una simplificación del código del algoritmo de detección de piel utilizado. El método convierte la imagen de RGB a YCbCr y en esta codificación aplica umbrales sencillos.

---

<sup>7</sup>la mejora introducida hace que el umbral de alineación sea inalcanzablemente alto en esos casos, para que la diferencia estadística sea significativa

```

1 def detectar(im):
2     (x,y) = im.size
3     while (x>ANCHO and y>ALTO):
4         dm = distmap.generate(im2.copy())
5         for i in range(0,x-ANCHO,2):
6             for j in range(0,y-ALTO,2):
7                 dm_sample = []
8                 for ii in range(ANCHO):
9                     for jj in range(ALTO):
10                        dm_sample.append(dm.getpixel((i+ii,j+jj)))
11                    (t,idx) = corrlista(lista_templates,dm_sample)
12                    if t<theta(dm_sample):
13                        marcar(draw,i,j,escala,lista[idx])
14                    else:
15                        pass
16                x = int(0.8*x)
17                y = int(0.8*y)
18                escala = 0.8*escala
19                im2 = im2.resize((x,y))
20                im_orig = im_orig.resize((x,y))

```

Figura 3.8: Una simplificación del código del algoritmo de detección implementado. La función **corrlista** calcula el error de alineación.

Una versión previa del sistema presentada por Chang y Brando [7] utilizaba un SVM sobre componentes RGB normalizadas:

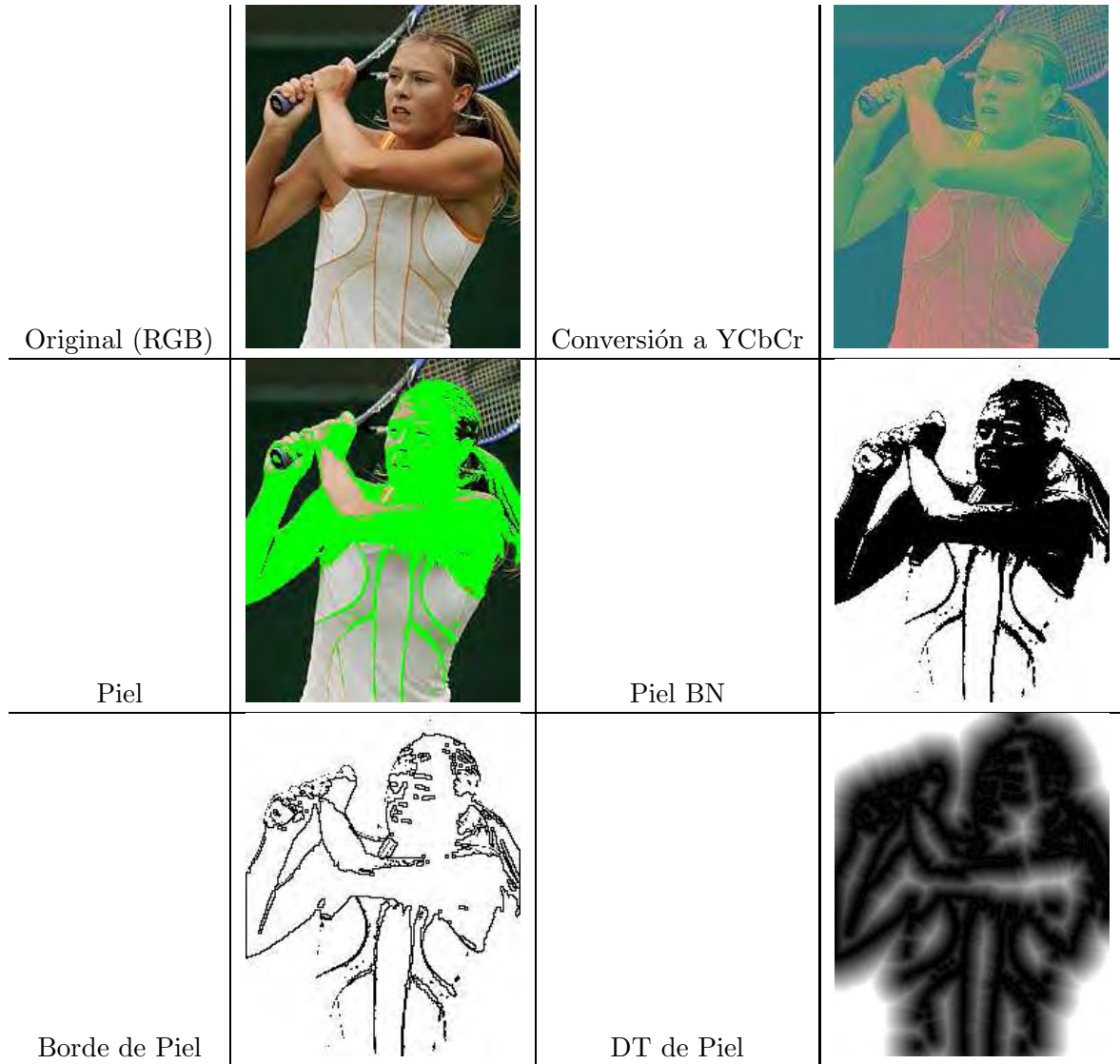
$$R_n = R/(R + G + B) \quad (3.11)$$

$$G_n = G/(R + G + B) \quad (3.12)$$

Pero la precisión no era buena, al menos para el ambiente de oficina donde se suele probar el robot y era particularmente lento (si se usaba un kernel cuadrático homegéneo se aplicaba el método de Burges, pero ese tipo de kernel no era buena elección). La precisión de la versión actual tampoco es buena, pero al menos es rápido, y es manualmente ajustable (característica importante cuando no se tiene buena precisión).

En el Cuadro 3.2 se muestra un ejemplo en el que el algoritmo de piel funciona relativamente bien. Sobre las regiones segmentadas por el algoritmo de detección de piel se obtienen los bordes y a estos bordes se les aplica una transformación de distancia sobre la cual se aplica el

proceso de *matching* anteriormente descrito. La inclusión de la información sobre la forma de la región segmentada de piel es una avance sobre la versión presentada en [7] que solo utilizaba la información de piel sin tomar en cuenta su configuración espacial.



Cuadro 3.2: Ejemplo de detección de piel. La imagen YCbCr tiene la banda Y en R, Cb en G y Cr en B.

### 3.5.3. Saliency, Esquinas y Tracking

Hasta este punto la detección del objeto de interés se hace exhaustivamente en cada cuadro. Esto es muy robusto porque no hace suposiciones sobre la dinámica cámara/escena. Pero lo

```

1 def skin(imin):
2     im = imin.copy()
3     imsal = imin.copy()
4     Rcr = [133, 173];
5     Rcb = [77, 127];
6     Th = [0, 7500];
7     xx, yy = im.size
8     meanr = mean2(imin,0)
9     meang = mean2(imin,1)
10    meanb = mean2(imin,2)
11    print meanr, meang, meanb
12    for i in range(xx):
13        for j in range(yy):
14            (r,g,b)= im.getpixel((i,j))
15            col = (1/3.0)*((r-meanr)**2+(g-meang)**2+(b-meanb)**2)
16
17            YYY = int(16 + 65.481 * r/255.0 + 128.553 * g/255.0 + 24.966 * b
18                    /255.0);
19            Cb = int(128 - 37.797 * r/255.0 - 74.203 * g/255.0 + 112 * b
20                    /255.0);
21            Cr = int(128 + 112 * r/255.0 - 93.786 * g/255.0 -18.214 * b
22                    /255.0);
23            im.putpixel((i,j),(YYY,Cr,Cb))
24
25            if (col>=Th[0] and col<=Th[1] and Cr>=Rcr[0] and Cr<=Rcr[1]):
26                imsal.putpixel((i,j),(0,255,0))
27
28    return imsal

```

Figura 3.9: Algoritmo detección de piel basado en YCbCr

que se está haciendo es suponer que el que la cámara aparece aleatoriamente capturando una imagen totalmente distinta, cuando en realidad se puede usar la continuidad del movimiento via un enfoque de tracking.

Para aplicar el esquema planteado de tracking basado en filtro de partículas se tiene que elegir una serie de puntos fáciles de hacerle seguimiento. Para ello se eligen las esquinas de la imagen por un criterio muy sencillo y rápido.

El criterio para elegir puntos de interés (regiones de interés) es que un punto de interés no está fuertemente correlacionado con sus vecinos. Perceptualmente se dice que estas regiones son *salient*. Hay muchos algoritmos para detectar puntos *salient*; en este trabajo se utilizó uno muy sencillo (y rápido) que se describe en esta sección:

Para detectar las esquinas se utiliza el operador de primeras derivadas de Canny [36] para calcular la primera derivada de la imagen. Se define una pequeña región de interés en la cual se van a detectar las esquinas: una matriz 2x2 de la suma de las derivadas se crea de la siguiente manera:

$$C = \begin{pmatrix} \sum D_x^2 & \sum D_x D_y \\ \sum D_x D_y & \sum D_y^2 \end{pmatrix} \quad (3.13)$$

Los autovalores son calculados resolviendo  $\det(C - \lambda I) = 0$ . Si  $\lambda_1 > t$  y  $\lambda_2 > t$ , donde  $t$  es un umbral constante, entonces en ese sitio hay una esquina (punto de interés).

#### 3.5.4. Puesta en Marcha en el Robot

El sistema corre a tiempo real sobre la salida de video de un robot ActivMedia Pioneer II. Debido a que el CPU que tiene el Pioneer II es un Pentium de 200 MHz se procedió a utilizar el procesador a bordo del robot únicamente para operar la funcionalidad de navegación y la captura de video. La detección de objetos corre en una máquina externa a la que el robot le transmite continuamente las imágenes capturadas. No se consideró que esto fuera un problema debido a que con un robot que tenga un procesador más rápido (tan rápido como el procesador externo que se usa, un Pentium IV de 2.8 GHz) se puede hacer detección objetos de una manera autónoma y autocontenida.

## Capítulo 4

# DETECCIÓN DE VÍCTIMAS EN SITUACIÓN DE BÚSQUEDA Y RESCATE

“Estamos especializados en una  
armoniosa repetición del desastre y la  
estupidez.”

---

Terenci Moix

En este capítulo se presenta una versión extendida y detallada del trabajo presentado por Castillo y Chang [14], en el cual se desarrolló un método que integra *template matching*, *saliency* [31] y piel para detectar víctimas en situaciones de búsqueda y rescate.

En esta parte del trabajo se procedió a verificar la factibilidad de utilizar este método, que ya se había mostrado efectivo en el caso de peatones, para detectar posibles víctimas en situaciones de búsqueda y rescate.

Se comenzó por obtener fotografías del objeto de interés y se diseñó un método para obtener automáticamente las plantillas de estas fotografías. El método consiste en filtrar los contornos de la fotografía y aplicarles un umbral para convertirlo en una imagen de dato/no-dato. La imagen resultante se puede ver como un grafo  $G$  donde los píxeles de dato (negros) son los nodos y los arcos son las vecindades inmediatas a otros píxeles de dato. En este grafo, se calculan las

componentes fuertemente conexas y se eliminan las que contengan menos de  $\tau$  nodos, siendo  $\tau$  un umbral experimentalmente fijado. Esto se hace para eliminar las componentes fuertemente conexas pequeñas que probablemente no pertenecen al objeto de interés.

## 4.1. Sistema de Extracción de *Templates*

El sistema de extracción de *templates* desarrollado, que se le dió el nombre TemplateExtractor utiliza las componentes fuertemente conexas filtradas para obtener *templates* del objeto de interés. El sistema tiene una limitante fuerte que es que todas las fotos originales tienen que estar relativamente centradas alrededor del mismo punto.

El sistema de extracción de *templates* requiere de información posicional del objeto de interés (lo que se llama *ground truth*). El sistema supone que le corresponde al especialista determinar que en la región de *ground truth* todos los bordes (para una configuración dada del algoritmo de extracción de bordes) son relevantes al objeto de interés. Corresponde al analista marcar los bordes que son *pedantic details* para su posterior eliminación. Por ello se considera que el sistema de detección visual de objetos que se presenta tiene un mecanismo de entrenamiento semiautomático.

### 4.1.1. Conglomerados y Momentos de Hu

El sistema provee varias facilidades para de un conjunto grande de imágenes rotuladas y de posibles *templates* a utilizar (no se pueden usar todos<sup>1</sup>) elegir un subconjunto suficientemente representativo (que capture suficientes poses) del objeto de interés. El sistema que se está presentando utiliza los momentos de Hu (con ciertas normalizaciones) como métrica de distancia entre dos *templates*.

La precisión de este método genérico de comparación es muy buena en el sentido que da agrupaciones (*clusters*, conglomerados) muy razonables. Sin embargo sigue siendo responsabilidad del

---

<sup>1</sup>el tiempo de detección por cuadros sería excesivo, sobre todo con las extensiones que se están planteando, p. ej. incrementalidad y bootstrapping



especialista elegir los *templates*, ya que el sistema sólo recomienda de acuerdo a su entendido de similitud.

Se implementó también un algoritmo de cálculo de similitud entre dos *templates* utilizando un algoritmo similar al presentado en Duta et al [19]. El enfoque de Duta et al es utilizar análisis de Procrustes para calcular la similitud entre dos figuras, en el caso presente los *templates*. El problema con este tipo de algoritmo es que para ser útiles necesitan tener adecuadamente el *landmarking* (selección de puntos de la figuras que son importantes, generalmente se usa algún criterio genérico como curvatura) y registro (asociación entre pares de *landmarks*).

Un momento espacial con orden en  $x$  de  $i$  y orden en  $y$  de  $j$  se define de la siguiente manera:

$$M_{i,j} = \sum_{x,y} I(x,y) x^i y^j \quad (4.1)$$

donde  $I(x,y)$  es la intensidad del píxel  $(x,y)$ . Un momento central con orden en  $x$  de  $i$  y orden en  $y$  de  $j$  se define como:

$$\mu_{i,j} = \sum_{x,y} I(x,y) (x - \bar{x})^i (y - \bar{y})^j \quad (4.2)$$

donde  $I(x,y)$  es la intensidad del píxel  $(x,y)$ . Y  $\bar{x}$  es la coordenada en  $x$  del centro de masa y  $\bar{y}$  es la coordenada en  $y$  del centro de masa, esto es:

$$\bar{x} = \frac{M_{1,0}}{M_{0,0}}, \quad \bar{y} = \frac{M_{0,1}}{M_{0,0}} \quad (4.3)$$

Un momento central normalizado con orden en  $x$  de  $i$  y orden en  $y$  de  $j$  se define como:

$$\eta_{i,j} = \frac{\mu_{i,j}}{M_{0,0}^{1+[(i+j)/2]}} \quad (4.4)$$

Los siete momentos de Hu se definen de la siguiente manera:

$$\begin{aligned}
h_1 &= \eta_{2,0} + \eta_{0,2} \\
h_2 &= (\eta_{2,0} + \eta_{0,2})^2 + 4\eta_{1,1}^2 \\
h_3 &= (\eta_{3,0} - 3\eta_{1,2})^2 + (3\eta_{2,1} - \eta_{0,3})^2 \\
h_4 &= (\eta_{3,0} + \eta_{1,2})^2 + (3\eta_{2,1} + \eta_{0,3})^2 \\
h_5 &= (\eta_{3,0} + 3\eta_{1,2})(\eta_{3,0} + \eta_{1,2})[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2] \\
&\quad + (3\eta_{2,1} - \eta_{0,3})(\eta_{2,1} + \eta_{0,3})[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2] \\
h_6 &= (\eta_{2,0} - \eta_{0,2})[(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2] + 4\eta_{1,1}(\eta_{3,0} + \eta_{1,2})(\eta_{2,1} + \eta_{0,3}) \\
h_7 &= (3\eta_{2,1} - \eta_{0,3})(\eta_{2,1} + \eta_{0,3})[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2] \\
&\quad - (\eta_{3,0} - 3\eta_{1,2})(\eta_{2,1} + \eta_{0,3})[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2]
\end{aligned}$$

Está demostrado que estos valores [28–30,47] son invariantes ante transformaciones de escala, rotación y reflexión, excepto el séptimo cuyo signo cambia como efecto de la reflexión.

Entonces la forma de calcular la distancia entre dos plantillas A y B es:

$$d(A, B) = \sum_{i=1}^7 | -f(m_i^A) + f(m_i^B) | \quad (4.5)$$

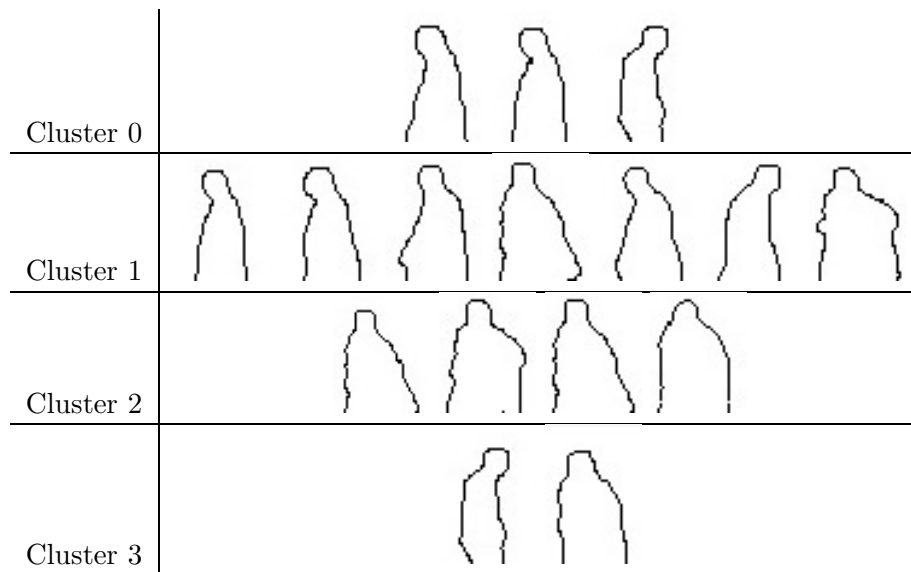
donde:

$$m_i^A = \text{sgn}(h_i^A) \log|h_i^A|, \quad m_i^B = \text{sgn}(h_i^B) \log|h_i^B| \quad (4.6)$$

y  $f(x)$  puede ser  $x$  o  $1/x$ .

Los resultados de estas agrupaciones (clustering) para el caso de los *templates* de peatones se presenta en el Cuadro 4.1. Al observar este cuadro, hay que notar que 6 de los 7 momentos que describen una figura son invariantes a transformaciones de reflexión simétrica, por lo que en el mismo cluster quedan figuras de peatones en ambos perfiles. Esto se podría resolver fácilmente dándole más peso al último invariante (cuyo signo cambia como efecto de reflexión simétrica),

pero aquí ese no es el objetivo, lo que se quiere hacer es evaluar la amplitud de las poses que puede capturar un conjunto de *templates* escogidos para hacer el *matching* a tiempo real.



Cuadro 4.1: Clusterización de los *templates* de peatones

## 4.2. Sistema de Detección de Víctimas




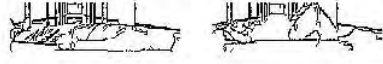
Se configuró el sistema de detección con los *templates* extraídos. Mediante experimentación se volvieron a fijar los umbrales para alineación sobre los bordes de la imagen RGB y los bordes sobre la imagen de piel.

Un problema que se hizo obvio inmediatamente después fue que los *templates* de las víctimas tienen muchos (cinco veces) más puntos de datos que los *templates* de los torsos de peatones.

En la Figura 4.1 se presenta un diagrama del procesamiento para la detección de víctimas.

Un paso adicional que se agregó fue un paso de clusterización al final de la fase de detección (revisión a múltiples escalas).

La Figura 4.2 muestra un ejemplo del procesamiento necesario para obtener las plantillas de una fotografía. La Figura 4.3 muestra el funcionamiento del sistema alrededor de escombros.

Cluster 0	
Cluster 1	
Cluster 2	
Cluster 3	

Cuadro 4.2: Clusterización de los *templates* de víctimas

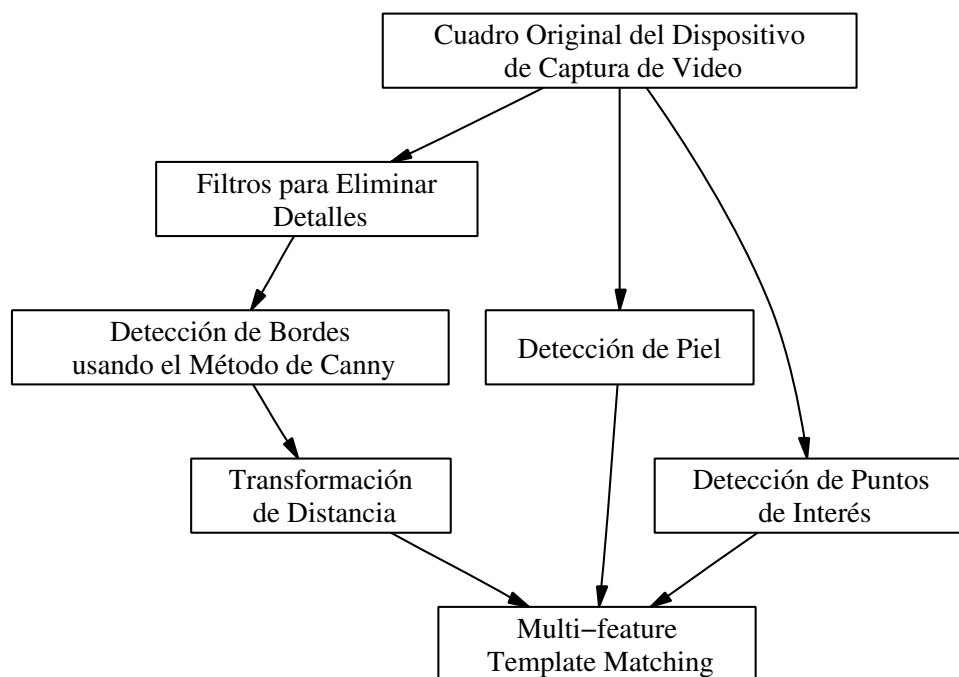


Figura 4.1: Diagrama del sistema de funcionamiento del sistema de detección de víctimas.



Figura 4.2: Ejemplos de extracción de templates: (a) es la imagen original, (b) es la imagen de contornos en escala de grises y (c) es una imagen de la componente fuertemente conexas más grande del grafo de vecindad resultante de la aplicación de un umbral de (b).

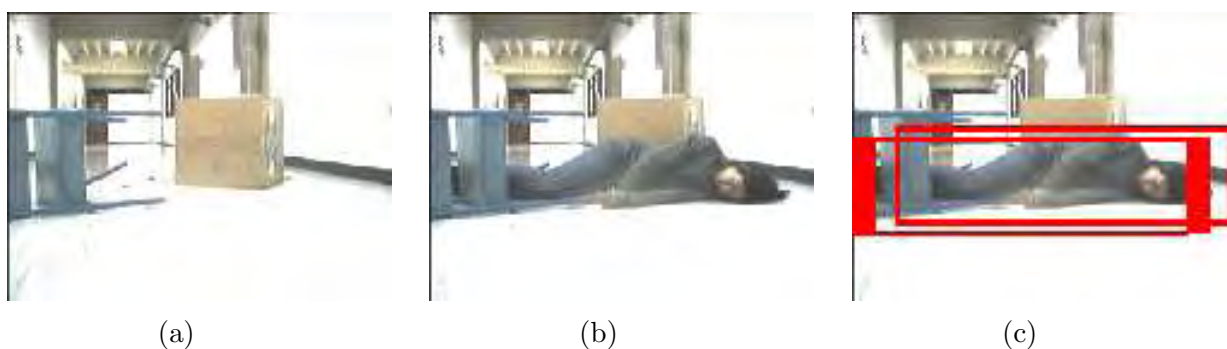


Figura 4.3: Detección de una víctima alrededor de escombros. (a) escombros, (b) víctima ocluida por escombros y (c) el sistema detectando una víctima alrededor de escombros.

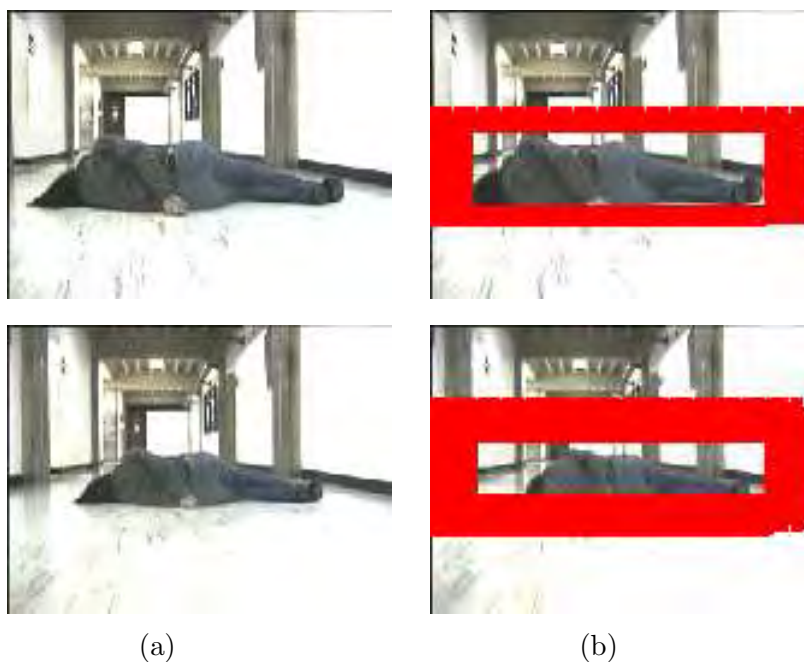


Figura 4.4: Detección de víctimas a varias distancias. (a) imagen original, arriba: detección más cercana, abajo: detección más lejana, (b) imagen marcada

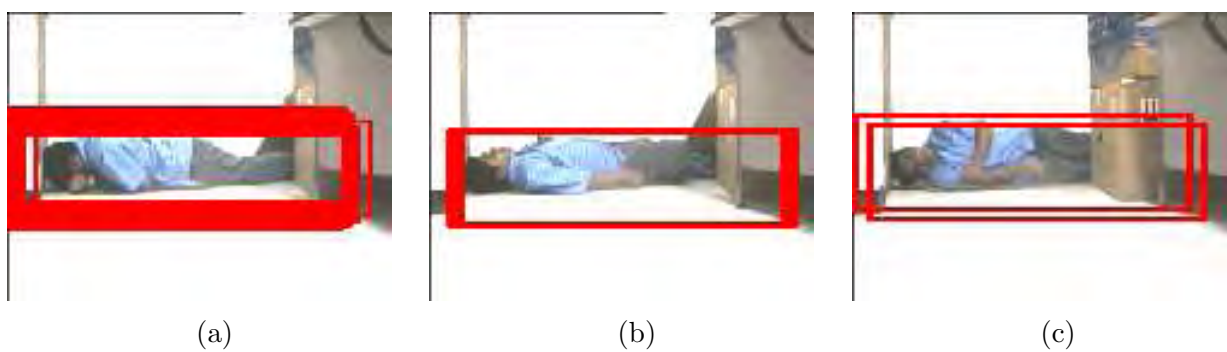


Figura 4.5: Resistencia a oclusión alrededor de características parciales no críticas. (a) una vista no muy ocluida, (b) una vista moderadamente ocluida y (c) una vista muy ocluida.

## Capítulo 5

# EXPERIMENTACIÓN Y RESULTADOS

“Son vanas y están plagadas de errores  
las ciencias que no han nacido de la  
experimentación, madre de toda  
certidumbre.”

---

Leonardo da Vinci

En esta sección se presenta la experimentación hecha y los resultados obtenidos. Se presenta a continuación los experimentos hechos con cada uno de los dos casos de estudio.

### 5.1. Experimentos con el Sistema de Detección de Peatones

Para hacer la detección de peatones se configuró el sistema TTK desarrollado con *templates* de 12 poses de peatones mostrado anteriormente. Por motivos de completitud experimental se utilizó una revisión exhaustiva (en lugar de una revisión aleatorizada).

#### 5.1.1. Comparación con el Método de Viola y Jones

El método de Viola y Jones representa una familia de métodos similares a los presentados por Oren et al [41] y Papageorgiu y Poggio [45].

Se comparó con el método de Viola y Jones [55] de detección de objetos a tiempo real. La implementación del sistema de Viola y Jones se hizo a través de un derivado de la implementación de Kruppa et al [33]. Se rotularon (*ground truth*) 32 imágenes de variados niveles de dificultad con peatones para tener un *ground-truth* y con estas imágenes como verdad absoluta se evaluaron ambos sistemas. En las 32 imágenes había un total de 79 peatones y un sin fin de no-peatones (86036).

### Descripción del Método de Viola y Jones

El método de Viola y Jones [55] es considerado uno de los métodos más rápidos y precisos de detección visual de objetos.

El sistema de detección de Viola y Jones utiliza características sencillas en lugar del valor de los píxeles. Estas características sencillas codifican conocimiento del dominio que es difícil de aprender utilizando datos de entrenamiento. Las características sencillas también presentan el beneficio de desempeño en tiempo. Las características sencillas utilizadas son similares al wavelet de Haar utilizado por Papageorgiou et al [45]. El valor de una característica de dos rectángulos es la diferencia de la suma de los píxeles en las regiones rectangulares. Asimismo, el valor de una característica de tres rectángulos es la suma de los valores de píxel de los dos rectángulos externos menos el valor de la suma de los rectángulos internos. Adicionalmente, el valor de una característica de cuatro rectángulos es la diferencia de los pares de rectángulos diagonales. No es trivial incorporar el uso de color en este método.

Viola y Jones definen una imagen integral que les ayuda a calcular las características rectangulares. La imagen integral en la posición  $x, y$  contiene la suma de los píxeles arriba y a la izquierda inclusive<sup>1</sup>.

Viola y Jones proponen un algoritmo basado en AdaBoost [22] para elegir las características a utilizar. Con una cascada de clasificadores débiles (un clasificador débil clasifica un patrón correctamente al menos 51 % de las veces). Esta cascada es similar a un árbol de decisión (via un

---

<sup>1</sup>El trabajo de Viola y Jones define una recurrencia para calcular la imagen integral en un sólo pase por la imagen.



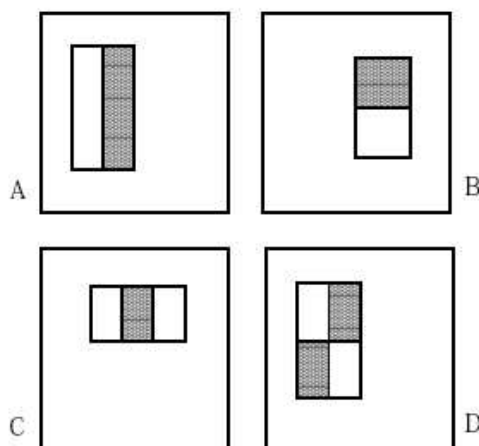


Figura 5.1: Las cuatro características de estilo de Haar utilizadas por el sistema de Viola y Jones

algoritmo similar a ID3) degenerado. El utilizar varios clasificadores en cascada es una idea que se observó inicialmente en el sistema de Rowley-Baluja-Kanade [50]. El uso que le dieron Viola y Jones a AdaBoost es similar a la construcción de un árbol de clasificación.

El método de Viola y Jones presenta sin embargo ciertas deficiencias que lo hacen poco competitivo para detección de torsos (y objetos semirígidos en general). Una de ellas, que deriva de su propia formulación, es que al usar sus características rectangulares (similares a la base de Haar) la captura de la silueta no es directa. Para objetos rígidos (como caras) el método es muy rápido y preciso, pero cuando el objeto es semirígido y puede adquirir varias poses, la precisión degrada significativamente<sup>2</sup>.

## Resultados de los Experimentos

Cuando se revisa exhaustivamente una imagen a múltiples escalas, es importante notar que el clasificador llamado “no”<sup>3</sup> es relativamente preciso. Por lo que la exactitud global dada por el

<sup>2</sup>Observar que se está comparando con un sistema de detección general pero no en su mejor condición.

<sup>3</sup>El clasificador que para cualquier dato lo clasifica automáticamente como cero

estadístico:

$$\frac{VP + VN}{VP + FP + VN + FN} \quad (5.1)$$

no sirve de mucho.

Se dará particular importancia a sensibilidad (probabilidad de que un torso sea clasificado como tal) y especificidad (probabilidad de que un no torso sea clasificado como tal). Esto es:

$$\frac{VP}{VP + FP} \quad (5.2)$$

$$\frac{VN}{VN + FN} \quad (5.3)$$

	Predicción Positiva	Predicción Negativa
Ejemplos Positivos	VP	FN
Ejemplos Negativos	FP	VN

Cuadro 5.1: Ejemplo de una tabla de contingencia o matriz de confusión

	Predicción Positiva	Predicción Negativa
Ejemplos Positivos	30	49
Ejemplos Negativos	34	86002

Cuadro 5.2: Tabla de contingencia para TTK sobre los ejemplos rotulados de prueba

	Predicción Positiva	Predicción Negativa
Ejemplos Positivos	7	72
Ejemplos Negativos	26	86010

Cuadro 5.3: Tabla de contingencia para CHLF sobre los ejemplos rotulados de prueba

En las figuras 5.2 y 5.3 se muestran las gráficas ROC de TTK y CHLF para el conjunto de imágenes rotuladas a mano (de prueba) con su respectivo AUC (Area Under the Curve). En los siguientes párrafos se discutirá la fundamentación teórica de una ROC sobre el ranqueo de valor de decisión del clasificador.

U-estadísticos de dos muestras: sea  $h(x_1, \dots, x_r, y_1, \dots, y_s)$  conocida, simétrica sobre  $x_1, \dots, x_r$  y  $y_1, \dots, y_s$  entonces:

$$U = \frac{1}{\binom{m}{r}\binom{n}{s}} \sum_{\alpha} \sum_{\beta} h(x_{\alpha_1}, \dots, x_{\alpha_r}, y_{\beta_1}, \dots, y_{\beta_s}) \quad (5.4)$$

claramente  $U$  es un estimador insesgado del parámetro:

$$\theta = Eh(X_1, \dots, X_r, Y_1, \dots, Y_s) \quad (5.5)$$

Un caso particular de un U-estadístico es el estadístico de Mann-Whitney<sup>4</sup>: se desea estimar el parámetro  $P[X \leq Y]$  y  $h(x, y) = 1\{x \leq y\}$  y U-estadístico es:

$$U = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n 1\{x_j \leq y_i\} \quad (5.6)$$

Es demostrable que el AUC es equivalente al estadístico de Mann-Whitney [37] (MW). La prueba de MW es útil para determinar si una de dos variables aleatorias es estocásticamente mayor que la otra. Esto es:

$$P[X \leq Y] \quad (5.7)$$

Si se elige  $X$  como los ejemplos de una clase y  $Y$  los ejemplos de la otra clase, y los valores de  $X$  y  $Y$  como el *score* estimado por el clasificador, se tiene que AUC es equivalente al estadístico de MW.

La AUC realmente estima la probabilidad de que si se elige un ejemplo de la clase 1 y un ejemplo de la clase cero, el clasificador le dará un mayor score al primero que al segundo.

Es importante notar que esto no significa que va a clasificar bien ambos ejemplos. Pero sí significa que:

---

<sup>4</sup>Y el estadístico de Wilcoxon que está muy relacionado.

- Existe un umbral para el que va a clasificar ambos correctamente.
- No puede clasificar ambos incorrectamente, cualquiera que sea el umbral.

El AUC para TTK es de 0.74 mientras que el AUC para CHLF es de 0.59.

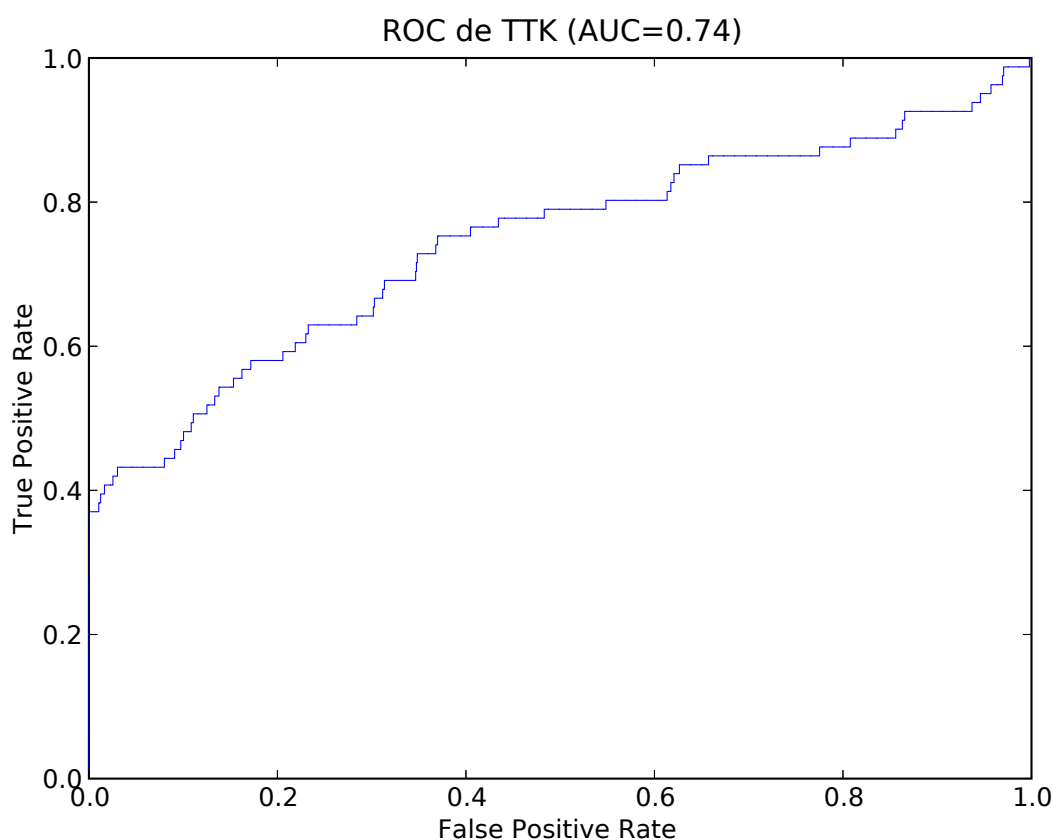


Figura 5.2: Curva ROC del sistema TTK sobre las imágenes rotuladas.

### 5.1.2. A Bordo del Robot

Se probó el sistema a bordo del robot (un robot móvil ActivMedia Robotics Pioneer 2). La versión en línea (a bordo del robot) utilizaba un método de revisión exhaustiva descrito previamente.

Es importante notar que debido a que la cámara no está estacionaria y el fondo está continuamente cambiando, técnicas simples como resta de fondo no se pueden utilizar para obtener

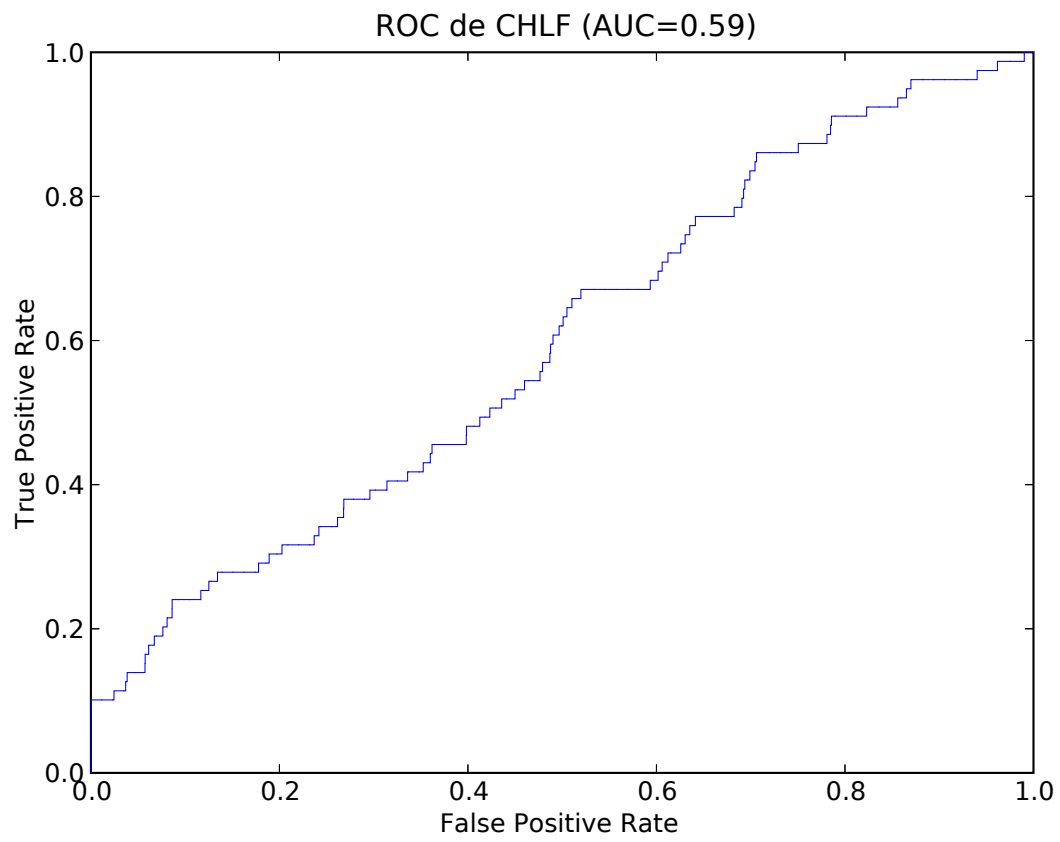


Figura 5.3: Curva ROC del sistema CHLF sobre las imágenes rotuladas.

objetos del frente. El enfoque aquí presentado ejecuta un método de revisión exhaustiva a cada cuadro sobre toda la imagen capturada.

Debido a que una aplicación robótica generalmente tiene que correr en equipos computacionales que no son de última generación, se consiguió que consultar a un SVM se convierte en un cuello de botella inaceptable. Para todas las versiones en línea (a bordo de plataformas móviles) se eliminó el SVM utilizando *template matching* sobre múltiples características con umbrales de detección calculados vía bootstrapping estadístico.

Si bien el desempeño en cuanto a la precisión del clasificador disminuyó un poco, este cambio hizo posible implementar una aplicación robótica a tiempo real. Los efectos de “clutter” se ven mejorados utilizando bootstrapping.

Dependiendo del tamaño de la imagen de entrada y si se usa tracking de features, el sistema puede llegar a funcionar a 3Hz.

## 5.2. Experimentos con el Sistema de Detección de Víctimas

Se configuró TTK para que funcione con el conjunto de *templates* obtenidos de la colección de fotos. Se escogió (manualmente, tras una inspección visual por un experto) un subconjunto de los *templates* obtenidos que representara adecuadamente la variedad de poses en que se puede conseguir a la víctima.

Para el caso del sistema de detección de víctimas no se conoce de ningún sistema de visión diseñado explícitamente para este tipo de actividades. Se intentó entrenar CHLF (la implementación hecha del sistema de Viola y Jones) para que funcionara con el conjunto de datos de víctimas pero no se pudo. El sistema de Viola y Jones no sirvió en 96 horas de entrenamiento. El conjunto de datos de víctimas es particularmente masivo hasta en estándares de conjuntos de datos multimedia (son alrededor 800 imágenes a color de 160x120 píxeles).

### 5.2.1. El Sistema sobre las Fotos Obtenidas

Se evaluó el desempeño de TTK sobre las fotos obtenidas de las víctimas. Las 800 imágenes fueron manualmente rotuladas (todas estaban razonablemente alineadas debido a que el objeto de interés estaba cerca al centro de la imagen).

El primer experimento trataba de generar una escena artificial que pareciera una víctima pero que en realidad no lo fuera. De aquí se obtuvo un resultado interesante: que es difícil tratar de engañar al sistema generando una escena artificial en el ambiente de oficina en que se hizo la experimentación.

Se experimentó con oclusión sistemática de las víctimas alrededor de escombros. El sistema se mostró relativamente robusto a la oclusión.

Un problema que se detectó en esta experimentación fue que en ocasiones el sistema no detectaba a víctimas cuando estaban presentes. En esta situación se descubrió que la revisión exhaustiva a múltiples escalas era muy sensible a la contextura (o talla) de la víctima en la escena. A este problema se le encontraron dos soluciones distintas: revisar la imagen a más escalas o incluir en los *templates* plantillas de personas con una variedad de contexturas. Desafortunadamente ambas soluciones implican mayor tiempo de procesamiento por cuadro.

Finalmente se evaluó el efecto de la disponibilidad de piel. La piel se utilizó como una característica adicional (no limitante). Este tipo de enfoque oportunista que no sólo toma en cuenta el color sino que la posición y la forma funcionó muy bien en este dominio.

### 5.2.2. El Sistema a Bordo del Robot

En el caso en el que el sistema funcionó a bordo del robot, se midió el alcance de la capacidad de detección de las víctimas cuando el sistema hacía una detección correcta.

Para este caso de prueba, las condiciones de iluminación, pose y escenario de rescate eran todas realistas. El robot era operado remotamente por una persona que conocía en detalle el funcionamiento del sistema.

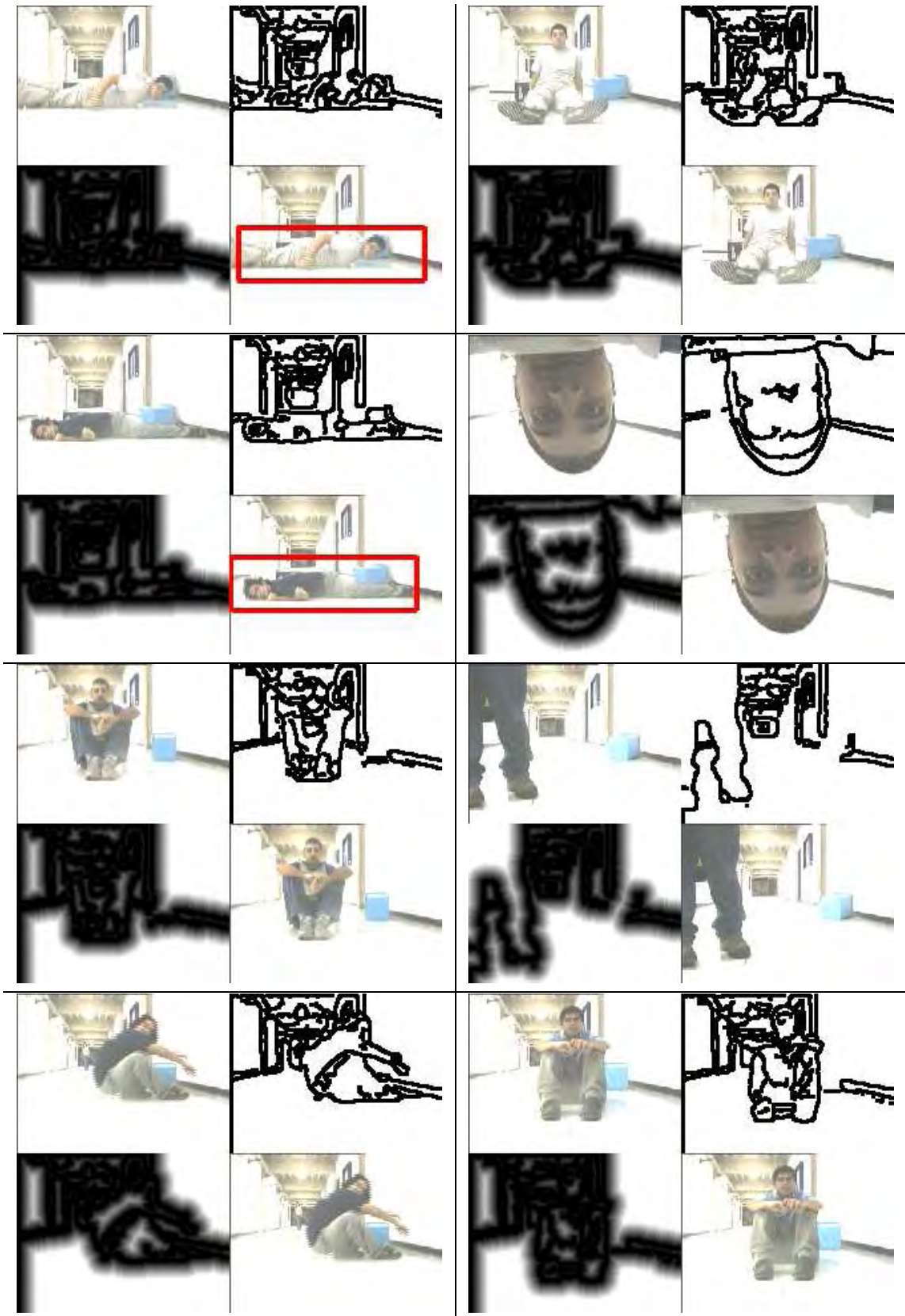
Se ajustó la claridad de la entrada de video para que los detalles (puntos) del piso no generaran bordes.

Con una vista no ocluida el sistema detecta una víctima desde entre 2.0m y 4.5m. Alrededor de oclusión leve el rango se reduce a entre 2.0m y 3.3m. Y a medida que la oclusión aumenta el rango se reduce aún más, hasta que se imposibilita hacer la detección correctamente.





Cuadro 5.4: El sistema funcionando a bordo del robot, en un pasillo en un ambiente de oficinas.



Cuadro 5.5: El sistema funcionando a bordo del robot, en un pasillo en un ambiente de oficinas.

## Capítulo 6

# CONCLUSIONES

”Mil cosas avanzan. Novecientas noventa y nueve retroceden. Esto es el progreso”.

---

Henri Frédéric Amiel (1821-1881);  
escritor suizo.

Se ha estudiado el uso de la transformación de distancia para hacer ajuste de plantillas en aplicaciones de visión por computador a tiempo real a bordo de plataformas móviles.

Se ha presentado un método para detección de objetos semirígidos en poses varias para aplicaciones robóticas. Si bien la fundamentación del método no es reciente, se han incorporado formas de modernizarlo y hacerlo más rápido y preciso. Entre estas modernizaciones están:

- **Múltiples características:** el método incluye información de color y *saliency*, además de forma.
- **Bootstrap:** en un punto clave del *template matching* se evalúa si la similitud de la imagen a un *template*  $D(T, I)$  es menor que un umbral  $\theta$ . Hasta ahora se había usado un  $\theta$  arbitrario experimentalmente fijado. El enfoque aquí desarrollado infiere el  $\theta$  adecuado (que es distinto de ventana a ventana) utilizando técnicas de bootstrapping estadístico.

- **Incrementalidad:** el proceso de *matching* incluye evaluar todos los puntos del *template*; esto se hace incrementalmente justo hasta que sea necesario para aceptar o rechazar la hipótesis de *matching*.

Los problemas más difíciles con los métodos basados en *template matching* sobre transformación de distancia tienen que ver con resistencia a escenas donde existe “clutter”. Las modernizaciones presentadas en este trabajo atacan un problema significativo al cual se le ha dado mucha relevancia recientemente; para enfoques complementarios, ver por ejemplo Thayanant-han et al [54], donde se aplican restricciones de continuidad del *template* sobre la ventana de la imagen que se ajusta y se obtienen mejoras significativas utilizando *shape contexts*.

En el presente trabajo se propuso un método basado en componentes fuertemente conexas para obtener plantillas a partir de fotos de un objeto de interés. El método funcionó adecuadamente para el caso de obtención de plantillas de las fotos de las víctimas en situación de búsqueda y rescate. El método sirve cuando las figuras de los objetos son cerradas y están relativamente centradas en la imagen. El método no es enteramente automático. A largo plazo, el éxito de los métodos basados en *templates* va a estar condicionado a que existan herramientas para extraer templates automáticamente de fotografías indistintamente de su forma y características intrínsecas.

Los matemáticos definen una figura (*shape*) como una clase de equivalencia sobre un grupo de transformaciones (rotación, escala, afín). Esta definición no es completa desde el punto de vista de análisis visual. La definición sólo indica cuándo dos figuras son exactamente idénticas. Se requiere saber sobre similitud o distancia entre dos figuras. La definición estadística de figura (ver por ejemplo Bookstein [4] o Kendall [32]) ataca este problema pero supone que la correspondencia entre pares de puntos de dos figuras es conocida, cosa que tampoco es generalmente cierta. Si se tuviera la correspondencia, la situación sería ideal. El enfoque aquí planteado toma descriptores genéricos basados en momentos de Hu, sacrificando un poco de información detallada sobre la figura a cambio de tener un mecanismo sencillo y directo para su descripción.

Se mostró cómo evaluar qué variedad de poses se están contemplando para detección del objeto de interés. Este criterio de comparación (medida de distancia entre dos figuras) mostró ser

efectivo con los dos casos de prueba que se evaluaron. Estas agrupaciones se pueden utilizar efectivamente para calcular con cuál subconjunto de un conjunto grande de plantillas se debe hacer *template matching*.

A medida que avanzan los métodos basados en cálculo de correspondencias, se podrán utilizar para calcular la distancia entre dos figuras. Se podrá evaluar la variedad de poses utilizando los mismos algoritmos de conglomerados aquí propuestos con la nueva medida de distancia. Por supuesto, al trabajar directamente sobre la figura y correspondencias sobre pares de puntos, los conglomerados serán mucho más precisos.

Se mostró que el método propuesto se compara muy bien con otros métodos del estado del arte.

El enfoque tomado de basar el estudio en aplicaciones robóticas (una cámara en una plataforma móvil) permitió obtener conocimientos sobre las características necesarias que deben tener estos sistemas. Y la aplicabilidad de estos sistemas a las situaciones propuestas es inmediata, no así otras implementaciones sobre versiones de los varios problemas tan idealizadas que el software desarrollado era prácticamente inútil.

## 6.1. Direcciones Futuras

Tanto el sistema de detección de peatones como el sistema de detección de víctimas necesitan mejorar en cuanto precisión y desempeño para ser útiles en situaciones críticas de la vida real.

La extracción automática de *templates* de fotos del objeto de interés independiente de la forma que eventualmente tenga el *template* sigue siendo un problema no resuelto.

El estudio e implementación de métodos que calculen correspondencias entre dos conjuntos de puntos (para posteriormente calcular la distancia o similitud entre esos dos conjuntos de puntos) es un área de investigación prometedora. Con este tipo de construcciones se podrá generar nuevas figuras a partir de figuras existentes (via centroides), estas nuevas figuras (que capturan

variedades de poses) podrán ser utilizadas en la fase de matching, potencialmente aumentando la precisión. Hay algunos resultados preliminares en esta área [24], pero nada definitivo.

Recientemente ha habido varios avances en la aceleración de las técnicas de *template matching*. Aunque este método es muy eficiente en desempeño en tiempo, esta dirección también es muy prometedora y los avances que se están haciendo son interesantes, ver Schweitzer et al [51] y Fredriksson y Ukkonen [21].

El mecanismo de bootstrap presenta gran flexibilidad en cuanto a las pruebas estadísticas que se realizan a tiempo real. Esta vía para evaluar las condiciones de *matching* presenta gran promesa que apenas se ha comenzado a explorar.

# BIBLIOGRAFÍA

- [1] M. Bertozzi, A. Broggi, R. Chapuis, F. Chausse, A. Fascioli, and A. Tibaldi. Shape-based pedestrian detection and localization. *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems*, pages 328–333, 2003.
- [2] M. Bertozzi, A. Broggi, P. Grisleri, A. Tibaldi, and M. D. Rose. A tool for vision based pedestrian detection performance evaluation. *Procs. IEEE Intelligent Vehicles Symposium*, pages 784–789, 2004.
- [3] D. Beymer and K. Konolige. Detection and tracking of people using stereo and correlation. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume I, pages 87–93, Los Alamitos, CA, Sept. 20–27 1999.
- [4] F. L. Bookstein. *Morphometric tools for landmark data: geometry and biology*. Cambridge Univ. Press, 1991.
- [5] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.
- [6] G. Bradski. The OpenCV library. *j-DDJ*, 25(11):120, 122–125, Nov. 2000.
- [7] A. Brando and C. Chang. Firefighter-robot interaction during a hazardous materials incident exercise. In *11th International Conference on Advanced Robotics*, volume 2, pages 658–663, 2003.
- [8] A. Broggi, M. Bertozzi, R. Chapuis, F. Chausse, A. Fascioli, and A. Tibaldi. Pedestrian localization and tracking system with kalman filtering. *Procs. IEEE Intelligent Vehicles Symposium*, pages 584–589, 2004.
- [9] R. Brunelli and T. Poggio. Template matching: Matched spatial filters and beyond. Technical Report AIM-1549, MIT Artificial Intelligence Laboratory, Oct. 18 1995.
- [10] C. J. C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, pages 71–77, 1996.

- [11] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [12] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- [13] C. Castillo and C. Chang. An approach to vision-based person detection in robotic applications. *2nd Iberian Conference on Pattern Recognition and Image Analysis*, 2005.
- [14] C. Castillo and C. Chang. A method to detect victims in search and rescue operations using template matching. *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2005.
- [15] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [16] M. J. Chen, M. C. Chi, C. T. Hsu, and J. W. Chen. Roi video coding based on h.263+ with robust skin-color detection technique. *IEEE Transactions on Consumer Electronics*, 2003.
- [17] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [18] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
- [19] N. Duta, A. K. Jain, and M. P. Dubuisson-Jolly. Automatic construction of 2d shape models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):433–446, 2001.
- [20] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. 2003.
- [21] K. Fredriksson and E. Ukkonen. Faster template matching without fft. In *ICIP01*, pages I: 678–681, 2001.
- [22] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
- [23] D. Gavrila. Pedestrian detection from a moving vehicle. *Proc. of the European Conference on Computer Vision*, 2(8), 2000.
- [24] D. Gavrila and J. Giebel. Virtual sample generation for template-based shape matching. In *CVPR01*, pages I:676–681, 2001.
- [25] D. Gavrila and V. Philomin. Real-time object detection for “smart” vehicles. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume I, pages 87–93, Los Alamitos, CA, Sept. 20–27 1999. IEEE.
- [26] D. M. Gavrila and L. S. Davis. Fast correlation matching in large (edge) image databases. Technical Report CS-TR-3334, University of Maryland, College Park, MD, Aug. 1994.



- [27] R. Gonzalez and P. Wintz. *Digital Image Processing*. 1982.
- [28] M. Hu. Visual pattern recognition by moment invariants. *IT*, 8(2):179–187, February 1962.
- [29] M. Hu. Visual pattern recognition by moment invariants. In *CMetImAly77*, pages 113–121, 1977.
- [30] M. Hu. Pattern recognition by moment invariants. In *PIRE*, 49.
- [31] T. Kadir. Scale, saliency and scene description. Technical report, University of Oxford, Robotics Research Group, 2002.
- [32] D. Kendall. Shape manifolds, procrustean metrics and complex projective spaces. *Bull. London Math. Soc.*, 16:81–121, 1984.
- [33] H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, October 2003.
- [34] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Gradient-based learning for object detection, segmentation and recognition. Technical report, AT&T Labs, 1999.
- [35] Z. Liang and C. Thorpe. Stereo- and neural network-based pedestrian detection, 1999.
- [36] E. P. Lyvers and O. R. Mitchell. Precision edge contrast and orientation estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(6):927–937, 1988.
- [37] H. Mann and D. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat*, 18(1):50–60, 1947.
- [38] A. Mohan. Robust Object Detection in Images by Components. Master’s thesis, May 1999.
- [39] A. Mohan, C. Papageorgiou, and T. Poggio. Example based object detection in images by components. 1999.
- [40] S.Ñayar, H. Murase, and S.Ñene. Parametric appearance representation. In *Early Visual Learning*. Oxford University Press, February 1996., 1996.
- [41] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *In Proc. Computer Vision and Pattern Recognition, pages 193–199, Puerto Rico, June 16–20 1997.*, 1997.
- [42] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *1997 Conference on Computer Vision and Pattern Recognition (CVPR ’97), June 17-19, 1997, San Juan, Puerto Rico*. IEEE Computer Society, 1997.
- [43] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proceeding of Intelligent Vehicles*, pages 241–246, October 1998.

- [44] C. Papageorgiou, F. Girosi, and T. Poggio. Sparse Correlation Kernel Analysis and Reconstruction. Technical Report 1635, 1998. (CBCL Memo 162).
- [45] C. Papageorgiou and T. Poggio. Trainable Pedestrian Detection. In *Proceedings of the 1999 International Conference on Image Processing (ICIP-99)*, pages 35–39, Los Alamitos, CA, Oct. 24–28 1999. IEEE.
- [46] C. Papageorgiou and T. Poggio. A trainable system for object detection. 2000.
- [47] T. H. Reiss. The revised fundamental theorem of moment invariants. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(8):830–834, 1991.
- [48] A. Rosenfeld. *Picture Processing by Computer*. Academic Press, 1969.
- [49] A. Rosenfeld and A. Kak. *Digital Picture Processing*. Academic Press, 1982.
- [50] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.
- [51] H. Schweitzer, J. Bell, and F. Wu. Very fast template matching. In *ECCV02*, page IV: 358 ff., 2002.
- [52] M. Sonka, V. Hlavac, and R. Boyle. Image processing, analysis, and machine vision. In *Chapman and Hall*, 1993.
- [53] B. Stroustrup. The evolution of C++: 1985 to 1989. 2(3):191–250, Summer 1989.
- [54] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes, 2003.
- [55] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.
- [56] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfnder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.