PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO DE CLASIFICACIÓN DE OBJETOS PELIGROSOS EN RADIOGRAFÍAS UTILIZANDO EL MODELO BAG OF WORDS

MARCO ANTONIO ARIAS FIGUEROA

Tesis para optar al grado de
Magister en Ciencias de la Ingeniería

Profesor Supervisor:
DOMINGO MERY QUIROZ

Santiago de Chile, Agosto, 2016

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO DE CLASIFICACIÓN DE OBJETOS PELIGROSOS EN RADIOGRAFÍAS UTILIZANDO EL MODELO BAG OF WORDS

MARCO ANTONIO ARIAS FIGUEROA

Tesis presentada a la Comisión integrada por los profesores:
DOMINGO MERY QUIROZ
KARIM PICHARA
SERGIO HERNANDEZ
GONZALO CORTÁZAR

Para completar las exigencias del grado de
Magister en Ciencias de la Ingeniería

Santiago de Chile, Agosto, 2016

A mi familia y toda la gente que me
ha apoyado en esta etapa de mi vida

AGRADECIMIENTOS

Como primera instancia, quiero agradecer a todas las personas que me han apoyado durante este período:

- Agradecer a mis padres: Marco Antonio Arias Cardoza y Eliana Figueroa Aravena, por ser los pilares fundamentales de ánimo y compañia para seguir adelante.
- Agradecer a mi profesor supervisor, Domingo Mery Quiroz, por todos sus consejos, su guía y su paciencia conmigo. De no ser por él no habría logrado completar este trabajo.
- Agradecer al profesor Yadran Eterovic, por apoyarme y ser como un padre académico para mí.
- Agradecer a Erick Svec, compañero de trabajo, que me ayudó con ideas, conocimiento y mutuo apoyo en el desarrollo de nuestro trabajo.
- Agradecer a Orlando Vásquez, compañero y amigo, por sus ideas y apoyo en momentos difíciles.
- Agradecer a Cristián Ruz y Rodrigo Toro, compañeros y vecinos de oficina, por sus ideas y su apoyo
- Agradecer a mis mejores amigos, por todo su apoyo, su compañía y su fe en mí.
- Agradecer a Vladimir Riffo, por facilitarme las imágenes de shurikens, pistolas y hojas de afeitar.
- Por último, quiero agradecer al Fondecyt Nro. 1130934 de CONICYT

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

La detección y clasificación de objetos es uno de los principales problemas a los que se ha enfocado el reconocimiento de patrones y el aprendizaje de máquina. Existen diversos estudios cuyo fin es poder crear poderosos programas automáticos de reconocimiento que permitan, mediante una imagen, obtener información clave de objetos y personas. No obstante, existen pocos estudios que utilicen imágenes de radiografías, debido a la falta de bases de datos robustas y al poco interés masivo en ellas. Estos estudios permitirían realizar programas que detecten objetos peligrosos en equipajes, entre otras aplicaciones.

En este trabajo se propone una metodología enfocada en detectar objetos peligrosos en equipajes de aeropuertos, mediante el uso de radiografías. Se utilizan técnicas de reconocimiento de patrones mezcladas en un modelo de Bag of Words. Los pasos de esta metodología son: procesamiento de las imágenes, extracción de descriptores SIFT y LBP, creación del Bag of Words y entrenamiento mediante Random Forest. Usando este método se obtienen 3 clasificadores binarios de 3 objetos diferentes que se consideran peligrosos: pistolas, shurikens y hojas de afeitar. Estos clasificadores son evaluados utilizando diversas métricas de desempeño y mediante un nuevo set de imágenes a clasificar. Los mejores resultados (mostrados en porcentajes de Precision-Recall respectivamente para cada clase) fueron 100% y 82% para hojas de afeitar; 100% y 92% para shurikens y 88% y 99% para pistolas. El promedio de rendimiento fue de un 90%.

Este estudio demostró que es factible crear un clasificador de objetos peligrosos enfocado en radiografías, pudiendo ser extendido a más objetos si se ajustan los parámetros adecuados. En este trabajo sólo se estudia la clasificación de objetos, y no la detección.

Palabras Claves: Radiografías, Equipajes, Clasificador, Desempeño, SIFT, LBP, K-MEANS, *Random Forest*, filtro Gaussiano, reconocimiento de patrones, Training, Testing.

ABSTRACT

Object detection and classification is one of the main problems adressed by pattern recognition and machine learning. Several studies aims to create powerful automated recognition programs which, using images, can get important information about people or objects. However, few studies are focused on the X-ray image classification, due to the lack of robust database and soon massive interest in them. These studies could provide programs that detect dangerous objects in baggage, among other applications.

This work presents a methodology focused on detecting dangerous objects in baggage at airports, using X-rays images. This methodology uses pattern recognition techniques mixed in a Bag of Words model. The steps of this methodology are: image processing, extraction of SIFT and LBP descriptors, creation of Bag of Words model and training with Random Forest. This method creates 3 binary classifiers for 3 different objects that are considered dangerous: razors, shurikens and guns. These classifiers are evaluated through a new set of images to be classified (containing any of these items, or neither), and their behavior is studied using various performance metrics. Best results (showed as Precision-Recall percentages) were 100% and 82% for razors, 100% and 92% for shurikens and 88% and 99% for guns. The average percentage of classification was 90%.

This study demonstrates the feasibility of creating a dangerous objects classifier that use X-ray images, and it can be extended to more objects, if the proper parameters are set. In this work, only object classification is studied, and not object detection.

Keywords: X-ray images, Baggage, Classifier, performance, SIFT, LBP, K-MEANS, Random Forest, Gaussian filter, pattern recognition, Training, Testing.

1. INTRODUCCIÓN

El ser humano aprecia todo lo que lo rodea mediante sus sentidos, principalmente utilizando la visión, ya que es de los sentidos que más información nos entrega, por lo cual se suele anteponerlo frente a otros. A medida que crecemos, solemos aprender a diferenciar todo tipo de objetos mediante nuestra vista, por ejemplo, podemos distinguir una mariposa de una mosca o una abeja, ya sea por las alas, sus colores, la forma de su cuerpo, entre otras características. Incluso podemos ir más alla, ya que tenemos la capacidad de distinguir diferentes tipos de mariposa, pero sabiendo que todas ellas son diferentes a una abeja, aún cuando el tamaño de sus alas sea diferente, sus colores o incluso su morfología . Estos simples hechos plantean diferentes preguntas: ¿Cómo es posible que logremos realizar estas acciones de manera tan certera, en un par de milisegundos y con un casi nulo porcentaje de error?, ¿en qué nos percatamos para decir que este insecto es una mariposa y no una abeja?.

Existen variados estudios que se enfocan a dar respuesta a estas preguntas, e incluso existen áreas que buscan el tratar de "imitar" esta capacidad humana tan útil. Dentro del área de computación, áreas como la visión por computador y el reconocimiento de patrones se dedican a estudiar y crear sistemas autónomos que puedan analizar y procesar información obtenida de imágenes o video, lo que sería equivalente a los ojos del computador. No obstante, aún estamos muy lejos de igualar la capacidad de diferenciación del cerebro humano (Grill-Spector, 2003), puesto que, una característica tan trivial como el color o tamaño de las alas de una mariposa, no lo es para un computador. Esto se debe a que el computador no utiliza las mismas características de diferenciación que utilizan los humanos.

Lo anterior propone la pregunta: ¿Qué características son las que puede utilizar un computador para extraer información útil de una imagen o video?. Para esto existen diferentes enfoques: por ejemplo, el uso de vectores que permitan "describir" la imagen, conocidos como descriptores, que pueden definir puntos de interés (Lowe, 2004), texturas (Ojala, 1994), color (Van De Sande et al., 2010), entre otros. Otros enfoques son la bolsa de palabras o Bag of Words (Tsai, 2012), descriptores de forma (Belongie et al., 2002), etc.

Pese a existir tantos estudios sobre el área, todavía puede haber personas que se pregunten lo siguiente: ¿por qué necesitaríamos un sistema que realice una labor que el ser humano maneja tan bien?. La respuesta es simple, se pueden llegar a obtener una gran cantidad de beneficios, ya sea automatizando y/o mejorando la velocidad de procedimientos o trabajos que para un ser humano pueden ser difíciles, aburridos o incluso imposibles de realizar. Algunos ejemplos son: sistemas de seguridad que detecten si hay una persona sospechosa (Z. Zhang et al., 2005), sistemas de revisión de alimentos (Brosnan and Sun, 2002), sistemas de alarma automática para accidentes (Wei and Hanbo, 2011), sistemas de detección de señalética que adviertan al conductor, sistemas de revisión de equipaje en aeropuertos (a lo que se enfoca este trabajo) (Mery, Mondragon, et al., 2013), entre muchos otros.

Sin embargo, pese a todos los avances en el área de clasificación de objetos, no existe un "super algoritmo" que sirva para todos los casos posibles (así como lo hace nuestro cerebro (Grill-Spector, 2003)), ya que cada problema es muy diferente de otro. Por esta razón, es necesario enfocar los estudios por diferentes puntos de vista, partiendo por el hecho de que las imágenes de entrada que reciben no son las mismas en todas las circunstancias.

1.1. Problema de clasificación de objetos en radiografías

Existen diversos estudios en el área de reconocimiento de patrones dedicados a investigar y desarrollar un algoritmo o metodología para un problema particular, ya sea de detección, clasificación, seguimiento, etc. No obstante, existe un área que no ha sido estudiada tan en profundidad como las otras, que es el caso de las radiografías, o imágenes de rayos X.

Esto se debe principalmente a que no existen grandes bases de datos que permitan su estudio, diferente al caso de imágenes fotográficas que basta con buscar en Google para encontrar millones de ellas. Otra razón de este hecho es que la cantidad de problemas que utilizan este tipo de imágenes son más acotados, pero no por esto menos importantes. Algunos ejemplos son: la detección de tumores cancerígenos (Vyborny and Giger, 1994), detección de fallas en soldaduras (Silva and Mery, 2007) o la clasificación de objetos peligrosos en equipajes (Mery, Mondragon, et al., 2013). Este último se presenta como motivación a este trabajo.

1.2. Motivación

Por razones de seguridad, en todos los aeropuertos del mundo se realiza inspección de equipaje a los pasajeros, con el fin de evitar el riesgo de llevar objetos potencialemnte peligrosos dentro de los aviones, tales como navajas, cuchillos, artículos cortopunzantes, armas, etc. que pueden resultar en una potencial amenaza.

En la actualidad, se capacitan operadores humanos que constantemente revisan los equipajes mediante rayos X. No obstante, estos operadores tienen pocos segundos para revisar cada uno de los equipajes, lo que baja la tasa de detecciones exitosas a un 90% o incluso 80% (Michel et al., 2007), sobre todo en los horarios punta, donde el personal no da abasto a tanta demanda. Debido a esta situación, de no detectar alguno de estos objetos a tiempo, podrían ocurrir problemas muy graves durante los viajes, desde heridas menores hasta terrorismo.

Una de las soluciones inmediatas que pueden surgir para optimizar la revisión de equipaje es usar la tecnología, o en este caso particular, la visión por computador y el reconocimiento de patrones. Por esta razón, surge la idea de un sistema de escaneo autónomo de equipajes, que logre reconocer objetos prohibidos en el equipaje de mano, que puedan ocasionar problemas durante el vuelo, tales como navajas, cuchillos, armas, etc.

Este trabajo surgió por la necesidad de evitar problemas similares a éste, donde la única opción que se tiene de revisar el contenido de los equipajes es mediante radiografías. Se pretende con este trabajo mostrar una metodología (algoritmo) de clasificación de objetos que funcione en un ambiente natural de forma eficiente. Además, se pretende mejorar el estado del arte mostrando un estudio basado en el algoritmo sobre los aspectos importantes a considerar para desarrollar, a futuro, un sistema óptimo completo que evite problemas similares al descrito.

FIGURA 1.1. Equipo de inspección de equipajes utilizado en el Aeropuerto de Bangkok.
Fuente: *Wikipedia - Airport security*, 2015

1.3. Descripción del Problema

1.3.1. Hipótesis

La hipótesis de este trabajo es que el desarrollo de un algoritmo de clasificación que utiliza técnicas de reconocimiento de patrones e inteligencia de máquina, tales como *Bag of Words* (Tsai, 2012) y características (*features*), tales como SIFT (Lowe, 2004) y LBP (Ojala et al., 1994), utilizadas frecuentemente en imágenes convencionales, puede llegar a tener un desempeño eficiente en cuanto a precisión y otras métricas, si es utilizado en pro-blemas de detección de objetos que involucren imágenes de rayos X, tales como inspección de equipajes.

1.3.2. Objetivos

En concordancia con la hipótesis propuesta, el objetivo general de esta tesis consiste en desarrollar una metodología que permita crear diferentes clasificadores, entrenados mediante una base de datos de imágenes rotuladas, que luego son utilizados para decidir si una nueva

imagen, no utilizada en el set de entrenamiento, pertenece o no a la clase buscada (arma, navaja, etc). Específicamente este trabajo se enfoca en la detección de objetos peligrosos en equipajes, tales como hojas de afeitar (*razors*), shurikens y pistolas (*guns*).

Dentro de los objetivos específicos se encuentran los siguientes:

- Diseñar e implementar una metodología de desarrollo propio que utilice Bag of Word en radiografías de equipajes.
- Crear un algoritmo con la metodología anterior que utilice un set de imágenes de objetos de equipaje previamente rotulado, con el cual se obtengan diferentes clasificadores para cada uno de los objetos.
- Presentar un estudio del desempeño de cada clasificador, ya sea por separado o en un mismo programa, utilizando un nuevo set de datos (radiografías de equipajes).

1.4. Estado del arte e investigaciones previas

El análisis de equipajes mediante imágenes de rayos X comenzó como una necesidad de evitar posibles problemas en los vuelos, además de brindar seguridad tanto a los pasajeros como a la tripulación. Se han utilizado diversas técnicas para impedir el paso de objetos potencialmente peligrosos dentro de los aviones, que pueden ser utilizados en actos delictuales, como asesinatos, secuestros o atentados. No obstante, su importancia aumentó drásticamente después del atentado del 11 de septiembre, dado que era la primera vez que ocurría un atentado a tan gran escala, que costó la vida de miles de personas. En estos años las tecnologás de inspección de equipajes han ido en aumento y en mejora constante, con el fin de evitar catástrofes como la del 11 de Septiembre en USA (Murphy, 1989).

1.4.1. Inspección de equipaje previo al 11 de Septiembre

Antes del atentado, las tecnologías que se utilizaban para el escaneo de los equipajes se enfocaban principalmente en la obtención y el procesamiento de las imágenes. Sin embargo, no existían metodologías que permitieran el análisis de las imágenes, de tal forma de detectar de forma automática una posible amenaza en los equipajes.

Algunas de las tecnologías utilizadas para detectar y detener las amenazas terroristas estaban basadas en Thermal-Neutron Activation (TNA), en donde un container es escaneado con rayos X para buscar posibles objetos; si se descubre un posible objetivo, se lanza un rayo de neutrones que produce reacciones térmicas con ciertas sustancias (Armistead, 1998). Otra tecnología ocupada fue la activación rápida de neutrones (FNA), donde se crea un escáner con rayos gamma y neutrones de rápida velocidad, los cuales producen imágenes de alta calidad de los equipajes sin necesidad de abrirlos (Eberhardt et al., 2005). Una tercera metodología aplicada fueron los rayos X de energía dual, los cuales fueron utilizados en medicina desde la década de los 70. Estos rayos poseen un porcentaje de error mucho menor que los rayos X convencionales, además de brindar un preciso análisis de composición de las personas, con una menor exposición a la radiación (Mazess et al., 1990).

En la década de los 90, se desarrollaron los sistemas de detección de explosivos basados en rayos X (EDS). Estos sistemas poseen un alto rendimiento en comparación a los anteriores, ya que combinan detecciones realistas con porcentaje adecuados de falsas alarmas, los cuales se requieren cuando hay un gran flujo de personas diariamente circulando en los aerpuertos (Murray and Riordan, 1995). Estos sistemas obtienen tomografías que comparan la estructura de los materiales comunes con estructuras referenciales que se poseen de explosivos y drogas (Strecker, 1998).

1.4.2. Detección de objetos automática usando radiografías

Desde el atentado del 11 de septiembre, comenzó a aumentar el desarrollo de técnicas de detección automática y semiautomática de amenazas, aumentando también el interés científico en este tipo de problemas debido a la importancia que obtuvo. Estas técnicas, en primera instancia, utilizaban reconocimiento 3D. No obstante, las radiografías presentan una serie de dificultades en comparación a imágenes convencionales, tales como la trasparencia de la imagen, su baja nitidez y la superposición y rotación de objetos, los cuales pueden perderse de vista (Zentai, 2010).

Algunas de las contribuciones importantes en esta última década se detallan a continuación. Por un lado se crean métricas de velocidad para estimar el rendimiento en tareas de

detección de equipaje usando rayos X (Wales et al., 2009). Por otro lado se utilizan imágenes de rayos X pseudo-coloreadas, que permitían una mayor efectividad en la detección humana, ya que el color es una característica que le permite a los humanos diferenciar con mayor efectividad diferentes objetos (Abidi et al., 2006). También se utilizan imágenes de rayos X mejoradas utilizando métodos de segmentación y redes neuronales, que buscan las posibles zonas de interés y eliminan las innecesarias, de modo de optimizar las imágenes y que sea más fácil su análisis (Singh and Singh, 2005). Por último, existen algoritmos de detección de objetos amenazantes (principalmente pistolas), los cuales utilizaban características de textura (Oertel and Bock, 2006), redes neuronales que lograron un 80% de desempeño (Liu and Wang, 2008) y clasificadores SVM (Nercessian et al., 2008).

Pese a los avances tecnológicos, y sumando la dificultad de trabajar con radiografías, actualmente provoca que el trabajo del personal que revisa constantemente los equipajes siga siendo de suma importancia hasta la actualidad, ya que es muy difícil (o casi imposible) desarrollar una tecnología que tenga un 100% de certeza y efectividad.

1.4.3. Detección en equipajes usando múltiples vistas

Actualmente existen diversos estudios e investigaciones enfocados en la detección y clasificación de objetos en radiografías. Muchos de estos trabajos comenzaron a utilizar múltiples vistas del mismo objeto, ya que estas imágenes aportaban información útil que resulta necesaria para la detección de algunos objetos, puesto que es casi imposible reconocerlos desde un sólo punto de vista. Estudios muestran que detectores humanos que usan múltiples vistas versus los que utilizan sólo un punto de vista mejoran su desempeño considerablemente, sobre todo cuando se encuentran en condiciones difíciles o imágenes de alta complejidad (Bastian et al., 2008). A continuación se detallan algunos trabajos que utilizan esta metodología de múltiples vistas.

Uno de los primeros trabajos realiza la creación y síntesis de nuevas imágenes de rayos X que muestran profundidad, utilizando Kinetic Depth Effect X-ray (KDEX), las cuales mejoran el desempeño de la detección (Abusaeeda et al., 2011). Estas imágenes se forman utilizando descriptores SIFT combinado con geometría epipolar (Hartley and Zisserman, 2003).

Otro trabajo realiza detección de objetos en múltiples vistas utilizando rayos X de energía dual (Franzel et al., 2012). En este trabajo se realizan 3 grandes contribuciones: primero se analizan las variaciones de apariencia de los objetos en imágenes de rayos X. Luego se adaptan métodos estándar de detección de objetos utilizando rayos X de e-nergía dual y las variaciones de apariencia antes mencionadas. Por último, se muestra una metodología de detección que mezcla vistas singulares con múltiples vistas y toma ventaja de las consistencias geométricas de los objetos.

Una tercera investigación estudia una metodología activa que permite buscar el mejor punto de vista del objeto de la imagen, utilizando diferentes vistas de la misma, con el objetivo de obtener la mejor imagen del objeto y su mejor análisis respectivo (Riffo and Mery, 2012).

Por otro lado, existen trabajos que se enfocan en realizar el seguimiento (*tracking*) de un objeto a través de múltiples vistas, con el objetivo de verificar que la decisión tomada sea consistente o no (Berclaz et al., 2011). De serlo, entonces la posibilidad de que sea un objeto de interés es alta. Uno de estos trabajos se enfoca en utilizar geometría epipolar e información de intensidad, con el fin de encontrar la ubicación de pistolas dentro de equipajes, enfocándose en la detección del gatillo de las mismas (Mondragon et al., 2013). Otra investigación de esta índole realiza un análisis monocular de cada vista para obtener posibles detecciones, para luego reconocer el objeto de interés realizando un seguimiento conjunto en todas las vistas (Mery, Riffo, et al., 2013). Un trabajo más reciente que ocupa esta metodología consiste en un modelo que realiza 2 pasos importantes: estima la estructura de los objetos a detectar a partir de múltiples vistas del objeto, para luego detectar partes de interés mediante un algoritmo de segmentación y seguimiento, basado en la geo-metría y en la apariencia (Mery, 2015).

1.4.4. Trabajos base de esta propuesta

Dado que este trabajo se enfoca en la clasificación y no en la detección de objetos, el enfoque de este trabajo se concentró en buscar metodologías que utilizaran imágenes únicas como set de entrenamiento, vale decir, imágenes que contengan un objeto central sin otros objetos distractores.

Uno de los trabajos que se utilizó como base para esta metodología fue el de Parkhi et al. (2012). Este trabajo no utiliza radiografías, pero contempla un modelo de apariencia que se tomó como base dada su alta efectividad y relativamente fácil entendimiento e implementación. En este trabajo se desarrolla un clasificador de perros y gatos según una imagen de los mismos. Dentro de toda su metología, existe una etapa que se encarga de extraer un modelo de apariencia (textura) del animal, utilizando una bolsa de palabras creada a partir de los descriptores SIFT y LBP, por medio del algoritmo *K-means*. Una idea similar fue utilizada en este trabajo, mezclando características de textura con una bolsa de palabras.

En lo que respecta a la clasificación, se tomó la idea de que utiliza Maturana et al. (2011) en donde se utilizan descriptores LBP y diferentes árboles de decisión como clasificadores, con el objetivo de clasificar diferentes caras según su tonalidad de piel, sexo o incluso raza. En este trabajo también se utilizó un clasificador SVM dada su alta efectividad y robustez en imágenes convencionales. Se comparó el desempeño de SVM con árboles de decisión utilizando los mismos descriptores.

También se utilizaron ideas de los papers mencionados en la sección anterior como base del algoritmo desarrollado en esta tesis.

1.5. Alcance y contribución de esta propuesta

La metodología desarrollada en este trabajo clasifica una imagen recibida en el objeto correspondiente (si es o no una pistola, shuriken u hoja de afeitar), una vez obtenida una posible imagen de un objeto peligroso. No se toma en consideración que el objeto puede tener una determinada ubicación dentro de la imagen, ni se asume que puedan existir otros objetos dentro de la misma, lo que sí ocurriría en un caso real.

Teniendo en consideración lo anterior, se puede explicar que la principal contribución de este trabajo al estado del arte actual es mostrar un estudio detallado de todos los factores que influyen en la clasificación de objetos en radiografías, desde los métodos utilizados hasta los parámetros que influyen en éstos.

El algoritmo propuesto es un método relativamente sencillo y rápido de clasificación de objetos, el cual puede ser extendido a más objetos (tales como cuchillos), si se cuenta con imágenes representativas del mismo. Esto puede realizarse entrenando un nuevo modelo que contemple la nueva clase, ajustando los parámetros adecuados.

Pese a ser una propuesta sencilla, este algoritmo es robusto a escalas y rotaciones de los objetos, dado los descriptores e imágenes de entrenamiento que utiliza. Además es robusto a ruido, siempre y cuando el objeto no se encuentre ocluído por otro similar.

Además se pretende mostrar un estudio del comportamiento que tienen técnicas mundialmente utilizadas como SIFT y LBP, mezcladas con el modelo Bag of Words y el algoritmo de clasificación *Random Forest* para las radiografías.

1.6. Organización del documento de Tesis

El siguiente documento de Tesis se organiza de la siguiente manera: en el capítulo 2 se muestran todos los conceptos básicos que son necesarios para entender la propuesta descrita, desde el filtro Gaussiano, descriptores SIFT y LBP, clasificador *Random Forest*, el modelo Bag of Words, el algoritmo de *clustering* K-means y las diferentes métricas de medición utilizadas en este trabajo. En el capítulo 3 se explica tanto el ambiente de desarrollo utilizado como la metodología propuesta, incluyendo su fase de *Training* y su fase de *Testing*, las cuales consideran todos los conceptos básicos previamente explicados. Más tarde en el capítulo 4 se muestra tanto la base de datos utilizada, como todos los resultados relevantes obtenidos de los experimentos, divididos en 4 secciones: ajuste de parámetros de cada clasificador, análisis de sensibilidad del algoritmo, desempeño particular y desempeño final de clasificación. Finalmente, en el capítulo 5 se dan a conocer conclusiones y posibles trabajos futuros que pueden surgir a partir de este trabajo.

2. CONCEPTOS BÁSICOS

En este capítulo se explica todo el marco conceptual necesario para entender la solución propuesta en este trabajo. Se abordan temas relacionados a los ámbitos de procesamiento de imágenes, reconocimiento de patrones y aprendizaje de máquina, los cuales son necesarios para entender los conceptos de filtros, extracción de características, aprendizaje supervisado y métricas de desempeño.

Este capítulo se divide en 4 secciones: La primera explica conceptos básicos sobre el procesamiento de imágenes, para luego explicar el caso particular del funcionamiento de un filtro Gaussiano utilizado en la metodología propuesta; la segunda parte explica la metodología básica utilizada en el reconocimiento de patrones, desde la extracción de características, su descripción y el posterior entrenamiento. Además se explica el funcionamiento de los descriptores SIFT y LBP. Luego en la tercera parte se explican conceptos de aprendizaje de máquina, desde *clustering*, el modelo de *Bag of Words* utilizado en este trabajo y el método de clasificación por *Random Forest*. Finalmente en la cuarta parte se explican las métricas de desempeño utilizadas en los resultados, explicando lo que es una matriz de confusión y un gráfico de Precision-Recall.

2.1. Procesamiento de imágenes

El área de procesamiento de imágenes se dedica al análisis y el uso de técnicas sobre imágenes con diferentes fines: mejorar una imagen, realizar efectos sobre la misma (por ejemplo aumento de ruido) o facilitar la búsqueda de información útil sobre la misma. Para entender el contexto del procesamiento de imágenes, se debe entender primero lo siguiente: cada pixel de la imagen se describe por un valor de intensidad $P(x, y)$, que varía entre 0 y 255, y donde x e y corresponden a las coordenadas donde se ubica el pixel en la imagen. En el caso de imágenes a color cada pixel tiene 3 valores de intensidad, correspondientes a la intensidad de rojo, azul y verde.

A esta representación de una imagen se le pueden aplicar diversas técnicas, donde una de ellas es el filtro de imagen. Básicamente un filtro consiste en una función aplicada a cada

pixel de la imagen, mediante una convolución entre la misma y la función, lo cual cambia el valor de cada pixel, generando una imagen diferente con información que no se puede percibir a simple vista en la imagen original.

Dentro de las utilidades del uso de filtros se encuentran: suavisado de imagen, donde se reduce la diferencia de intensidad (valores) entre pixeles vecinos, lo que provoca que la imagen se vea más borrosa, principalmente en los bordes; eliminación de ruido, vale decir, eliminar pixeles cuyo valor de intensidad es muy diferente al de sus vecinos, lo que entrega información confusa; realce y detección de bordes (esquinas), entre otros.

Para este trabajo es necesario entender el funcionamiento de un filtro Gaussiano.

2.1.1. Filtro Gaussiano

Un filtro Gaussiano consiste en una convolución entre una imagen y una función Gaussiana. Esta convolución tiene el efecto de reducir los componentes de alta frecuencia de una imagen, atenuándolos y acercándolos a la intensidad más cercana que tiene una mayor probabilidad de aparecer, lo que muestra como resultado una imagen más borrosa.

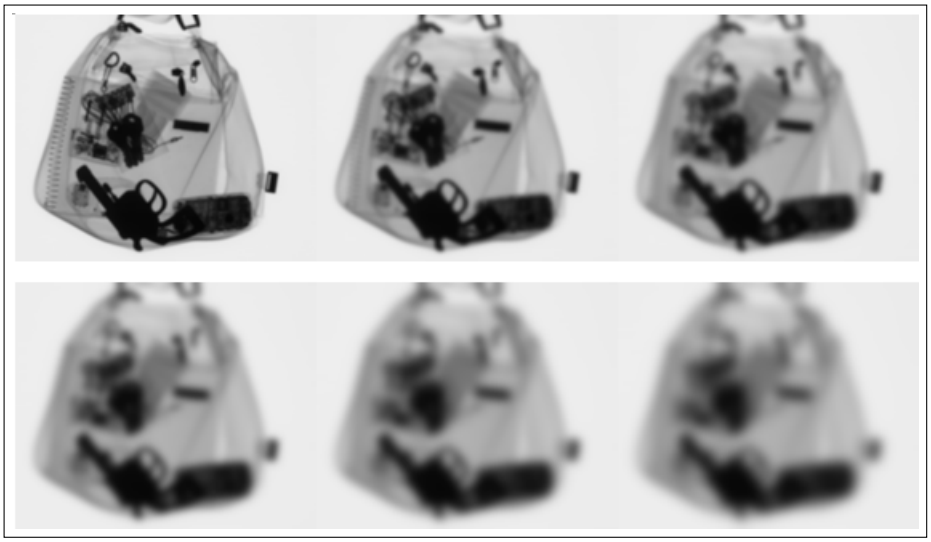

FIGURA 2.1. Aplicación de filtro Gaussiano a una imagen de rayos X utilizando diferentes valores de σ. De izquierda a derecha avanzando de arriba hacia abajo, valores de σ iguales a 2, 4, 6, 8, 10 y 12.

La función Gaussiana se define de la siguiente forma:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

donde x e y son las coordenadas correspondientes al pixel y σ corresponde a la desviación estándar de la función, que se aplica a cada pixel. Mientras más alto el valor de σ, mayor será la cantidad de pixeles vecinos afectados y por ende, más borrosa será la imagen final. La figura 2.1 muestra un filtro Gaussiano aplicado a la misma imagen, pero con diferentes valores de varianza.

2.2. Reconocimiento de Patrones

El reconocimiento de patrones es un área encargada del análisis de diferentes objetos con el fin de detectarlos y/o clasificarlos buscando patrones comunes. Un patrón común es una descripción o característica presentada en varios objetos que permite clasificarlos en una categoría común. Existen diversos tipos de patrones: algunos hablan de la forma, otros de los colores, estructura, acciones, etc. Un patrón es descrito extrayendo información del objeto y, mediante algún proceso u algoritmo, lograr una representación numérica que pueda caracterizar el patrón. Existen patrones que no son percibibles al ojo humano, pero son de vital importancia para un computador, por ejemplo, transformada de Fourier (Lim, 1990), histograma de gradientes orientados (Dalal and Triggs, 2005), entre otros.

Las etapas del proceso de reconocimiento de patrones son: lectura de imágenes, extracción de características similares (*features*) que describan el patrón, representación de las mismas, entrenamiento de una función y toma de decisión final. La representación de estas características se conoce como descriptores, los cuales consisten en un modelo matemático que representa el patrón estudiado o utilizado. La fase de entrenamiento será explicada en la sección siguiente.

En este trabajo se utilizan 2 tipos de descriptores: SIFT y LBP. Tanto para entender SIFT como LBP, es necesario explicar el concepto de histograma, que se explica a continuación.

2.2.1. Histograma

Un histograma corresponde a un gráfico de barras, en donde cada observación se coloca en el eje X y la cantidad de la misma en el eje Y. Para el caso de las imágenes, el histograma sirve para tener una visión cuantitativa de las diferentes características que describen una imagen, por ejemplo, cuantos pixeles rojos tiene, cuantos bordes, etc. Si se utilizan descriptores, tales como SIFT y LBP, el histograma sirve para saber cuantos descriptores similares posee, para así disminuír la cantidad de cómputo sobre todos ellos. Las observaciones del eje X se conocen como los diferentes intervalos del histograma.

Un histograma también puede ser representado como un vector v, donde la posición p del vector representa el bin en la posición p del gráfico, mientras que los números en $v[p]$ corrresponden a las cantidades del eje Y. Un ejemplo de un histograma de colores se muestra en la figura 2.2

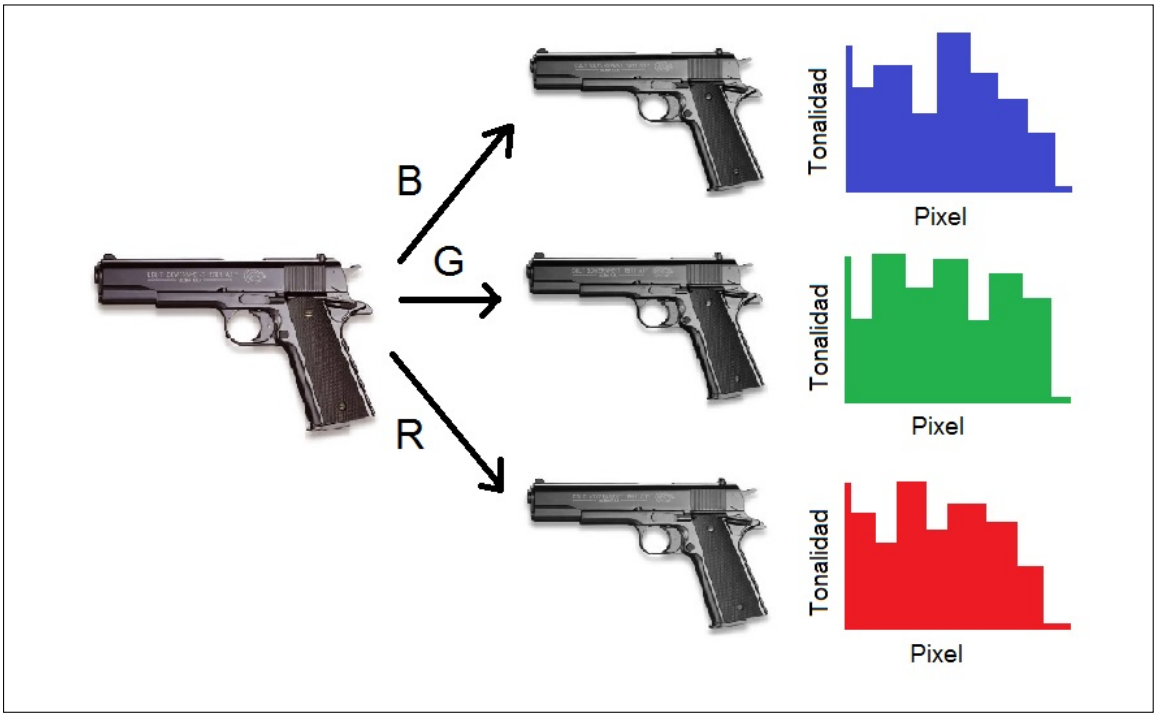

FIGURA 2.2. Ejemplo que muestra un histograma para cada una de las 3 tonalidades de colores de la imagen. A la izquierda la imagen original; al centro las 3 imágenes en las escalas B, G y R respectivamente, y a la derecha sus respectivos histogramas.

2.2.2. Descriptores SIFT

Scale-Invariant Feature Transform (SIFT) es un descriptor descrito por Lowe (2004) que tiene la gracia de ser invariante a escalas y rotaciones, además de comportarse muy bien ante variaciones de puntos de vista, luminocidad, etc. En otras palabras, este descriptor permite detectar y describir puntos importantes de la imagen, que son identificables en otra imagen similar, aunque el objeto en la nueva imagen se encuentre rotado, opacado, más grande o con mayor o menor luminosidad.

Para construir el descriptor se realizan una serie de pasos:

En primer lugar, se toma la imagen $I(x, y)$ y ésta se convoluciona con un kernel Gaussiano $G(x, y, \sigma)$, como se explicó en la sección 2.1.1, a una determinada escala σ según la función 2.2.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.2}$$

Para construir el kernel Gaussiano se utiliza una diferencia de Gaussianas, es decir, una resta de 2 imágenes a las que se le aplica un filtro Gaussiano con varianzas de σ y $k * \sigma$, como indica la ecuación 2.3.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k \cdot \sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k \cdot \sigma) - L(x, y, \sigma) \end{aligned} \tag{2.3}$$

Para construir $D(x, y, \sigma)$, se aplican variados filtros Gaussianos, variando cada σ elegido con un k determinado hasta llegar a $2 \cdot \sigma$, formando un espacio de diferentes imágenes filtradas, las cuales se restan para obtener las diferencias, como se muestra en la figura 2.3.

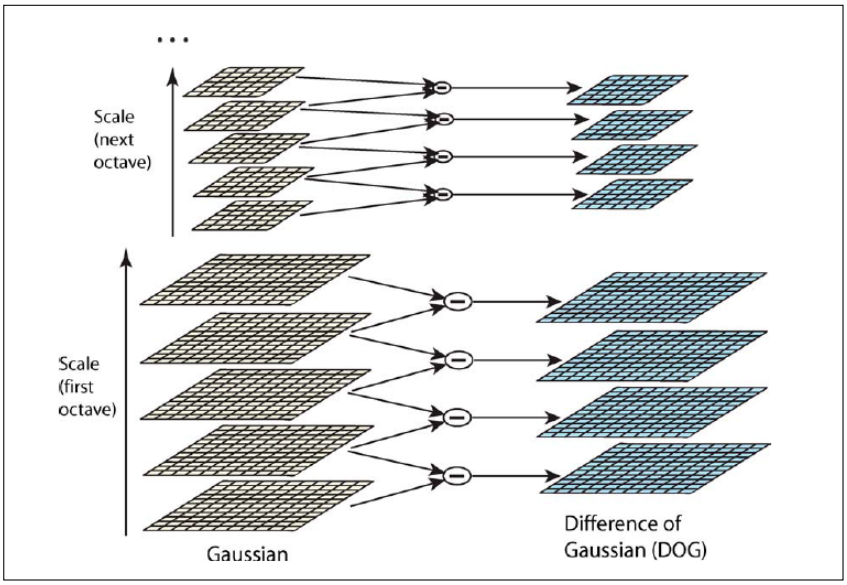

FIGURA 2.3. Esquema de resta de Gaussianas con diferentes valores de sigma aplicadas a una misma imagen. Cada imagen a la izquierda corresponde a la imagen original con un filtro Gaussiano y las de la derecha corresponden a la resta entre 2 imágenes adyacentes.
Fuente: *Lowe (2004)*

Esta resta de imágenes corresponde a $G(x, y, k \cdot \sigma) - G(x, y, \sigma)$, la cual luego se convoluciona con la imagen original, formando cada $D(x, y, \sigma)$.

Una vez terminado este paso, se buscan los *keypoints* como los máximos y mínimos en D. Para encontrarlos, cada pixel $D(x, y)$ se compara con sus 8 vecinos en la misma escala, mas los 9 vecinos de las escalas superior e inferior respectivamente. Si un pixel es mayor o menos a sus 26 vecinos, entonces es candidato a ser punto clave (*keypoint*) y se guarda su ubicación y la escala a la cual pertenece (el valor de $\epsilon = k \cdot \sigma$), lo que deja un punto (x, y, ϵ). Al final de este paso se obtiene una cantidad n de posibles *keypoints*.

Luego, se utilizan 2 criterios para descartar puntos clave: el primero se encarga de remover aquellos puntos con bajo contraste (y que son sensibles al ruido), mientras que el segundo criterio elimina los puntos que son posibles bordes, utilizando su radio de curvatura.

Para el primer paso, se construye la expansión de Taylor de $D(x)$, la cual se deriva e iguala a 0, como indican las ecuaciones 2.4 y 2.5.

$$D(x) = D + \frac{\partial D^{-1}}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x \tag{2.4}$$

$$\hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \tag{2.5}$$

Posteriormente se eliminan todos los puntos $D(\hat{x})$ que están bajo un threshold, donde normalmente se utiliza un threshold de 0,03, ya que este valor elimina los valores de los extremos que no son relevantes, según los experimentos de Lowe (2004).

Para el segundo criterio, se utiliza el radio de curvatura. Si su curvatura r es mayor que 10, entonces el punto se encuentra sobre una línea recta, siendo difícil determinar su posición en presencia de ruido, así que se elimina. Matemáticamente se calcula la matriz Hessiana y se comprueba con la siguiente ecuación 2.6, tomando $r = 10$.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{2.6}$$

El siguiente paso es calcular a cada pixel (x_0, y_0, ϵ) encontrado su orientación. Para esto, primero se busca la imagen L cuya convolución con el filtro Gaussiano sea lo más parecida a la escala ϵ del punto. Luego, a cada punto $L(x, y)$ se le calcula su magnitud $m(x, y)$ y su orientación $\theta(x, y)$, según las ecuaciones 2.7 y 2.8 respectivamente. Cada maginitud $m(x, y)$ se pondera por un peso igual a $1.5 \cdot \epsilon$, centrado en (x_0, y_0). Con esta información, se construye un histograma de 36 bins, que representa los 360 grados de orientación y a éste se le agrega $m(x, y)$ según $\theta(x, y)$ y ponderada por el peso anterior. El resultado es un histograma ponderado de todas las orientaciones de los pixeles obtenidos como *keypoints*, donde el valor p más alto será el que represente la orientación (ángulo) del punto (x_0, y_0) correspondiente.

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{2.7}$$

$$\theta(x, y) = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \tag{2.8}$$

Con la ubicación, orientación y escala de cada *keypoint*, el siguiente paso es construir el descriptor como tal. Para construir este descriptor se realiza lo siguiente: para cada *keypoint* (x, y, ϵ, θ) se le genera una ventana de 16x16 pixeles centrada en (x, y). Estos pixeles son separados en grupos de 4x4 pixeles y rotados según el ángulo θ, para que el descriptor sea invariante a las rotaciones. Además, cada pixel es ponderado por un peso depediendo de su distancia al pixel central: mientras más alejado, menor será su peso.

Con estos datos, para cada celda de 4x4 pixeles se le crea un histograma de 8 bins de las orientaciones. Como las orientaciones de cada pixel no caen en un valor central, éstas son interpoladas con los pixeles vecinos, de modo de repartir la magnitud en cada orientación según corresponda. Dado que cada ventana posee 16 celdas de 4x4, significa que se generan 16 histogramas de 8 bins. Éstos son concatenados uno detrás de otro, formando el descriptor final, que es un vector que posee un largo de $4 \cdot 4 \cdot 8 = 128$. La figura 2.4 muestra un ejemplo de un descriptor SIFT.

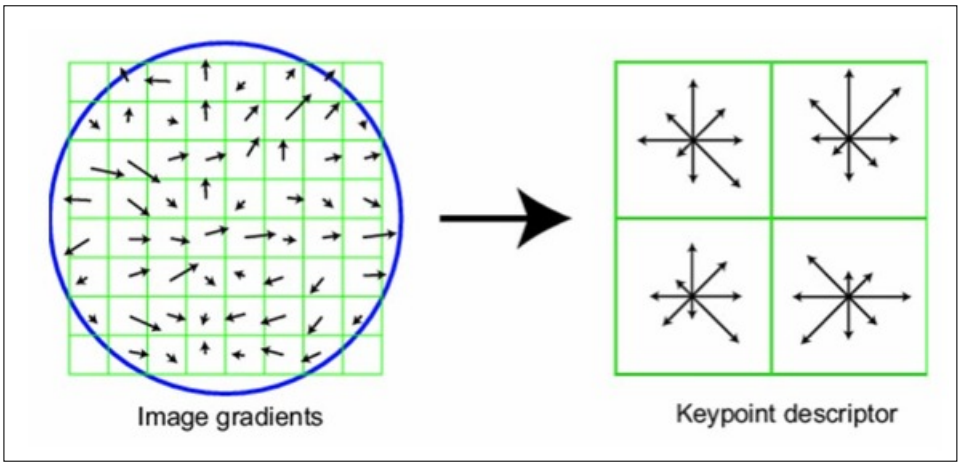

FIGURA 2.4. Esquema que muestra la creación del descriptor SIFT. Para este ejemplo se utilizaron celdas de 8x8 en grupos de 2x2. A la izquierda se encuentra la celda de 8x8 con la orientación de cada pixel vecino. A la derecha se encuentra el histograma de orientaciones formado en cada grupo de 2x2.
Fuente: *Lowe (2004)*

Lo que se obtiene finalmente es una lista de vectores, donde cada vector es un descriptor SIFT de la imagen, cada uno de ellos de largo 128. La cantidad de *keypoints* obtenidos varía de una imagen a otra.

2.2.3. Descriptores LBP

Local Binary Patterns (LBP) es un descriptor utilizado en reconocimiento de patrones, que observa el espectro de textura que poseen los objetos en la imagen, utilizando la imagen en escala de grises (vale decir, sólo una intensidad por pixel). El descriptor se construye de la siguiente manera:

- Se divide la imagen en celdas, donde cada celda contiene una cantidad fija de pixeles. Usualmente se utlizan celdas de 16x16, 32x32 ó 64x64
- Cada pixel se compara con sus 8 vecinos, en dirección reloj o contrareloj, observando si el pixel comparado es mayor o menor en intensidad.
- En cada comparación, si el pixel es mayor en intensidad, se coloca un 1 y si es menor, un 0. Con esto, se obtiene un código binario de 8 dígitos para cada pixel que representa la comparación de éste con sus vecinos.
- Luego se realiza un histograma de los pixeles de cada celda, utilizando los códigos obtenidos.
- Este histograma se normaliza, para luego ser concatenado con todos los otros histogramas, obteniendo el descriptor final.

El vector obtenido consta de 36 bins, cada uno representando uno de los códigos obtenidos. Este vector es una forma de representar las características de textura de la imagen, ya que representa, mediante un código binario, la diferencia de intensidad entre pixeles vecinos.

Adicionalmente en este trabajo, se realizó una variante de este método, donde en vez de obtener LBP completo para toda la imagen, se obtuvieron las posiciones (x, y) donde se encuentra un *keypoint* obtenido por SIFT. Alrededor de este punto se creó un *patch* de un ancho igual a la escala ϵ del keypoint multiplicada por un factor. A cada uno de estos patch se le extrajo LBP, no solamente a la imagen completa.

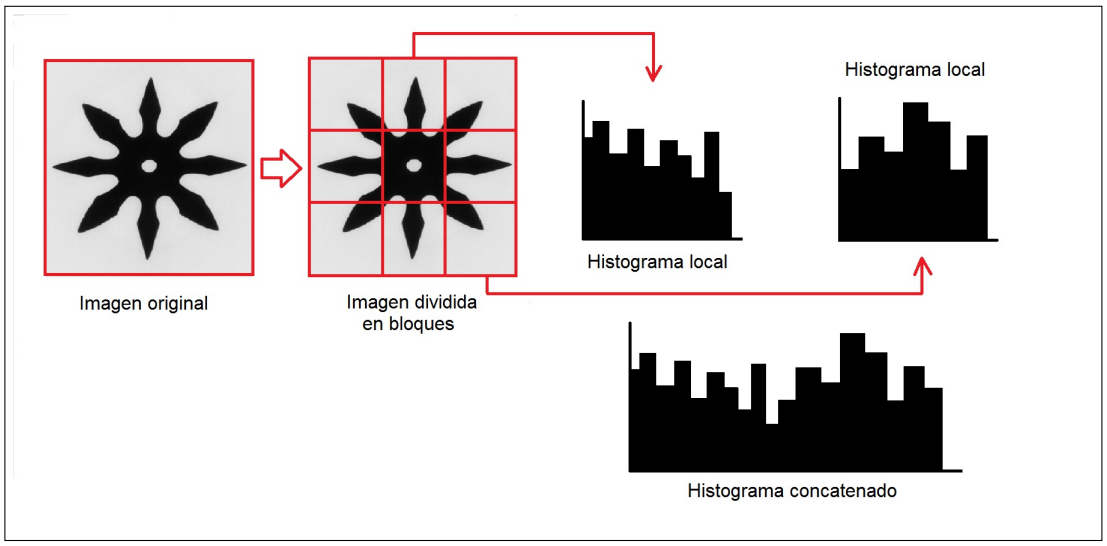

FIGURA 2.5. Esquema que muestra el funcionamiento del LBP. Se divide la imagen en celdas, se crea el código para cada pixel, el cual luego es colocado en un histograma.

2.3. Aprendizaje de Máquina

El aprendizaje de máquina es una rama de la inteligencia artificial que se dedica a desarrollar técnicas que permitan a un computador "aprender". En otras palabras, consiste en crear programas que logren generalizar un comportamiento particular a partir de ejemplos previamente mostrados al programa. En el reconocimiento de patrones se utiliza esta área en la fase de entrenamiento del algoritmo, donde mediante los descriptores y algún algoritmo de aprendizaje de máquina, se entrena para generar un modelo matemático capaz de tomar una decisión frente a un nuevo objeto. Principalmente existen 3 enfoques para realizar este entrenamiento:

(i) **Aprendizaje Supervisado:** Conociendo qué características (u objetos) pertenecen a cada clase se realiza un entrenamiento que genera una función, la cual permite decidir a qué categoría pertenece un nuevo objeto.

(ii) **Aprendizaje no Supervisado**: Sin conocer a qué categoría pertenece cada imagen, se entrena una función que forma grupos de patrones que son similares entre sí. Algunos ejemplos son *clustering*, PCA, etc.

(iii) **Aprendizaje Reforzado**: Se enseña a un software qué acciones debe escoger mediante la repetición y la recompensa a medida que el programa va eligiendo opciones. Se basa en el aprendizaje por conducta.

En este trabajo se utiliza tanto un método de aprendizaje supervisado para el entrenamiento, conocido como Random Forest (Breiman, 2001), además de un método de aprendizaje no supervisado conocido como K-MEANS (Kanungo et al., 2002), el cual es utilizado para generar el modelo de Bolsa de Palabras o *Bag of Words* (Tsai, 2012) que es explicado en la sección 2.4.

2.3.1. K-MEANS

El algoritmo K-MEANS (Kanungo et al., 2002) consiste en un método de *clustering*, o agrupación de características, que toma todos los descriptores e iterativamente ajusta K centros, vale decir, K representantes a los cuales cada *feature* se asemeje a alguno de ellos. La semejanza se obtiene a partir de la distancia Euclidiana entre uno de los K representantes y cualquier *feature*. Matemáticamente hablando, k-means ocupa la ecuación:

$$\underset{S}{\operatorname{argmin}} \Sigma_{i=1}^{k} \Sigma_{x \in S_i} \, d(x, \mu_i) \tag{2.9}$$

donde μ_i representa la media de las características del conjunto S_i y $d(x, \mu_i)$ corresponde a la medida de distancia entre una característica y otra.

Este algoritmo utiliza la distancia Euclidiana, la cual se define como:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \tag{2.10}$$

donde p y q son 2 características diferentes.

Antes de comenzar, K-MEANS elige de forma aleatoria K centroides iniciales, los cuales se irán "moviendo" hasta ajustarse al centroide definitivo. En cada iteración, K-MEANS realiza lo siguiente:

- Para cada característica se calcula la distancia Euclidiana entre ella y todos los centroides.
- Para cada característica se elige el centroide con la menor distancia calculada, y se agrega al grupo perteneciente a este centroide.
- Una vez obtenidos los grupos correspondientes se calcula el promedio de todas las caracteristicas de un grupo. Este promedio pasa a ser el nuevo representante del grupo.
- Se repite la iteración con los nuevos centroides hasta que éstos no cambien (o su cambio sea despreciable), tras lo cual, el algoritmo termina.

En la figura 2.6 se muestra una representación gráfica de k-means, donde a la izquierda vemos un conjunto de características, representadas como puntos de color azul, mientras que a la derecha vemos los centros encontrados, donde cada característica está pintada de un color, que muestra a qué grupo pertenece.

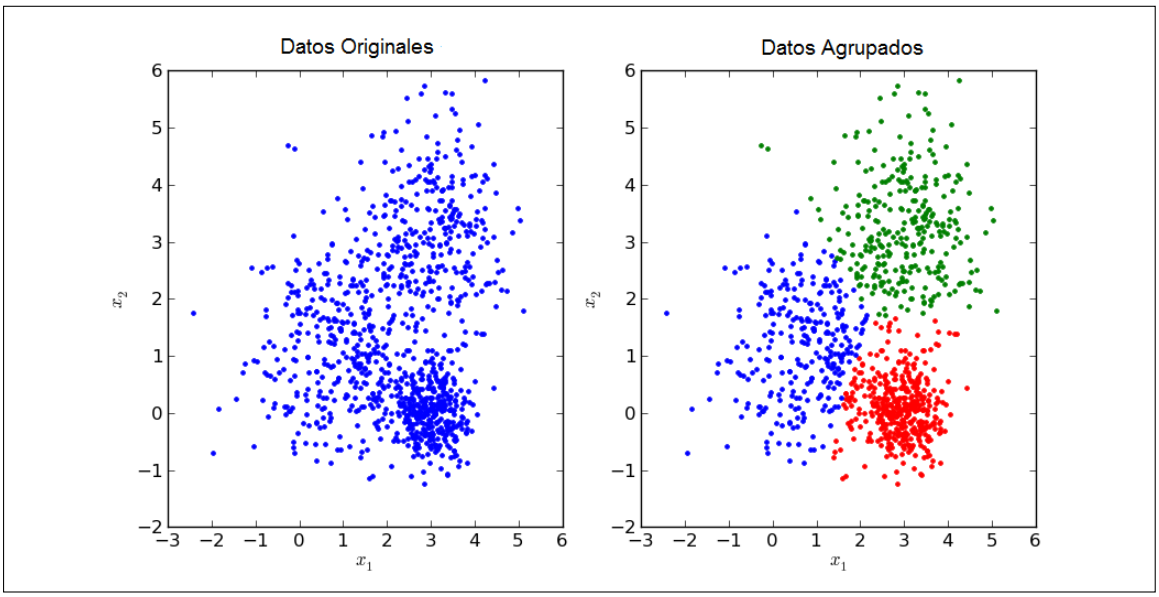

FIGURA 2.6. Ejemplo de k-means con K = 3, donde cada observación se encuentra en un espacio 2D, donde x_1 y x_2 son las coordenadas de cada observación en este espacio.

2.3.2. Random Forest

Random Forest (Breiman, 2001) es un método de aprendizaje supervisado basado en ensambles, es decir, utiliza varios clasificadores simples para realizar la clasificación completa. Cada clasificador simple corresponde a un árbol de decisión, los cuales en una gran cantidad forman un Random Forest.

Un árbol en teoría de grafos corresponde a un grafo acíclico en donde sólo existe un camino entre un vértice u y un vértice v cualquiera. Cada árbol posee una raíz (un nodo inicial que es padre de todos los siguientes nodos), nodos intermedios (que poseen padre e hijos) y hojas (nodos sin hijos).

Un árbol de decisión (Quinlan, 1986) es un modelo de predicción utilizado en inteligencia artificial, que se representa como un árbol en teoría de grafos. La entrada corresponde a objetos u observaciones, las cuales se inician en la raíz del árbol, mientras que la salida corresponde a una decisión final, que es tomada a partir de la observación de entrada y el camino que realizó a la hoja correspondiente. Cada hoja corresponde a una decisión diferente del árbol.

En cada nodo intermedio existe una decisión tomada por un atributo de la observación, donde se elige el camino correspondiente dependiendo de la información del mismo, hasta llegar a una hoja que posee la decisión final (o clasificación final). Un ejemplo de un árbol de decisión se encuentra en la figura 2.7.

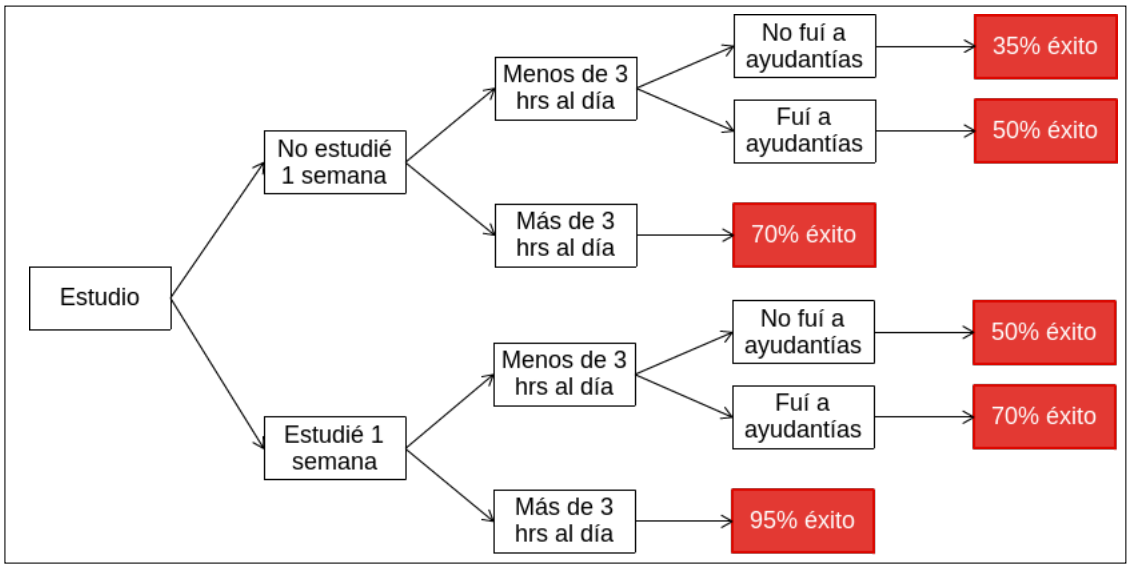

FIGURA 2.7. Esquema que muestra un árbol de decisión. Cada nodo intermedio tiene una decisión diferente (dependiendo del tipo de estudio realizado), mientras que cada hoja posee una predicción del rendimiento de una prueba.

Para la construcción de cada árbol de decisión, se toman todas las características del objeto e iterativamente se va realizando lo siguiente:

- Si todos los atributos son de la misma clase
  - Retornar un nodo hoja con la clase respectiva
- Si todos los atributos tienen el mismo valor para su respectiva clase
  - Retornar un nodo hoja con la clase más común
- De lo contrario:
  - Seleccionar el atributo que “mejor” separa los registros de las diferentes clases
  - Usar este atributo como nodo
  - Dividir el set de entrenamiento de acuerdo a este atributo

- Volver a realizar el mismo procedimiento de forma recursiva para cada rama del árbol

Para encontrar el "mejor" atributo que separe el conjunto, se requiere alguna métrica de homogeneidad. Se entiende por homogeneidad a grupos que sean lo más similares posible. En otras palabras, lo ideal es que cada grupo sólo contenga observaciones de una misma clase.

Existen varias métricas de homogeneidad, donde las más usadas son la ganancia de información (Kullback, 1987) y la impunidad de Gini (Gini, 1912). Para este trabajo se utilizó la segunda. La impunidad de Gini corresponde a la probabilidad de qué tan seguido una elección aleatoria de una característica será incorrectamente clasificado de acuerdo a las etiquetas de las características del subset. Esta métrica se calcula según la siguiente ecuación:

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i). \tag{2.11}$$

donde f_i corresponde a la característica f siendo clasificada como clase i. La ecuación muestra la suma de las probabilidades de la clasificación correcta multiplicada por la clasificación incorrecta. Se escoge la característica que posea una menor impunidad de Gini para cada nodo.

Para generar cada árbol del Random Forest, se toman para cada caso un subset aleatorio de observaciones, de modo de que cada árbol trabaje de forma distinta, con características diferentes al resto de los mismos. Con cada subset, se construye un árbol de decisión utilizando el método explicado anteriormente. Cada árbol construído es diferente, puesto que ninguno utiliza el mismo subset, por lo que sus ramas y mejores atributos fueron obtenidos con la información disponible, y no con la información total de observaciones. Luego, al momento de realizar testing a una nueva observación, cada árbol de decisión mostrará una decisión según la construcción que tuvo y las características iniciales que recibió. Para decidir la clasificación final, se reúnen todas las decisiones finales de cada uno de los árboles de decisión y por medio de alguna métrica se toma la decisión final. La métrica más utilizada es la mayoría de votos,

vale decir, la clase que tenga una mayor cantidad de árboles que la eligió, será la clase del objeto nuevo a clasificar.

Algunas de las ventajas de Random Forest son:

- Es relativamente robusto a los *outliers* y al ruido.
- Es eficiente y rápido en bases de datos grandes.
- Puede detectar las interacciones entre cada variable, sin excluir ninguna.
- Permite encontrar patrones globales usando todas las características.

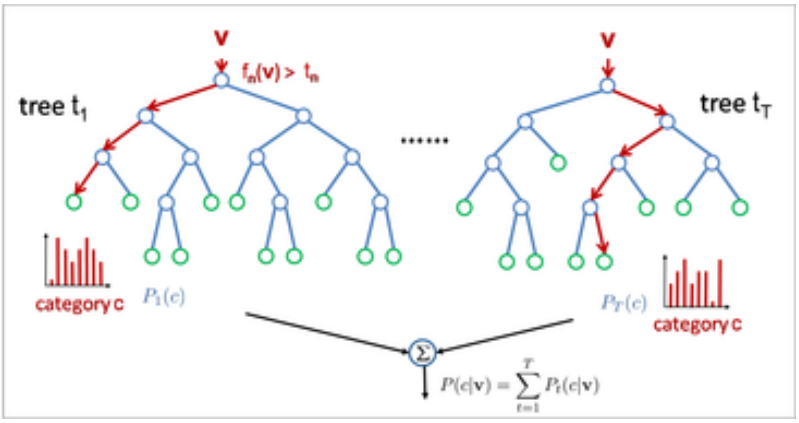

FIGURA 2.8. Esquema de la toma de decisión final de un Random Forest. Se observan 2 árboles con una determinada decisión en sus hojas, para luego ser unidos mediante mayoría de votos para la decisión final.
Fuente: *ICCV Tutorial - Boosting and Random Forest for Visual Recognition*, 2015

2.4. Bag of Words

La bolsa de palabras o Bag of Words (Tsai, 2012) es una técnica utilizada para simplificar la representación de los objetos. Principalmente se utiliza para la representación de documentos a través de las palabras que los constituyen. Para el caso de las imágenes se utilizan "palabras visuales", es decir, descriptores o partes de la imagen que expresan la forma y texturas de esta, y que se utilizan como los representantes de cada grupo.

Para la construcción del Bag of Words, lo primero que se necesita realizar es la extracción de los descriptores de los objetos de interés, ya sea imágenes, palabras, etc. En este trabajo se utilizaron los descriptores SIFT y LBP anteriormente descritos.

Una vez obtenidos los descriptores, se procede a construir un vocabulario o *codebook*. Se entiende por vocabulario a un conjunto de objetos divididos en grupos, donde cada objeto es agrupado de acuerdo a su similitud con los otros para simplificar la representación. En otras palabras, los objetos que pertenecen a un mismo grupo se usan como si fueran iguales.

Por ejemplo, si lo llevamos a un ejemplo con palabras, tenemos lo siguiente: las palabras "conversación", "admiración" y "descripción" pertenecen al grupo de palabra que terminan en "ción", mientras que las palabras "florecer", "florero" y "florista" pertenecen al grupo que contiene la misma raíz "flor". Para el caso de las palabras visuales, el concepto es el mismo; por ejemplo, pueden pertenecer al grupo de "esquinas", "colores fríos" o algún grupo más complejo como por ejemplo, puntos de interés de la imagen, que no se aprecian a simple vista.

Para construir el *codebook* se utiliza algún método de *clustering*, como por ejemplo el algoritmo K-MEANS explicado anteriormente. Estos métodos dividen los descriptores en grupos, los cuales poseen la característica de tener similitud con el vecino más cercano. El conjunto de estos *clusters* forma el *codebook* buscado.

Cada *cluster* representa una bolsa de palabras, o Bag of Word, donde los descriptores pertenecientes a cada *cluster* corresponden a las palabras visuales de la bolsa y cada centroide corresponde al representante (o palabra representativa) de la misma. El representante permite simplificar la visualización de cada bolsa, ya que cada palabra visual se reemplaza con su representante, utilizando una menor cantidad de descriptores que el caso original. Un ejemplo visual de la representación de una bolsa de palabras se puede observar en la figura 2.9

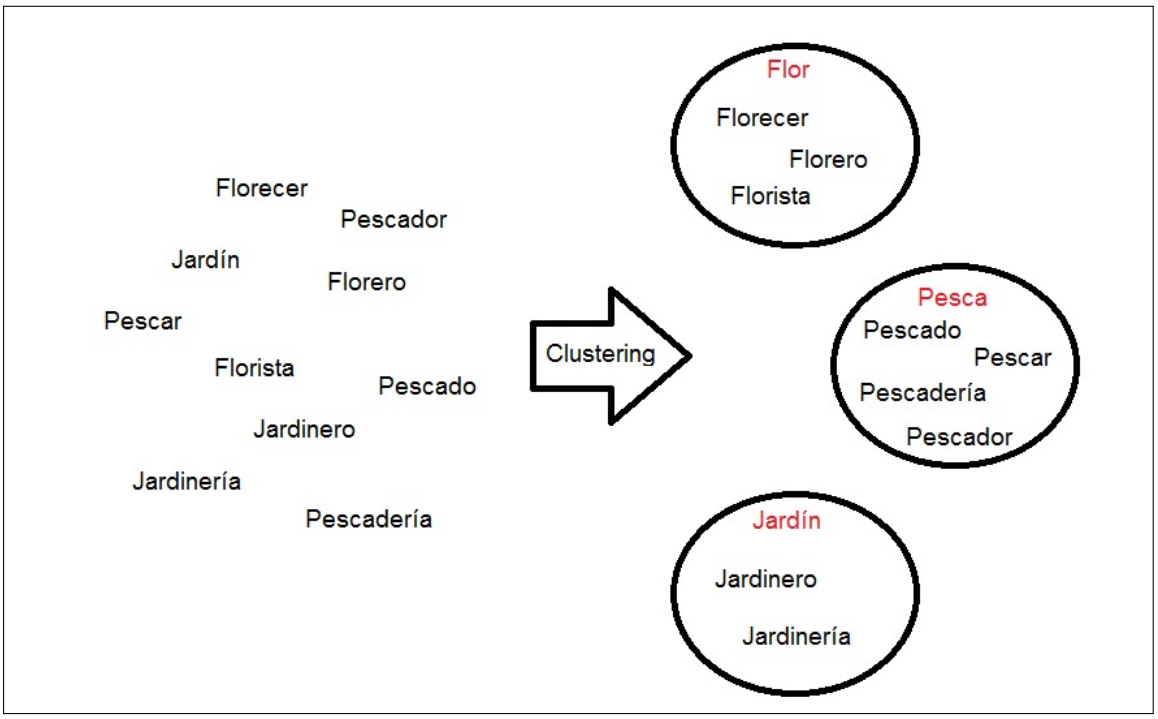

FIGURA 2.9. Dibujo donde se muestra la representación de una bolsa de palabras. A la izquierda, varias palabras sin orden. A la derecha, cada bolsa de palabras formada luego del *clustering*, donde cada representante se encuentra escrito en rojo.

El paso final para este procedimiento, es buscar una forma de representar una nueva imagen según las Bag of Words anteriormente creadas. Para esto, se utiliza un histograma. En el modelo Bag of Words, cada bin del histograma corresponde a cada representante de cada bolsa de palabras.

Para cada nueva imagen que se quiere representar, primero se le extraen sus descriptores, para luego comparar cada descriptor con los representantes de cada Bag of Words. Para la comparación, existen variadas métricas, donde generalmente se utilizan las métricas de distancia. En este trabajo se utiliza la distancia Euclidiana, descrita previamente en la ecuación (2.10).

Luego se reemplaza cada descriptor por el representante de la bolsa de palabras más cercano (vale decir, con el que posee la menor distancia). Con esto, se procede a construir el histograma de la imagen, donde se cuenta la cantidad de veces que cada descriptor aparece en la imagen y se construye el gráfico respectivo. Con esto, la imagen queda representada según

el modelo Bag of Words. Un ejemplo gráfico de este procedimiento se observa en la figura 2.10.

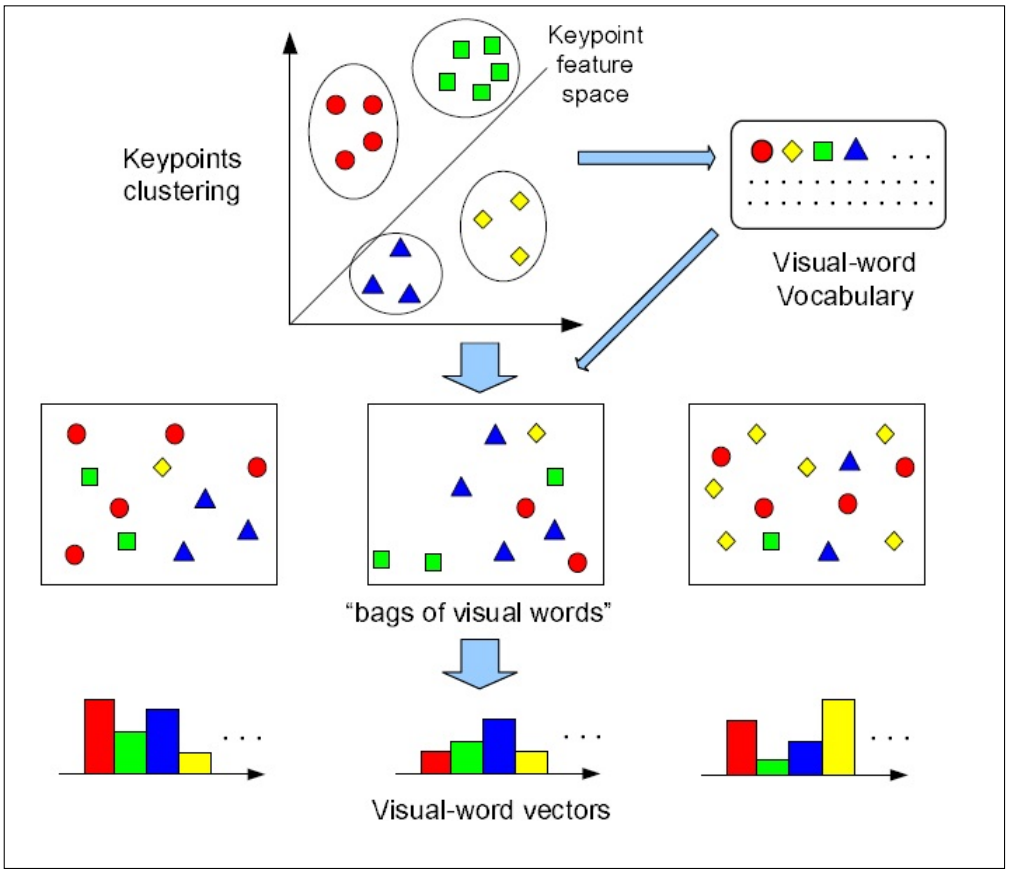

FIGURA 2.10. Esquema del procedimiento de creación de las Bag of Words. Arriba a la izquierda se realiza el *clustering* de las observaciones iniciales. Arriba a la derecha se encuentran los representantes de cada grupo. En la fila central se encuentra cada nueva imagen esquematizada con los representantes anteriores. En la fila de abajo se encuentra el histograma para cada una de estas nuevas imágenes.
Fuente: *Tsai (2012)*

2.5. Evaluación de desempeño de un clasificador

Para evaluar un clasificador existen diversas técnicas, las cuales utilizan una gran variedad de métricas de desempeño diferentes. Se le conoce como métrica de desempeño a una forma de cuantificar los resultados obtenidos, de modo de obtener números que sean comparables entre sí.

En este trabajo se utilizan 2 métricas de desempeño: lo que se conoce como matriz de confusión y un gráfico de Precision-Recall, que se construye a partir de lo anterior. Además, en esta sección se explica un problema habitual en estos trabajos, conocido como sobreajuste, y cómo solucionarlo. Estos conceptos se detallan a continuación.

2.5.1. Matriz de confusión

Esta es una matriz que permite comparar 2 principios fundamentales: la clase a la cual un objeto pertenece vs la clasificación que le dio el clasificador creado. Por ejemplo, si tenemos un clasificador que separe cuchillos, armas y tijeras, la matriz de confusión permite observar cuántos objetos fueron clasificados correctamente y cuantos incorrectamente, como se ve en la figura 2.1. El ideal se logra maximizando la diagonal.

TABLE 2.1. Ejemplo de matriz de confusión

		Clasificación		
		Cuchillo	Arma	Tijera
Real	Cuchillo	90	10	0
	Arma	3	95	2
	Tijera	4	7	89

Para este trabajo se utilizaron 2 clases: pertenencia y no-pertenencia de un objeto, por lo cual esta matriz consiste en una matriz de T_{2x2} que se ve como la figura 2.2

TABLE 2.2. Matriz de confusión para 2 clases

		Clasificacion	
		Pertenece	No pertenece
Real	Pertenece	TP	FN
	No Pertenece	FP	TN

Donde:

- TP o True Positive: cantidad de detecciones correctas, vale decir, cuántos objetos que pertenecían a la clase fueron clasificados correctamente.

- TN o True Negative: cantidad de objetos que no pertenecen a la clase, y que fueron clasificados correctamente.
- FP o False Positive: cantidad de objetos que, sin pertenecer a la clase, fueron clasificadas dentro de la misma. Por ejemplo, una tijera se clasificó como cuchillo.
- FN o False Negative: cantidad de objetos que pertenecían a la clase, pero que fueron clasificados como que no pertenecían a la misma. Por ejemplo, una pistola que es clasificada como no-arma.

El ideal, como en el caso anterior, se logra maximizando la diagonal, vale decir, cuando la cantidad de TP y TN es máxima y hay una cantidad de cero FP y FN. Además cabe destacar lo siguiente:

- TP + FN: es la cantidad real de objetos que el sistema debió detectar
- TP + FP: es la cantidad total de detecciones realizadas por el sistema

Por último, en este trabajo se utilizan métricas de porcentaje con respecto a la cantidad de TP, TN, FP y FN. Este porcentaje se muestra como un número entre 0 y 1. Las métricas mencionadas se conocen como True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR) y False Negative Rate (FNR), las cuales se definen de la siguiente manera:

- TPR = TP / (TP + FN)
- TNR = TN / (TN + FP)
- FPR = FP / (TN + FP)
- FNR = FN / (TP + FN)

2.5.2. Gráfico de Precision-Recall

El gráfico de Precision-Recall es un gráfico que se construye a partir de los parámetros anteriores. *Precision* corresponde a la fracción de instancias (en este caso objetos) recuperadas que corresponden a la clase, mientras que *Recall* corresponde a la fracción de objetos de la clase que son recuperados. Su objetivo es obtener una métrica visual que muestre cuántos objetos son detectados correctamente, dada una cantidad que se quiere recuperar, o en otras

palabras, dado un *recall* específico, cual es el *precision* del mismo. Un ejemplo visual de este gráfico se muestra en la figura 2.11.

Tanto el *precision* como el *recall* se calculan de la siguiente manera:

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

donde TP, TN, FP y FN corresponden a los explicados en la sección 2.5.1

Para la construcción del gráfico, se utilizan los porcentajes de creencia de un clasificador o *scores*, además de los *labels* reales de las imágenes. Con esto, se va calculando un threshold que sirve como medida de separación entre las clases: mientras más pequeño este valor, menor *recall*. Esto permite obtener todos los valores del gráfico.

Hay que considerar que no todos los gráficos de Precision-Recall son decrecientes, ya que los parámetros no son dependientes entre sí. Por la definición de *precision*, ésta es dependiente del threshold utilizado, ya que por ejemplo, si el threshold es muy alto, los nuevos resultados pueden ser todos TP, aumentando la precisión; si el threshold es bajo, ésto introduce FP, que bajan el valor de *precision*. Sin embargo, *recall* no es dependiente del threshold, lo que puede provocar fluctuaciones de *precision*.

Lo importante a considerar es el área bajo la curva de ambos y el punto de la esquina superior izquierda (que se considera el caso ideal). A mayor área, mayor *precision* y *recall* respectivamente.

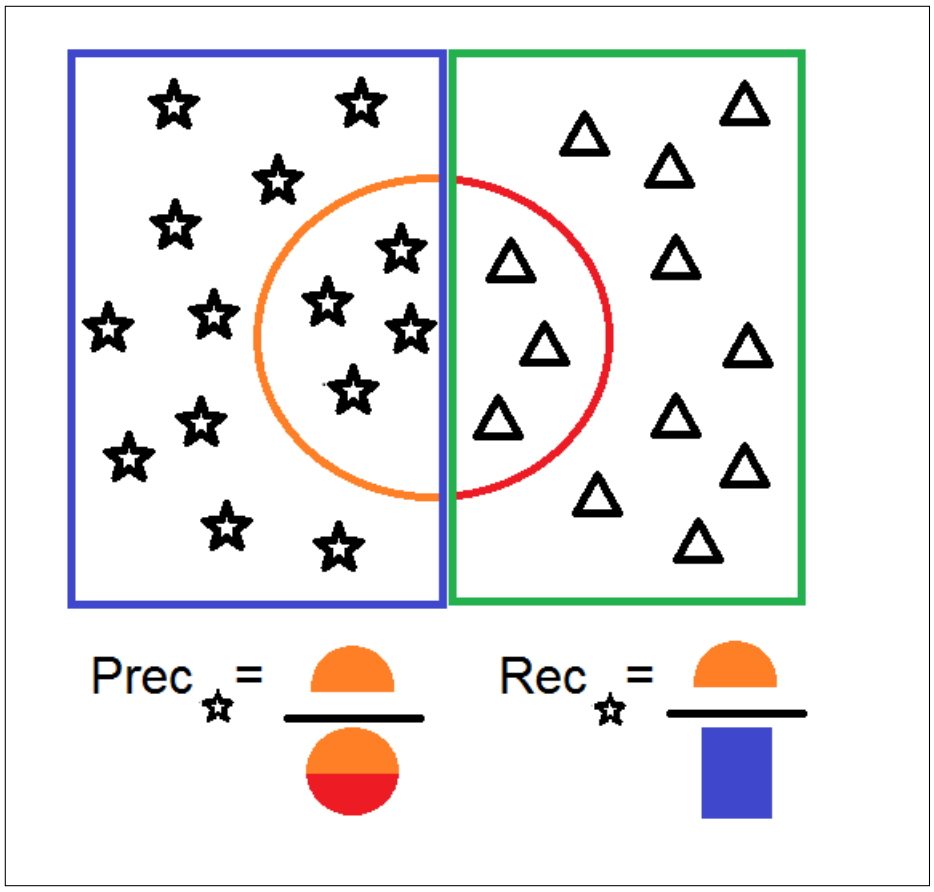

FIGURA 2.11. Esquema donde se muestra cada uno de los objetos de la muestra, y su correspondiente clasificación. Arriba: Visión de los TP, TN, FP y FN de una muestra, Abajo: cálculo de *precision* y *recall* de las estrellas visto esquemáticamente.
Fuente: *Wikipedia - Precision and Recall*, 2015

2.5.3. Sobreajuste

Un problema habitual en los clasificadores es el sobre entrenamiento o sobreajuste, en donde el clasificador obtiene un buen resultado sobre las imágenes utilizadas en el entrenamiento, pero al momento de realizar cambios en el conjunto de imágenes, o agregar nuevas imágenes al mismo, éstas son clasificadas incorrectamente. Esto provoca que el clasificador se **sesgue sólo a las imágenes de entrenamiento**.

Para evitar este problema, se utilizan por lo menos 2 sets de imágenes: el set de entrenamiento y el set de testing. El primer set se utiliza, como su nombre lo indica, para entrenar el clasificador y crear el modelo adecuado, mientras que el segundo se utiliza para comprobar

su desempeño. Las imágenes de ambos sets deben ser diferentes, ya que de lo contrario, el clasificador se sesga siempre a las mismas imágenes, provocando sobreajuste. Puede existir más de un set de testing para el mismo clasificador.

El comportamiento habitual de un clasificador es que logre un desempeño de casi un 100% si se usa el set de entrenamineto en el Testing, mientras que al utilizar el set de testing en el mismo, su desempeño debería bajar. El ideal es que con ambos sets el desempeño sea lo más alto posible.

Para este trabajo se utilizó un set de validación, con el fin de evitar el sobreajuste. Este set de validación consiste en un subset de los datos de entrenamiento, mucho más pequeño que los datos totales. Con este set se validaron todos los parámetros posibles del algoritmo, buscando la media más alta y el mejor desempeño posible. Posteriormente, se utilizaron estos mismos parámetros en el set total de imágenes de prueba.

3. MÉTODO PROPUESTO

El método propuesto se basa en un algoritmo básico de reconocimiento de patrones, cuyo funcionamiento se explicó en la sección 2.2. El diagrama simple del método propuesto se muestra en la figura 3.1 el cual consta de 3 etapas (recuadros violeta): Training, donde se pretende construir 3 clasificadores capaces de predecir resultados; Testing, donde se obtienen las predicciones de cada clasificador entrenado usando su modelo matemático y las Bags of Words obtenidas; y Validación de Scores, donde se compara el resultado de cada clasificador particular, obteniendo una decisión final para cada objeto. Tanto el Training Set como el Testing Set se componen de imágenes de objetos, los cuales corresponden a las clases que se utilizaron en este trabajo. Estas clases son: Razor (hojas de afeitar), Shuriken, Gun (pistolas) y Other, correspondiente a objetos no mencionados anteriormente. De ahora en adelante se usarán los términos Razor, Shuriken, Gun y Other para referirise a cada una de las clases a las que pertenecen los objetos mencionados anteriormente.

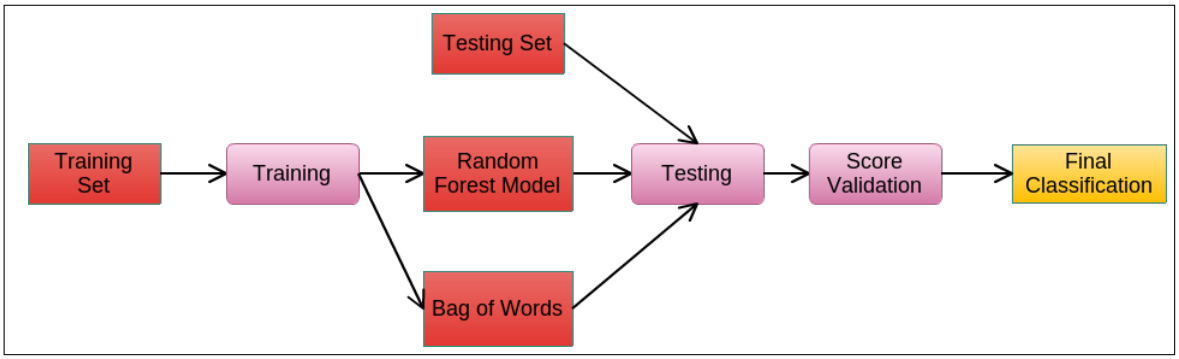

FIGURA 3.1. Esquema básico de la metodología propuesta, basado en un esquema general de un algoritmo de reconocimiento de patrones.

En las siguientes secciones se explicará el funcionamiento de cada una de las 3 grandes etapas de la metodología propuesta. Posteriormente, una cuarta sección explicará el ambiente de desarrollo que se utilizó para desarrollar este trabajo.

3.1. Training

Durante la fase de Training, se entrenan 3 clasificadores, cada uno especializado en alguna de las 3 clases principales utilizadas (Gun, Shuriken y Razor), sin considerar la clase Other. El diagrama de funcionamiento para cualquiera de los 3 clasificadores se muestra en la figura 4.1. Las principales fases de la etapa de Training corresponden a: preprocesamiento de las imágenes, obtención de descriptores, obtención de centroides con K-MEANS, creacion de los histogramas de Bag of Words y entrenamiento usando Random Forest.

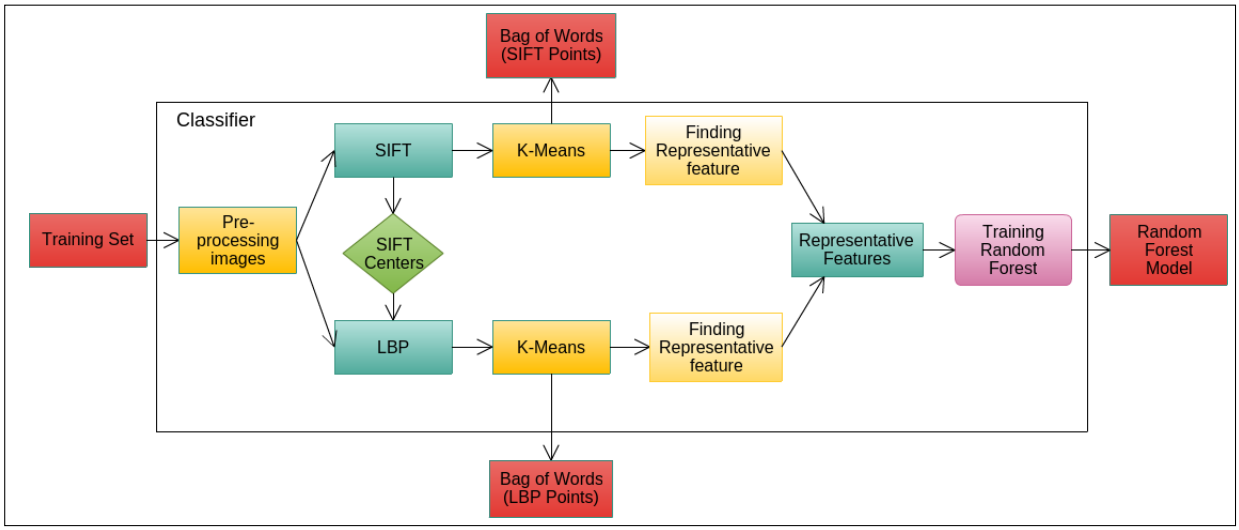

FIGURA 3.2. Diagrama de la etapa de Training del algoritmo propuesto.

Al finalizar la fase de entrenamiento, por cada clasificador se obtienen 3 archivos (recuadros en rojo), que corrresponden a:

- El modelo matemático que permite predecir la clasificación de nuevas imágenes.
- La bolsa de palabras, para cada una de las *features* utilizadas en la producción del modelo. En este caso una bolsa para SIFT y otra para LBP.

Estos archivos son de suma importancia, ya que son los que se utilizan en la etapa de Testing. A continuación se describe cada una de las fases del Training en detalle.

3.1.1. Preprocesamiento de las imágenes

El primer paso del algoritmo es un pre-procesamiento de las imágenes, en donde a todas las imágenes de entrada se le aplica un filtro Gaussiano, explicado en la sección 2.1.1. El resto del algoritmo trabaja con estas imágenes procesadas y no con las originales.

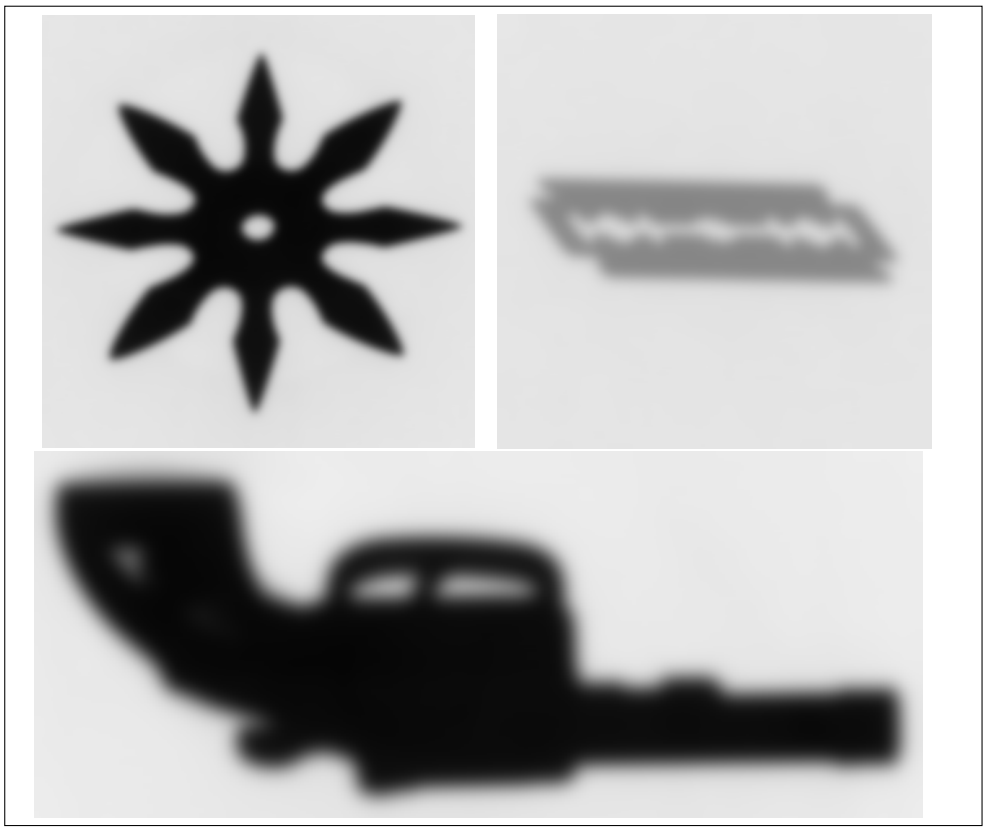

FIGURA 3.3. Ejemplos de imágenes del set de entrenamiento aplicándole filtro Gaussiano con $\sigma = 6$. Arriba: Clases Shuriken y Razor, Abajo: Clase Gun

La finalidad del filtro Gaussiano es suavizar las partes relevantes del objeto, permitiéndole al descriptor SIFT obtener estos puntos. Por ejemplo, para el clasificador de Shuriken, algunos puntos relevantes son las puntas y los espacios entre las mismas. De no realizar este filtro, el brusco cambio de intensidad de gris (entre el fondo y la shuriken) se vuelve intolerante a la ubicación, ya que de estar en otra posición, el descriptor SIFT no es capaz de reconocerlo. No obstante, este filtro no actúa de igual manera para cada objeto. Para el caso de las hojas de afeitar, al no poseer cambios bruscos de intensidad (ya que sus tonos de gris

son más claros) no se requiere un alto valor de σ para detectar estos puntos. Para encontrar un valor de σ adecuado para cada objeto, se realizó un ajuste de parámetros para cada clasificador particular, donde los valores de σ variaban entre cada uno. Esto se muestra en detalle en la sección 4.

Para el ajuste de parámetros, en este trabajo se aplicó un valor de σ entre 0 y 8, donde 0 equivale a no realizar el filtro a la imagen.

3.1.2. Obtención de descriptores SIFT y LBP

El siguiente paso que se muestra en la figura 4.1 es la obtención de descriptores SIFT y LBP. En primer lugar se procedió a obtener SIFT, para esto, a cada imagen preprocesada en cada clasificador, se le realizó el algoritmo de la sección 2.2.2.

Al finalizar el algoritmo anterior, se obtuvo para cada imagen una cantidad variada de puntos SIFT, o *keypoints*, entre 10 y 200, dependiendo de la imagen. A cada *keypoint* se le calculó su vector descriptor, de largo 128. Además, por cada *keypoint* se guardó su ubicación (x, y) dentro de la imagen y su escala como datos adicionales. Algunos ejemplos de los *keypoints* obtenidos se muestran en la figura 3.4

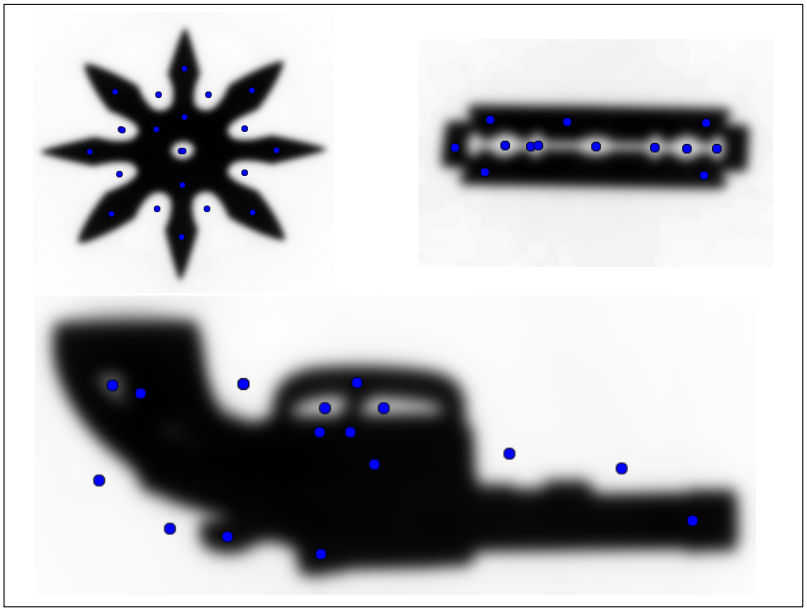

FIGURA 3.4. Ejemplos de imágenes del set de entrenamiento con sus respectivos *keypoints* obtenidos. Arriba: Clases Shuriken y Razor, Abajo: Clase Gun

Cada vector descriptor SIFT se agregó a 2 listas diferentes: en la primera se guardaron todos los vectores descriptores, mientras que en la segunda se guardaron separadamente los vectores de cada imagen. Esto se realizó dado que la primera lista se utilizó en el paso siguiente, mientras que la segunda se utilizó en la sección 3.1.4

Una vez obtenidos los keypoints, se tomaron todas las ubicaciones (x, y) y la escala s de cada *keypoint* y alrededor de este punto, se creó una nueva imagen de tamaño $2s \times 2s$, cuyo centro es el punto (x, y) de la imagen original. Un ejemplo de esta nueva imagen se muestra en la figura 3.5.

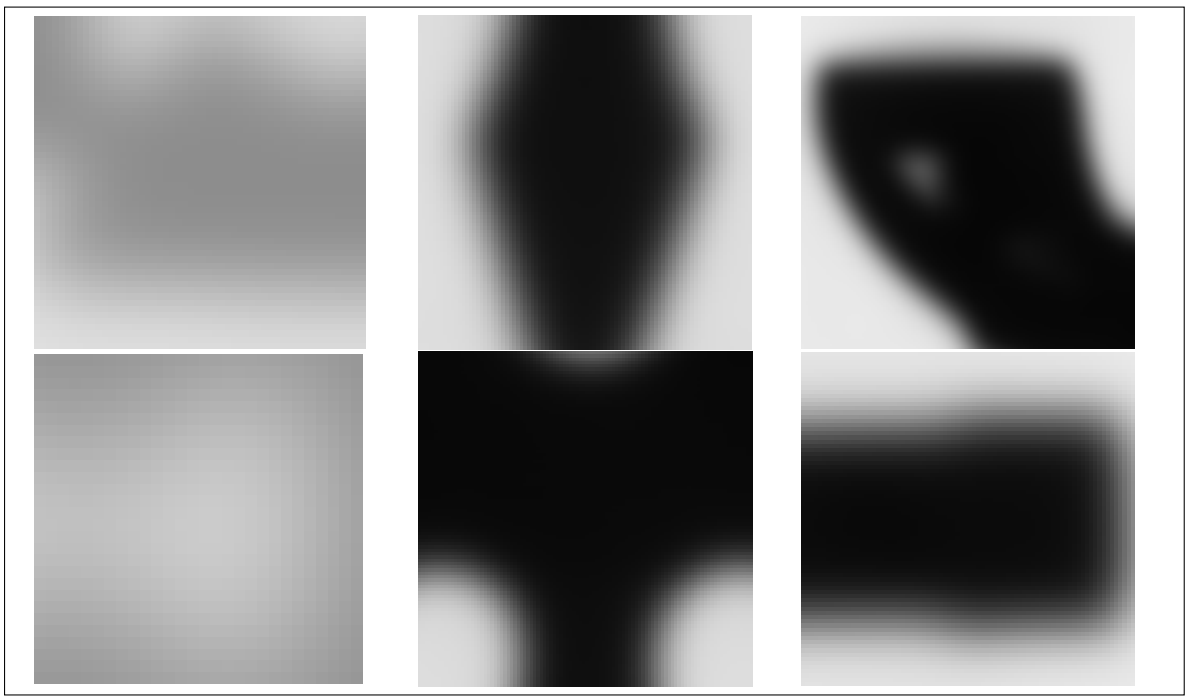

FIGURA 3.5. Ejemplos de patches de imágenes del set de entrenamiento para posteriormente aplicar LBP. Por columna de izquierda a derecha: Clases Razor, Shuriken y Gun

A esta nueva imagen, se le aplicó el algoritmo de LBP descrito en la sección 2.2.3, con lo que se obtuvo por cada imagen un vector de largo 36 que representa el histograma de la imagen correspondiente normalizado. Por cada imagen se obtuvo una cantidad de vectores LBP igual que su cantidad de *keypoints*.

De la misma forma que los descriptores SIFT, éstos se guardaron en 2 listas: la primera que contiene todos los vectores LBP y la segunda que los guarda separadamente por imagen.

3.1.3. K-Means sobre SIFT y LBP

Al finalizar la fase anterior, se toman de forma separada las 2 listas que contienen a los descriptores de SIFT y LBP respectivamente, procediendo a realizar un K-MEANS a cada uno, también de forma separada, algoritmo descrito en la sección 2.3.1, utilizando los valores de K correspondientes a cada uno.

Lo primero que se verifica es que el número de muestras no exceda el valor del K, sino no se puede ejecutar el algoritmo. Luego se procede a realizar K-MEANS para los descriptores SIFT y para los descriptores LBP, obteniendo diferentes centroides para cada uno.

El resultado por *feature* son 2 listas: la primera contiene los K centroides encontrados luego de las iteraciones correspondientes, mientras que la segunda contiene los *labels* a los cuales pertenece cada centroide del *cluster*. Cada ubicación del centroide en la primera lista es concordante con la ubicación de su respectivo label en la segunda. Estas listas son guardados en un archivo (Bag of words en rojo en la figura 4.1), ya que luego son utilizadas tanta en la fase siguiente como en la etapa del Testing.

3.1.4. Creación histogramas de Bag of Words: Descriptores SIFT y LBP

Una vez que tenemos los centroides, se procede a construir la bolsa de palabras para cada imagen. Este procedimiento se realiza de forma separada para SIFT y LBP, utilizando sus respectivos centroides encontrados en la fase anterior. Se realiza lo siguiente:

Para cada descriptor por imagen:

- Se compara el descriptor con todos los centroides, mediante distancia Euclidiana, descrita en la ecuación 2.10.
- Cada descriptor se reemplaza con el centroide más cercano, es decir, con el que tiene una menor distancia Euclidiana.

Una vez reemplazado cada descriptor se procede a crear el histograma. Este histograma pasa a ser la representación final de las características de cada imagen. Para crear el histograma se realiza lo siguiente:

- Se crea un vector inicializado en 0 para todos sus valores, donde cada posición corresponde a uno de los centroides encontrados. El largo del vector corresponde al total de centroides (vale decir, la suma de los valores K de los descriptores SIFT y de los vectores LBP).
- Se cuentan cada uno de los representantes de cada imagen. Por cada representante se aumenta en 1 el valor en la posición del centroide. Como consecuencia, se forma un vector que representa el histograma de la frecuencia en que aparece cada centroide en la imagen, como se explicó en la sección 2.2.1.
- Se normaliza el vector, vale decir, cada valor en cada posición se divide por el total de características de la imagen.

La finalidad de la normalización es convertir cada una de las cantidades en un decimal equivalente a un porcentaje, lo cual permite la comparación entre diferentes vectores de diferentes imágenes que tienen una variada cantidad de observaciones. De no realizarse en porcentajes, la comparación sería imposible.

Finalmente se crea una matriz de *features*, de tamaño $N \times M$, donde N corresponde al total de imágenes y M corresponde al largo del vector anteriormente creado (vale decir, la suma de los K centroides SIFT y los K centroides LBP). Cada vector anterior se coloca como una fila de esta matriz y corrresponde a las *features* de la imagen.

Esta matriz final será la que se utilizará en la siguiente fase.

3.1.5. Training: Algoritmo Random Forest

En esta etapa se procede a ejecutar el algoritmo descrito en la sección 2.3.2. Para esto se utilizan 2 elementos: la matriz de *features* creada en la fase anterior, y un vector con todos los *labels* de cada imagen. Se entiende por *label* a la clase a la que corresponde cada vector de *features* por imagen, descrita por un número.

Para construir el vector de *labels*, se crea un vector de largo igual al total de imágenes, donde cada posición del vector corresponde al número de la imagen. Por cada clasificador se crea un label de 2 números (0 y 1) donde un 0 corresponde a un objeto que pertenece a la clase, mientras que 1 corresponde a un objeto que no pertenece a la clase. Por ejemplo, de ser el clasificador de Shuriken, todas las imágenes que tienen shurikens poseen un label 0, mientras las que no tienen shurikens poseen label 1.

Una vez obtenidas la matriz de *features* y el vector de *labels*, se entrena el clasificador Random Forest llamando al método *fit* de la clase Random Forest del paquete *Scikit-learn*, pasándole los atributos anteriores . Como ya se explicó, la métrica de homogeneidad utilizada es la impunidad de Gini, cuya ecuación es 2.11. La cantidad de árboles utilizada varió entre 1000, 2000 y 4000 árboles, mientras que el total de *features* correspondía a la suma de los valores K tanto de SIFT como de LBP por imagen. Otro parámetro importante es la cantidad de features aleatorias que se seleccionan en cada split para cada uno de los árboles, la cual en este trabajo se utilizó la raíz cuadrada del total de *features*. No obstante, el algoritmo busca exhaustivamente hasta encontrar un split válido, aunque esto implique inspeccionar más que el total de *features* fijado.

3.2. Testing

El diagrama de la etapa de Testing se muestra en la figura 3.6. De la misma forma que el Training, la etapa de Testing se realiza para cada uno de los 3 clasificadores entrenados de forma separada. La gracia de esta etapa es que es mucho más rápida en comparación al Training dado que no se realiza K-MEANS, que es uno de los procedimientos que más tiempo consume.

Las imágenes utilizadas en esta etapa son diferentes al entrenamiento, las cuales también vienen rotuladas con sus respectivos *labels*. No obstante, los labels se utilizan sólo para comparar el resultado obtenido con el ideal.

Como se puede apreciar en la figura 3.6, el inicio de la etapa de Testing es igual que el de la etapa de Training, ya que se parte con un preprocesamiento y continúa con la obtención

de descriptores SIFT y LBP. Posteriormente se crea el histograma de *features* utilizando la bolsa de palabras obtenida en la etapa anterior. Finalmente se realiza el testing del histograma obtenido anteriormente utilizando el modelo matemático entrenado.

Un punto importante de esta etapa es que se necesitan tanto las bolsas de palabras como el modelo entrenado por cada clasificador en la etapa anterior, los cuales se muestran como cuadrados rojos en el diagrama.

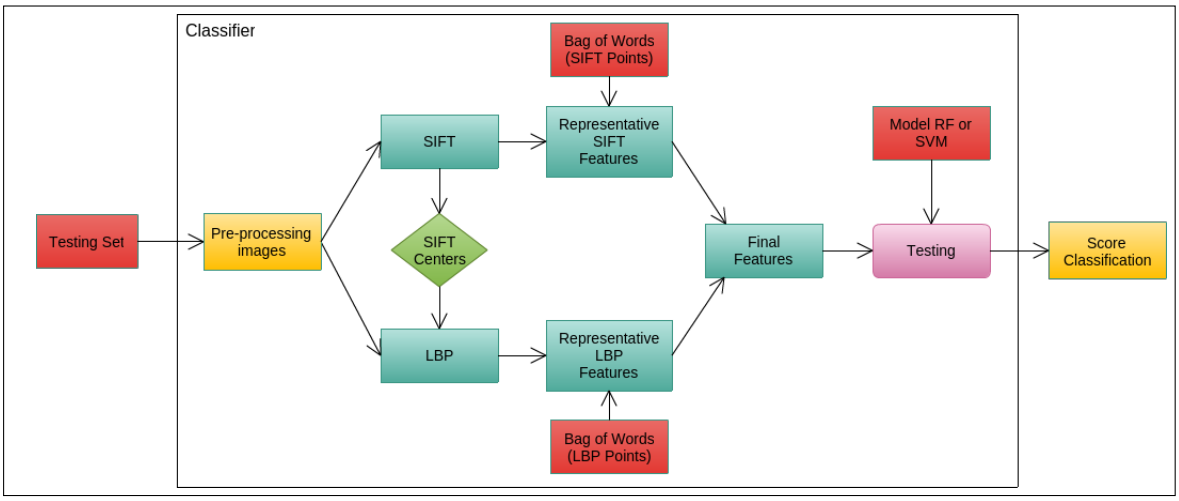

FIGURA 3.6. Diagrama de la etapa de Testing del algoritmo propuesto.

El resultado de esta etapa es un vector del mismo largo que el total de imágenes de Testing. Cada índice del vector es concordante con el número de la imagen. Cada posición del vector con tiene un número (0 o 1), indicando si el algoritmo "piensa" que la imagen correspondiente es o no a la clase que está buscando.

Dado que son 3 clasificadores, el resultado son 3 vectores del mismo tamaño: para las clases Gun, Shuriken y Razor respectivamente. Cada posición equivale a una de las imágenes utilizadas.

A continuación se describe cada fase de la etapa de Testing.

3.2.1. Preprocesamiento de las imágenes

Esta fase es exactamente igual que la del Training, descrita en la sección 3.2.1. El parámetro utilizado para el filtro Gaussiano en cada clasificador es el mismo que el utilizado

en el clasificador correspondiente entrenado anteriormente. Por ejemplo, si el clasificador de Guns utilizó un valor de $\sigma = 4$, el Testing de Guns, también utiliza un valor de $\sigma = 4$ en esta fase.

3.2.2. Obtención de descriptores SIFT y LBP

Esta fase se realiza de la misma forma que en la etapa de Training, descrita en la sección 3.1.2. Al igual que en el preprocesamiento, se utilizan los mismos valores de K tanto para los descriptores SIFT como LBP usados en la etapa de Training.

3.2.3. Creación histogramas del Bag of Words: Descriptores SIFT y LBP

En esta etapa se utilizan las bolsas de palabras de SIFT y LBP obtenidas en la etapa de Training luego del K-MEANS descrito en 2.3.1. Estas bolsas de palabras aparecen como un recuadro rojo en la figura 3.6.

Utilizando estos archivos, se toma cada descriptor y éste se reemplaza con su respectivo representante, utilizando la distancia Euclidiana.

Finalmente se crea la matriz de *features*, de la misma forma que en la etapa de Training.

3.2.4. Testing: Random Forest

En la etapa de Testing hay un cambio fundamental en comparación al Training. Lo primero que se realiza es un Testing utilizando la matriz de *features* de la etapa anterior y el modelo entrenado obtenido en el Training (que aparece en rojo en la figura 3.6).

Para el test se realiza lo siguiente:

Por cada descriptor por imagen (cada fila de la matriz):

- Se introduce el descriptor en cada uno de los n árboles entrenados
- Dependiendo del contenido del descriptor, cada árbol crea una ruta diferente para el mismo descriptor.
- Al final del proceso, por cada árbol se obtiene una clasificación (o decisión), dependiendo de la construcción del árbol.

- Utilizando una métrica de unión de árboles, se obtiene una decisión final.

La métrica utilizada es la siguiente:

- Por cada árbol, se entrega la cantidad de hojas que lo clasifican como objeto y como no-objeto.
- Con todos los resultados, se entrega el porcentaje de hojas que lo clasifican como objeto y no-objeto.

Esta métrica sirve como una medida de "qué tan seguro está el clasificador de lo que cree", vale decir, el nivel de certeza de su decisión.

Lo que se obtiene de esta etapa es un vector de ancho 2 y largo el total de imágenes. Por cada imagen, se obtienen 2 porcentajes donde el primero es el porcentaje de certeza de clasificación como objeto, mientras que el segundo es el porcentaje de certeza de clasificación como no-objeto.

Para el estudio de cada clasificador particular basta con finalizar esta etapa, donde se toma como clasificación final el porcentaje mayor (vale decir, el que es mayor al 50%). Con esto, se obtiene un vector **por clasificador** del largo igual al total de imágenes, que contiene la clasificación para cada una de ellas según la decisión que tomó el algoritmo. En caso de no requerir una comparación entre clasificadores, este vector pasa directo a la sección 3.3.2, sin realizar la comparación de Scores.

3.3. Score Validation

La tercera etapa del algoritmo es la validación de Scores. En esta etapa interac-túan los 3 clasificadores anteriormente entrenados y testeados, como se muestra en la figura 3.8. El resultado final de esta etapa es un vector que posee las clasificaciones finales para cada imagen probada en el set de Testing. Cabe destacar que esta etapa no es completamente necesaria para que funcione el algoritmo, ya que cada clasificador puede trabajar de forma independiente. Sólo se realiza este proceso en la tercera parte de los resultados y experimentos, cuando se estudia el desempeño general del algoritmo.

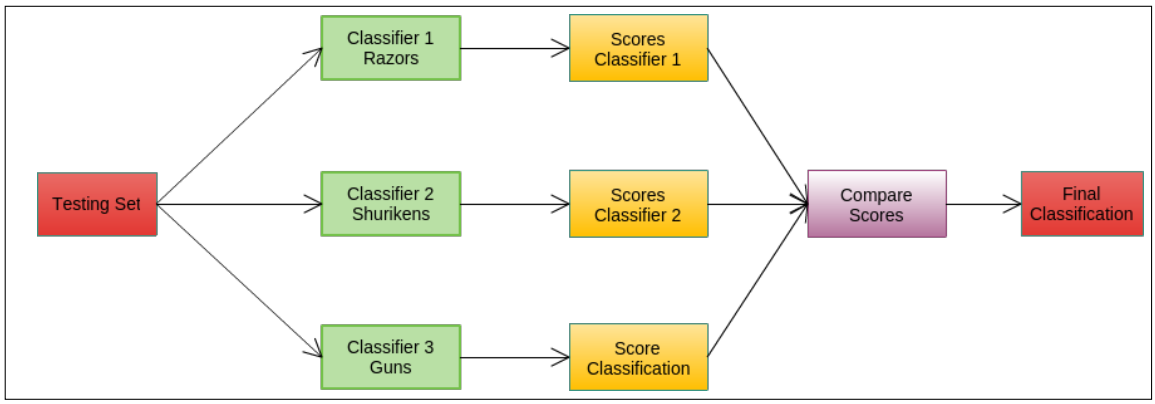

FIGURA 3.7. Diagrama de la etapa de comparación de Scores del algoritmo propuesto.

3.3.1. Comparación de scores

Primero que nada hay que tener en cuenta que este procedimiento no se realiza a todas las imágenes, sino solamente a las imágenes donde existen **discrepancia en la clasificación**, vale decir, cuando una misma imagen es clasificada al mismo tiempo como 2 (o más) objetos diferentes. Por ejemplo, si una misma imagen es clasificada como clase Razor por un clasificador, pero como clase Shuriken por otro, significa que hay discrepancia y se debe buscar una forma de tomar una decisión final. Para el desempate, se utiliza una métrica que, en simples palabras, lo que busca es clasificar la imagen según el clasificador que tiene más certeza de su decisión. Esta métrica se describe a continuación.

Cuando existen casos como el anterior, lo que se hace es comparar los scores de ambos clasificadores, los que se obtuvieron en la fase final del Testing. Estos scores se comparan, buscando el mayor. Finalmente, el clasificador que posea el mayor score, será el que clasifique a la imagen, ya que significa que su certeza para clasificar el objeto es mayor al otro. Por ejemplo, si el primer clasificador cree en un 60% que la imagen es de la clase Razor, mientras que el segundo cree en un 65% que la imagen es de la clase Shuriken, entonces la clasificación final para la imagen será la clase Shuriken.

En el caso de haber ocurrido empate, se hubieran comparado la cantidad de hojas que lo clasificaron. Sin embargo, dado que se tomaron hasta 4 decimales significativos para la

comparación, y la cantidad de casos de ocurrencia no fue muy alta, no fue necesario realizar este procedimiento.

Los casos en los cuales **no** fue necesario realizar la comparación (y por ende, cambiar la clasificación final) fueron:

- Cuando un clasificador clasificó la imagen como objeto y los otros 2 lo hicieron como no-objeto. Por ejemplo: el clasificador de Razor dijo que si, mientras que los de Shuriken y Gun dijeron que no.
- Cuando ninguno de los 3 clasificadores dijo que la imagen era un objeto que clasificaba. Esto quiere decir que la imagen testeada no correspondía ni a la clase Razor, Shuriken ni Gun, según el algoritmo.

3.3.2. Evaluación y matriz de confusión Final

Para la clasificación final, basta con tomar el vector de los labels reales por imagen y el vector con los labels clasificados por el algoritmo y compararlos. El resultado se muestra en una matriz de confusión, como se observa en la figura 3.8.

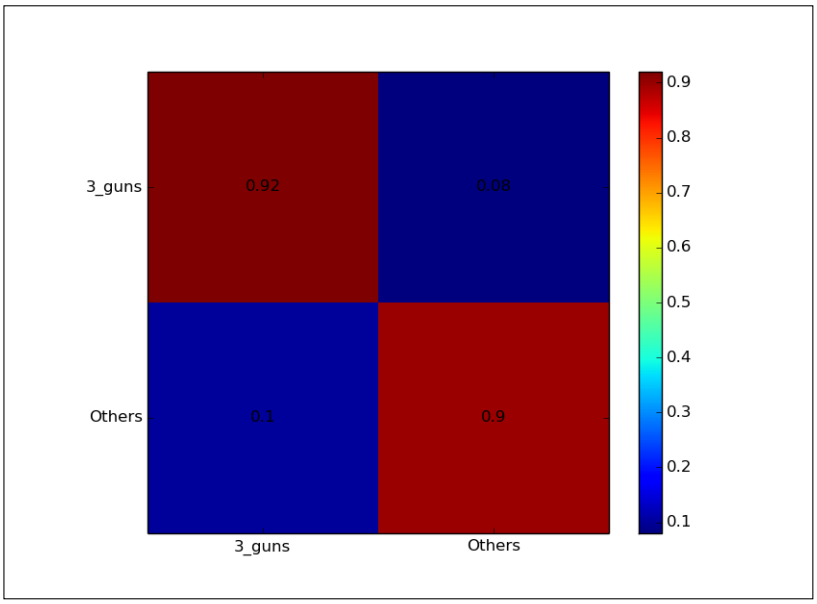

FIGURA 3.8. Ejemplo de una matriz de confusión para clasificador de Guns. SIFT $K = 100$, LBP $K = 50$ y $\sigma = 6$

En la primera fila se muestra el porcentaje de TPR y FNR, mientras que en la 2da fila se muestra el porcentaje de FPR y el de TNR. Todos los conceptos se explican en la sección 2.5.1.

Para el estudio de cada clasificador en particular, en este paso se obtienen 3 matrices de confusión (una por cada clasificador), donde cada una sólo posee 2 clases: objeto y no-objeto.

Para el caso del clasificador general, la matriz que se obtiene es única y posee las 4 clases utilizadas: Razor, Shuriken, Gun y Other.

Para la evaluación general del desempeño del algoritmo, se pretende lo siguiente:

- Mostrar matrices de confusión con el desempeño general de los 3 clasificadores.
- Mostrar gráficos de Precision-Recall que comparen la detección de cada uno de los objetos utilizados.
- Mostrar tablas que comparen el desempeño particular de cada clasificador, variando los parámetros del mismo, ya sea el número de árboles de decisión, la cantidad de *clusters* para el BoW, etc.

3.4. Ambiente de Desarrollo

Este trabajo se encuentra desarrollado en lenguaje *Python* versión 2.7.6 , compilado en 2 computadores, ambos con sistema operativo Ubuntu 14.04. El primer computador, utilizado para las pruebas pequeñas, fue un Dell Inspiron 14 con procesador Intel Inside CORE i5 y memoria RAM de 4.0 GB, mientras que el segundo, utilizado para las pruebas con todo el *dataset* fue un computador con procesador AMD FX(tm)-6300 Six Core con 3.5 GHz y memoria RAM de 8.0 GB.

Para el funcionamiento del algoritmo se utilizaron variadas librerías disponibles para el lenguaje *Python*, las cuales se detallan a continuación.

- Para el procesamiento de los arreglos, matrices y listas, se utilizó la librería Numpy versión 1.8.2
- Para el filtro Gaussiano se utilizó la librería *Scipy* versión 0.13.3

- Para el descriptor SIFT se utilizó una implementación realizada por Solem (2015) para el lenguaje *Python*, basada del Toolbox VLFeat para MatLab.
- Para el descriptor LBP se utilizó la librería Mahotas 1.4.0
- Para el K-MEANS, el clasificador Random Forest, el *Testing* y la creación de las matrices de confusión se utilizó la librería *Scikit-learn* versión 0.16.1
- Para el procesamiento de imágenes en general (lectura, escritura y muestra de imagen) se utilizó la librería OpenCV para *Python* versión 2.4.8

4. RESULTADOS Y DISCUSIÓN

Los experimentos y resultados obtenidos se dividieron en 3 secciones: primero se analizó la relevancia de los parámetros por clasificador, para lo cual se realizaron múltiples pruebas por clasificador, variando los parámetros dentro de un rango específico. En segundo lugar, se realizó un análisis de sensibilidad del algoritmo, en donde se compara éste con implementaciones más simples del mismo (es decir, sin SIFT, sin filtro Gaussiano, sin LBP, etc). En tercer lugar, se estudió el comportamiento y desempeño de cada clasificador en comparación a los otros, con el fin de evaluar la detección específica de cada objeto. Finalmente, se estudió el comportamiento del algoritmo como un todo, evaluando los casos donde el desempeño fue más alto que el resto.

Cada experimento, independiente de cual fuese, demoraba aproximadamente entre 45 minutos y 1 hora, incluyendo el Training y el Testing. El training en promedio demoraba aproximadamente 30 minutos, mientras que el testing se demoraba 15 minutos aprox., considerando extracción de SIFT y LBP para todas las imágenes. En promedio, el Testing **por imagen** se demoraba 0.7 segundos aprox., lo que es aceptable para un ambiente real de detección de objetos.

4.1. Base de Datos utilizada

Las imágenes utilizadas para todas las pruebas corresponden a objetos tales como hojas de afeitar, shurikens y pistolas, las que corresponden a las clases Razor, Shuriken y Gun respectivamente. Además se utilizaron imágenes de objetos (o partes de ellos) que no pertenecen a los 3 antes mencionados, los cuales conformaron la clase Others. Estas imágenes se encuentran en la base de datos GDXRay (Mery, Riffo, et al. (2015)), en la sección de *baggages*. De esta base de datos se utilizaron las carpetas B0049, B0050, B0051 y desde la B0078 a la B0082. Para todos los experimentos se utilizaron las mismas imágenes.

Las imágenes de entrenamiento en total utilizadas son 700, que consisten en:

(i) 100 imágenes de pistolas, localizadas en la carpeta B0049

(ii) 100 imágenes de shurikens, localizadas en la carpeta B0050

(iii) 100 imágenes de hojas de afeitar, localizadas en la carpeta B0051

(iv) 400 imágenes obtenidas de la carpeta B0078, que conformaron la clase Others.

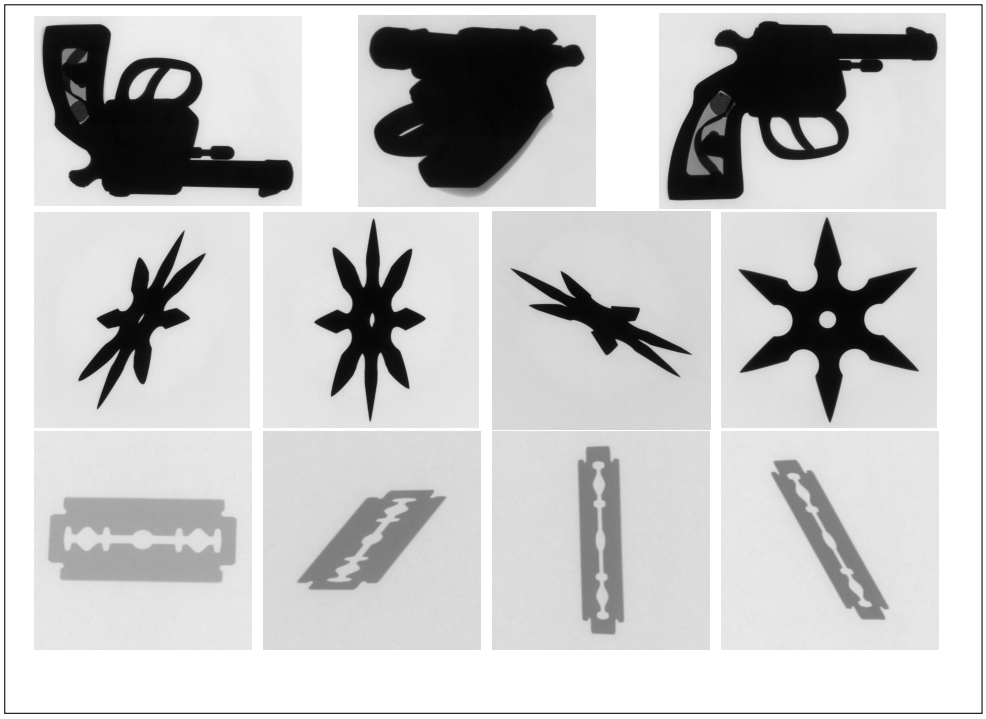

FIGURA 4.1. Ejemplos de imágenes del set de entrenamiento. pistolas (primera fila), shurikens (segunda fila) y hojas de afeitar (tercera fila)

Las imágenes de validación consisten en un set de imágenes similares al set de testing, pero en una menor cantidad. Estas imágenes permiten estimar los parámetros adecuados al algoritmo, además de realizar pruebas que permitan observar que el algoritmo funciona de forma correcta, sin necesidad de utilizar todo el set de testing. Estas imágenes sólo fueron utilizadas en la sección de ajuste de parámetros descrita en la sección 4.2. El set consiste en un total de 350 imágenes, descritas a continuación:

(i) Las primeras 50 imágenes de pistolas, localizadas en la carpeta B0079

(ii) Las primeras 50 imágenes de shurikens, localizadas en la carpeta B0080

(iii) Las primeras 50 imágenes de hojas de afeitar, localizadas en la carpeta B0081

(iv) Las primeras 200 imágenes obtenidas de la carpeta B0082, que conformaron la clase Others.

El set de testing consiste en un set de 700 imágenes, a las cuales se le realizaron las pruebas de análisis de desempeño del algoritmo, es decir, una vez realizado el entrenamiento, se utiliza este set para obtener los resultados de clasificación. Estas imágenes contienen alguno de los objetos estudiados, dentro de una maleta con otros objetos distractores. Este set consiste en las siguientes imágenes:

(i) Las últimas 100 imágenes de pistolas, localizadas en la carpeta B0079
(ii) Las últimas 100 imágenes de shurikens, localizadas en la carpeta B0080
(iii) Las últimas 100 imágenes de hojas de afeitar, localizadas en la carpeta B0081
(iv) Las últimas 400 imágenes obtenidas de la carpeta B0082, que conformaron la clase Others.

Las imágenes de validación y testing son muy similares. No obstante, ninguna imagen de validación y testing es la misma.

4.2. Ajuste de parámetros

El objetivo de este experimento fue buscar los mejores parámetros a utilizar en cada uno de los 3 clasificadores, de modo de ajustarlos para que cada objeto obtuviera un buen desempeño en su clasificación. Para esto, se utilizó el set de validación descrito en la sección 4.1. Un punto importante es que en este experimento se testeó cada clasificador de forma separada, saltándose la etapa de score validation descrita anteriormente.

Los parámetros que se ajustaron para cada clasificador fueron: la cantidad n de árboles del Random Forest, el valor de K para las Bag of Words de los descriptores SIFT, el valor de K para las Bag of Words de los descriptores LBP y el valor de σ utilizado en el filtro Gaussiano para cada clasificador. Estos parámetros fueron elegidos dado que se consideraron relevantes para el desempeño final.

Las principales variaciones de cada parámetro fueron:

- La cantidad de árboles del Random Forest utilizada fue de 1000, 2000 y 4000 árboles para los experimentos.
- Los valores de K para la Bag of Words de los descriptores SIFT fueron de 100, 200, 400 y 800.
- Los valores de K para la Bag of Words de los descriptores LBP fueron de 50, 100 y 200.
- Los valores de σ utilizados en el filtro Gaussiano variaron entre 0 y 8, donde el valor 0 indica que no se realiza filtro (se deja la imagen original).

Se decidió variar cada parámetro al doble en cada experimento, como una forma de que hubiera una verdadera influencia del mismo en cada uno. Además, es una forma de no realizar tanto experimento, debido a que la cantidad de permutaciones posibles es demasiada (en este caso se realizaron $3 \cdot 4 \cdot 3 \cdot 9 = 324$ experimentos como mínimo en esta etapa, que demoraron mas o menos 1 mes). No obstante, habían parámetros que daban resultados muy bajos, los cuales no era necesario probar con todas las permutaciones posibles para el mismo.

Para cada experimento se calculó su porcentaje de TPR, TNR, FPR y FNR (que se muestra como un número decimal) y junto a ellos su respectivo Precision-Recall y el promedio de la diagonal (promedio de los porcentajes de TPR y TNR). Dada que la cantidad de permutaciones existentes es demasiada para mostrarla en esta sección, se muestran las más relevantes en las tablas del Anexo A, al final del documento. En las tablas de esta sección se muestra el TPR, TNR y su promedio.

Dado que se validaron todos los parámetros de forma conjunta, se mostrará un resumen de cada parámetro tomando el resultado promedio de todos los experimentos. Por ejemplo, para obtener el valor de $K = 200$, se tomaron todos los experimentos que usaron un valor de $K = 200$ y se obtuvo el promedio de los mismos.

4.2.1. Resultados del ajuste de parámetros

Para el caso de la cantidad N_Tree de árboles utilizados, en las tablas 4.1, 4.2 y 4.3 se muestra un resumen del promedio de resultados, el Precision y el Recall. Como se puede

observar, el promedio en cada clasificador es muy similar para los 3 valores estudiados, con lo que se puede inducir que el valor de N_Tree no influye en los resultados finales. No obstante, el tiempo de ejecución aumentaba a medida que N_Tree lo hacía, con lo cual se debe tener en cuenta al momento de elegir el valor apropiado. Otro punto importante es evitar el sobreajuste de resultados, en donde los resultados se sesgan a sólo las imágenes probadas, obteniendo conclusiones erradas cuando aparece una nueva imagen. Se conoce que al aumentar el valor de N_Tree, es más probable que ocurra un sobreajuste.

TABLE 4.1. Resumen de validación de N_Tree para el clasificador de Razor

Razors

N_Tree	TPR	TNR	Promedio TPR y TNR
1000	0,5263	0,9951	0,7607
2000	0,5286	0,9951	0,7618
4000	0,5168	0,9959	0,7563

TABLE 4.2. Resumen de validación de N_Tree para el clasificador de Shuriken

Shurikens

N_Tree	TPR	TNR	Promedio TPR y TNR
1000	0,3549	0,9989	0,6769
2000	0,3502	0,9983	0,6742
4000	0,3543	0,9992	0,6767

TABLE 4.3. Resumen de validación de N_Tree para el clasificador de Gun

Guns

N_Tree	TPR	TNR	Promedio TPR y TNR
1000	0,6403	0,8797	0,7600
2000	0,6356	0,8883	0,7619
4000	0,6362	0,8856	0,7609

El segundo parámetro validado fue el valor de σ para el filtro Gaussiano, cuyos resultados se muestran en las tablas 4.4, 4.5 y 4.6. En este caso se observa un particular comportamiento.

TABLE 4.4. Resumen de validación de σ para el clasificador de Razors

Razors			
Sigma	**TPR**	**TNR**	**Promedio TPR y TNR**
0	0,4889	0,9978	0,7433
1	0,6867	0,9967	0,8417
2	0,7622	0,9889	0,8756
3	0,6556	0,9944	0,8250
4	0,4133	0,9933	0,7033
5	0,3422	0,9967	0,6694
6	0,3511	0,9978	0,6744
7	0,3933	0,9922	0,6928
8	0,4000	0,9944	0,6972

TABLE 4.5. Resumen de validación de σ para el clasificador de Shurikens

Shurikens			
Sigma	**TPR**	**TNR**	**Promedio TPR y TNR**
0	0,0000	0,9985	0,4993
1	0,0007	0,9981	0,4994
2	0,0674	0,9967	0,5320
3	0,1933	0,9985	0,5959
4	0,5222	1,0000	0,7611
5	0,7837	1,0000	0,8919
6	0,9044	0,9996	0,9520
7	0,9044	0,9989	0,9517
8	0,8800	1,0000	0,9400

TABLE 4.6. Resumen de validación de σ para el clasificador de Guns

Guns			
Sigma	**TPR**	**TNR**	**Promedio TPR y TNR**
0	0,2763	0,9563	0,6163
1	0,3037	0,9252	0,6144
2	0,4711	0,9267	0,6989
3	0,7933	0,8711	0,8322
4	0,8578	0,8170	0,8374
5	0,8622	0,8448	0,8535
6	0,8970	0,8504	0,8737
7	0,9222	0,8611	0,8917
8	0,9444	0,8767	0,9106

Para la clase Gun, el mejor desempeño se obtuvo para el valor $\sigma = 8$, donde claramente se observa que, a medida que el valor aumenta, su desempeño mejora. No obstante, para la clase Razor, el mejor desempeño se obtuvo claramente para el valor $\sigma = 2$, mientras que a medida el valor de σ aumentaba, el desempeño cae drásticamente.

Finalmente, para el clasificador de Shuriken, existe una alza en el desempenño hasta el valor $\sigma = 6$, donde comienza a descender.

Para este parámetro se decidió entonces tomar los valores $\sigma = 2$ para el clasificador de Razor, $\sigma = 6$ para el de Shuriken y $\sigma = 8$ para el de Gun.

El siguiente parámetro validado fue el valor de K representantes SIFT, donde sus resultados se muestran en las tablas 4.7, 4.8 y 4.9. El valor $K = 800$ no aparece en estos resultados dado que no se probaron todas las posibles permutaciones del mismo, además de que su resultado era similar (o en algunos casos peor) a $K = 400$, por lo cual se consideró similar al de $K = 400$.

Aquí se puede apreciar que existe una ligera alza del desempeño a medida que el valor aumenta. No obstante, tanto para Razor como para Shuriken, al aumentar de 200 a 400 puntos SIFT el alza comienza a ser mucho mas leve que el caso de 100 a 200, sobre todo para la clase Shuriken. Para el clasificador de Guns, el mayor aumento ocurre con valor de $K = 100$, para luego ir disminuyendo el desempeño a medida que K aumenta.

Tomando en consideración que a mayor valor de K, mayor es el tiempo de ejecución, se decidió tomar como parámetros en cada clasificador los siguientes: para los de Razor y Shuriken, valores de $K = 200$ y $K = 400$, mientras que para el de Gun, valores de $K = 100$ y $K = 200$.

TABLE 4.7. Resumen de validación de K centroides SIFT para el clasificador de Razors

Razors

K SIFT	TPR	TNR	Promedio TPR y TNR
100	0,4260	0,9937	0,7098
200	0,5540	0,9959	0,7749
400	0,5917	0,9965	0,7941

TABLE 4.8. Resumen de validación de K centroides SIFT para el clasificador de Shurikens

Shurikens

K SIFT	TPR	TNR	Promedio TPR y TNR
100	0,3255	0,9985	0,6620
200	0,3546	0,9983	0,6764
400	0,3705	0,9995	0,6850

TABLE 4.9. Resumen de validación de K centroides SIFT para el clasificador de Guns

Guns

K SIFT	TPR	TNR	Promedio TPR y TNR
100	0,6661	0,8824	0,7743
200	0,6540	0,8794	0,7667
400	0,5895	0,8917	0,7406

El último parámetro validado fue el valor de K para los centroides LBP, donde los resultados se muestran en las tablas 4.10, 4.11 y 4.12. Como se puede apreciar, en este caso se observa que existe una tendencia a aumentar entre los valores 50 y 100, para luego mantenerse

estable entre los valores 100 y 200. Esto explica que el valor de K de los centroides LBP no es muy influyente en el desempeño, por lo cual, se decidió probar con todos ellos para cada clasificador.

TABLE 4.10. Resumen de validación de K centroides LBP para el clasificador de Razors

Razors			
K LBP	**TPR**	**TNR**	**Promedio TPR y TNR**
50	0,5172	0,9797	0,7484
100	0,5295	0,9952	0,7624
200	0,5168	0,9956	0,7562

TABLE 4.11. Resumen de validación de K centroides LBP para el clasificador de Shurikens

Shurikens			
K LBP	**TPR**	**TNR**	**Promedio TPR y TNR**
50	0,3337	0,9838	0,6587
100	0,3559	0,9984	0,6771
200	0,3559	0,9983	0,6771

TABLE 4.12. Resumen de validación de K centroides LBP para el clasificador de Guns

Guns			
K LBP	**TPR**	**TNR**	**Promedio TPR y TNR**
50	0,6314	0,8722	0,7518
100	0,6289	0,8838	0,7563
200	0,6387	0,8835	0,7611

4.2.2. Análisis del ajuste de parámetros

Cada uno de los parámetros ajustado merece un poco de análisis, ya que cumplen diferentes roles en el desempeño de cada clasificador. Se pudo apreciar que existen parámetros que no influyen en la clasificación final, mientras que otros son de mucha relevancia en la misma. Una tabla resumen de los parámetros elegidos en cada caso se presenta en la tabla 4.13.

TABLE 4.13. Resumen de los parámetros a utilizar para cada clasificador

	Clasificadores		
	Razors	**Shurikens**	**Guns**
N_Tree	[1000, 2000, 4000]	[1000, 2000, 4000]	[1000, 2000, 4000]
Sigma	2	6	8
K SIFT	[200, 400]	[200, 400]	[100, 200]
K LBP	[50, 100, 200]	[50, 100, 200]	[50, 100, 200]

A continuación presentamos un análisis detallado de cada uno de ellos:

(i) Valor de n árboles en el Random Forest:

Este parámetro presentó un desempeño similar, independiente del valor utilizado, por lo cual se podía utilizar cualquier valor que se estimara conveniente. No obstante, existe una compensación entre desempeño y tiempo, lo que es necesario considerar al momento de elegir un parámetro adecuado. Los resultados mostraron que valores superiores a 4000 aumentaban drásticamente el tiempo de ejecución del Training, por lo cual se decidió utilizar principalmente valores entre 2000 y 4000.

(ii) Valor de σ del filtro Gaussiano:

El valor σ sin duda fue el más relevante, ya que se marcó una clara tendencia al alza en el desempeño con este valor para cada clasificador. Por ejemplo, el clasificador Razor tiene claramente como valor óptimo $\sigma = 2$, mientras que para los otros casos fue $\sigma = 6$ y $\sigma = 8$ respectivamente. Valores diferentes diferenciaban con gran contraste el desempeño.

Una posible explicación a este comportamiento es que las imágenes de hojas de afeitar utilizadas son más claras que las de shurikens y pistolas, por lo cual, un valor σ alto provoca que ésta se confunda con el fondo y se vuelva borrosa e imperceptible. Por el contrario, las imágenes de shurikens y pistolas son mucho más oscuras, lo que provoca que valores bajos de σ no permitan detectar de forma clara los puntos SIFT que se necesitan (no se logra apreciar el efecto del filtro), mientras

que de cierto valor en adelante, se logra dicho objetivo. Esto se logra con valores iguales o mayores a $\sigma = 5$.

Se debe tener mucha consideración en este parámetro al momento de agregar un nuevo clasificador (si se quiere realizar a futuro).

(iii) Valor de K para los centroides de los descriptores SIFT:

Para los clasificadores de Shuriken y Razor, a medida que el valor de este K aumenta, también lo hacía el desempeño del mismo. No obstante, existe un punto en el cual los valores tienden a mantenerse, por lo cual se demuestra que no es necesario seguir aumentando el valor del mismo. Una explicación de este hecho es que tanto las hojas de afeitar como las shurikens son simétricas, por lo cual, sólo es necesaria una cierta cantidad de puntos (centroides) para describirlas, ya que después tienden a repetirse, sobre todo considerando el hecho de que los puntos SIFT son invariantes a la escala y rotación.

Por otro lado, en el caso del clasificador de Guns, el valor óptimo encontrado es de $K = 200$. Esto podría explicarse al hecho de que las pistolas no son simétricas, por lo cual una cantidad muy alta de puntos SIFT implica que el algoritmo pueda confundirse con los otros objetos (no-pistolas), que tampoco tienen una estructura definida. Por ende, es necesario utilizar un valor de K preciso para describir las pistolas de la mejor forma posible.

(iv) Valor de K para los centroides de los descriptores LBP:

Este parámetro no mostró una clara tendencia al alza o a la baja en el desempeño de los clasificadores, por lo que se decidió utilizar todos los valores posibles. Esto puede explicarse al hecho de que las áreas de interés para LBP son fijas, tendiendo a repetirse a valores mayores (similar al caso de los descriptores SIFT). Otros experimentos no mostrados en las tablas mostraron que valores bajo $K = 50$ presentaban uan gran baja en el desempeño, lo que refuerza la idea anterior.

4.3. Análisis de sensibilidad

En esta sección se muestra y realiza una comparación entre el algoritmo desarrollado y diversas variaciones del mismo, con el fin de justificar cada una de las etapas de la metodología expuesta. Para esto se utilizó un algoritmo base, el cual sólo posee 2 etapas: extracción de descriptores SIFT y entrenamiento con Random Forest.

Las variaciones utilizadas en esta sección son las siguientes:

- Algoritmo base (antes descrito)
- Extracción de descriptores, sin utilizar Filtro Gaussiano
- Extracción de descriptores, con filtro mediana
- Extracción de descriptores SIFT solamente
- Extracción de descriptores LBP solamente
- Entrenamiento utilizando SVM, usando un kernel sigmoideal.
- Algoritmo completo (el utilizado en este trabajo)

Los resultados de cada uno de los algoritmos anteriores se muestra en la tabla 4.14.

TABLE 4.14. Resultados de cada experimento del análisis de sensibilidad

	Razors		**Shurikens**		**Guns**	
Algoritmo	TPR	TNR	TPR	TNR	TPR	TNR
Basico	0,54	1,00	0,00	1,00	0,30	0,76
Sin Filtro	0,72	1,00	0,00	1,00	0,40	0,96
Mediana	0,80	1.00	0,02	1,00	0,48	0,94
Sin LBP	0,86	1,00	0,92	1,00	0,96	0,79
Sin SIFT	0,04	0,97	0,00	0,97	0,02	0,89
SVM	0,74	0,98	0,74	0,98	0,48	0,70
Completo	0,82	1,00	0,92	1,00	0,99	0,86

A simple vista se puede observar que cada experimento presenta un comportamiento particular. En primer lugar, se observa que el algoritmo con peor desempeño es el que no utiliza puntos SIFT, ya que no logra un porcentaje de clasificación mayor a 5% en cada objeto, lo cual

es un resultado extremadamente bajo. Esto lo vuelve infactible de realizar y a la vez, reafirma la importancia de los puntos SIFT como descriptores en este trabajo.

Los siguientes algoritmos que presentan un desempeño bajo son el algoritmo básico y el que no utiliza filtro, en los cuales se puede observar que el desempeño para las clases Razor y Gun es bajo (50%) y para Shuriken es nulo. Ambos no poseen ningún filtro, lo que puede ser la razón de su desempeño.

Después se observa una mejora al utilizar el filtro de mediana, sobre todo al detectar la clase Razor. No obstante, para los otros objetos la mejora no es notoria, por lo cual se descartó como posible filtro.

Luego se encuentra el algoritmo SVM con una mejora notoria de desempeño. Si bien el alza es justificada, al compararla con el algoritmo que utiliza Random Forest, se observa que la diferencia no es menor. Además, el desempeño en Guns sigue siendo menor al 50%. Por estas razones, sumando el hecho de que el tiempo de ejecución de ambos es similar, se prefirió el uso de Random Forest por sobre SVM.

Finalmente, observamos que entre el algoritmo que no usa LBP y el que usa LBP no existe una diferencia muy notoria de desempeño, lo que lleva al pensamiento de que tal vez LBP no es necesario para el desempeño logrado. Sin embargo, en los gráficos a continuación se explicará la razón del uso de esta *feature*, sobre todo para pistolas.

A continuación se presenta un análisis más detallado en la detección de cada objeto por separado, utilizando gráficos de Precision-Recall.

Para el caso de las Razors, en la figura 4.2 se observa un comportamiento relativamente parejo para cada algoritmo probado, con la excepción del algoritmo que no utiliza puntos SIFT. Si tomamos como referencia el algoritmo completo (azul) podemos observar que existen 3 algoritmos con desempeño más bajo y 2 con desempeño más alto.

Los que presentan un desempeño más alto que el algoritmo completo son el que utiliza filtro mediana y el que no utiliza LBP. Observando la tabla 4.14 podemos decir que la diferencia entre el filtro mediana y el Gaussiano es sólo de 2%, lo que es despreciable. Para el caso

del algoritmo que no utiliza LBP, la diferencia es mayor. No obstante, se prefiere utilizar LBP en este trabajo. La razón de esto se dirá cuando se estudie el comportamiento del gráfico del clasificador de Guns.

Lo otro que se puede destacar es que el uso de Random Forest y filtro sí es importante para la detección de este objeto, ya que la diferencia en porcentaje es de un 10% aprox.

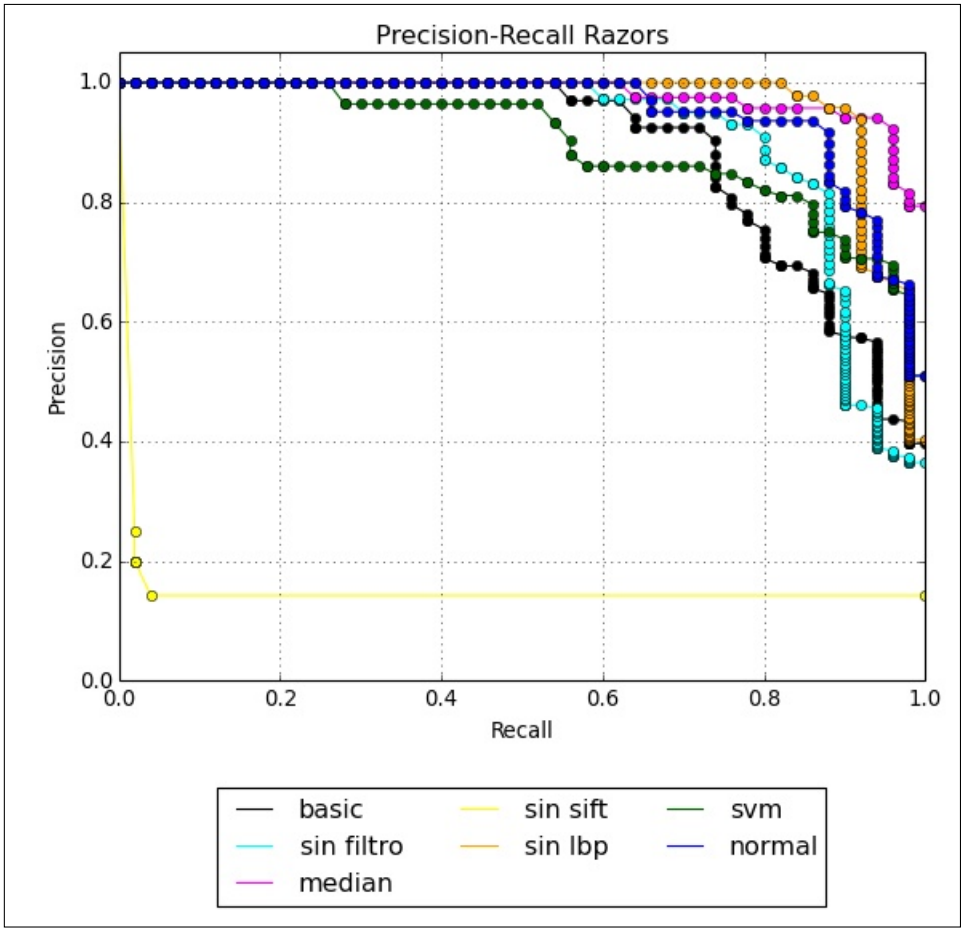

FIGURA 4.2. Gráfico de Precision-Recall para los diferentes experimentos mencionados anteriormente, con respecto a la detección de hojas de afeitar.

Para la clase Shuriken, la figura 4.3 muestra un comportamiento más radical que para la clase Razor, dado que existen 4 algoritmos con un desempeño muy pobre, los cuales comparten el hecho de que ninguno de ellos utiliza un filtro Gaussiano. Esto muestra la importancia que requiere el uso de este filtro para la detección de shurikens, lo cual se descubrió cuando se realizaron las primeras pruebas en este trabajo. La razón de este comportamiento se

puede explicar de la siguiente manera: el hecho de suavizar las puntas de las shurikens permite al SIFT detectar estos puntos, vitales para describir la misma. De no detectar estos puntos, el algoritmo no es capaz de clasificar shurikens, lo cual se corroboró con los resultados de la sección anterior.

Por otro lado, los otros algoritmos (sin LBP, SVM y el completo) presentan un resultado razonable, aunque el único que logra un gráfico perfecto es el algoritmo completo. No obstante, el algoritmo que no utiliza LBP también presenta un buen desempeño, lo que demuestra que no es de vital importancia el utilizar esta *feature* para clasificar shurikens.

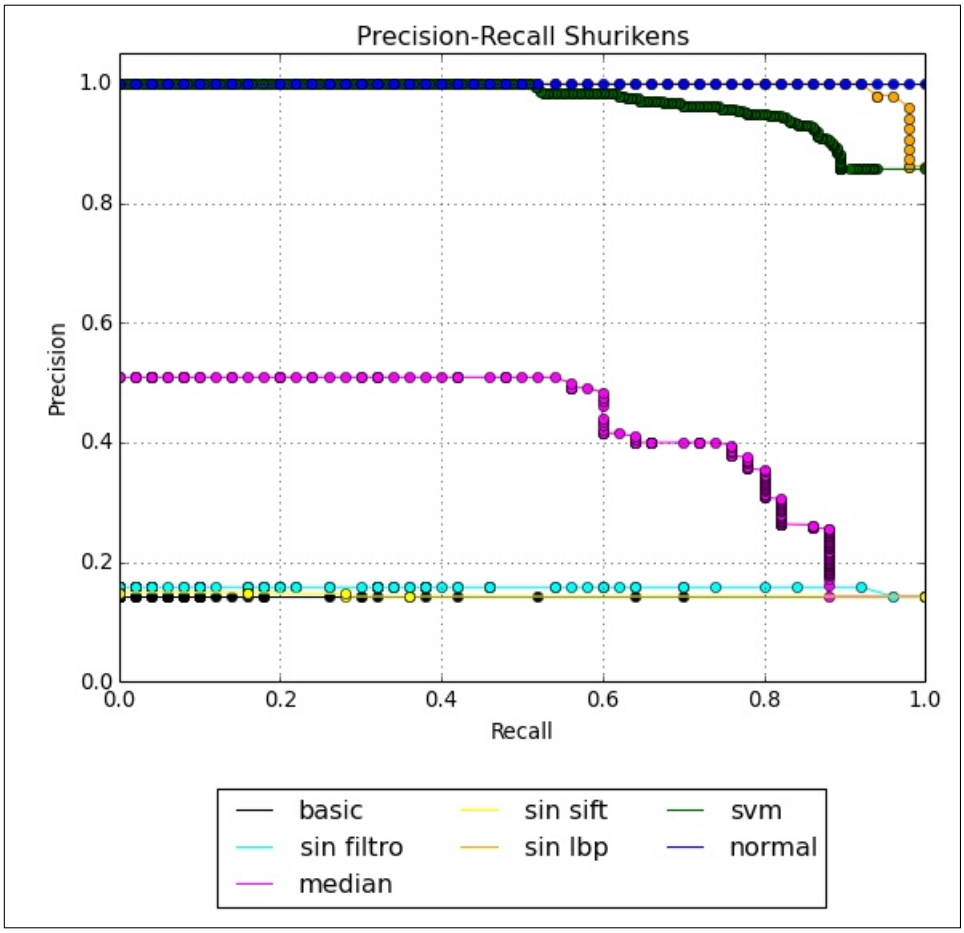

FIGURA 4.3. Gráfico de Precision-Recall para los diferentes experimentos mencionados anteriormente, con respecto a la detección de shurikens.

Finalmente, para el caso del clasificador de Guns, la figura 4.4 muestra un comportamiento en ascenso para cada uno de los algoritmos utilizados, siendo el con mejor desempeño (nuevamente) el algoritmo completo. Los algoritmos con peor desempeño fueron el sin SIFT y el que utiliza SVM, lo que se podía predecir observando la tabla 4.14. El SVM presenta una mayor dificultad al momento de clasificar pistolas, lo cual no se observó en los casos anteriores. Esto puede deberse al hecho de que las pistolas son un objeto asimétrico, a diferencia de los anteriores.

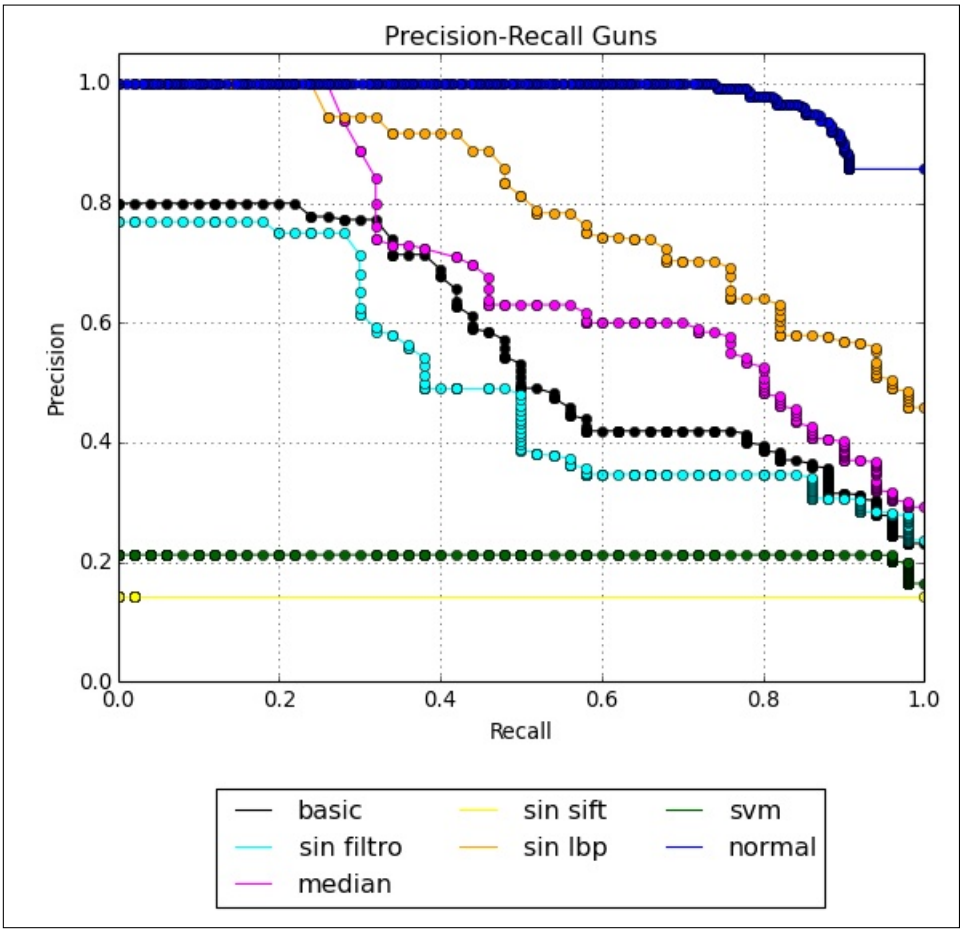

FIGURA 4.4. Gráfico de Precision-Recall para los diferentes experimentos mencionados anteriormente, con respecto a la detección de pistolas.

Posteriormente se pueden ordenar los algoritmos de la siguiente manera según su desempeño: sin filtro, básico, mediana, sin lbp y finalmente (con una mayor diferencia) el completo. Aquí se puede observar la diferencia de utilizar cada uno de los procesos realizados,

sobre todo el LBP. A diferencia de los otros objetos, LBP resulta ser vital para la detección de pistolas, de la misma forma que el filtro Gaussiano lo es para shurikens. Esta es la razón por la cual se utilizó LBP en este trabajo, pese a que presentaba una baja en la detección de Razor, la cual no se compara a la baja con respecto al clasificador de Gun.

No se decidió realizar algoritmos separados (por ejemplo, que el clasificador de Gun utilizara LBP y los otros no) por el hecho de que se buscaba una metodología lo más homogénea posible entre los 3 objetos, de otro modo, los scores obtenidos en la sección 4.5.1 se verían influenciados.

4.4. Desempeño por objeto

En este experimento el objetivo fue observar y analizar el desempeño de cada clasificador, en lo que respecta a la clasificación del objeto que busca. Nuevamente, en esta etapa no se realiza la etapa de unión de validación de scores explicada en la sección 3.3.2. Las imágenes utilizadas en este experimento son las del set de testing.

Los resultados se presentarán de 2 formas: en la primera parte, cada clasificador será interpretado con un gráfico de Precision-Recall, como el explicado en la sección 2.5.2. Dado que el parámetro más discriminatorio fue el valor de σ del filtro Gaussiano, se utilizará este parámetro para cada una de las curvas del gráfico. Los otros parámetros se mantienen fijos, utilizando los descritos en la sección 4.2.2. Al final de esta parte se muestra un gráfico de Precision-Recall que compara las 3 mejores curvas de cada objeto.

En la segunda parte se muestran 2 tablas: en la primera aparecen los promedios de todos los clasificadores por objeto (independiente de los parámetros de cada uno) mientras que la segunda tabla muestra los clasificadores con mejor y peor rendimiento. Todos los clasificadores estudiados en esta sección utilizaron los parámetros de la tabla 4.13.

Todos los experimentos más importantes de esta sección se pueden observar en detalle en el Anexo B.

4.4.1. Resultados del desempeño por objeto

4.4.1.1. Gráficos de Precision-Recall

Para el caso del clasificador de Razor, su gráfico de Precision-Recall se muestra en la figura 4.5 a diferentes valores de σ.

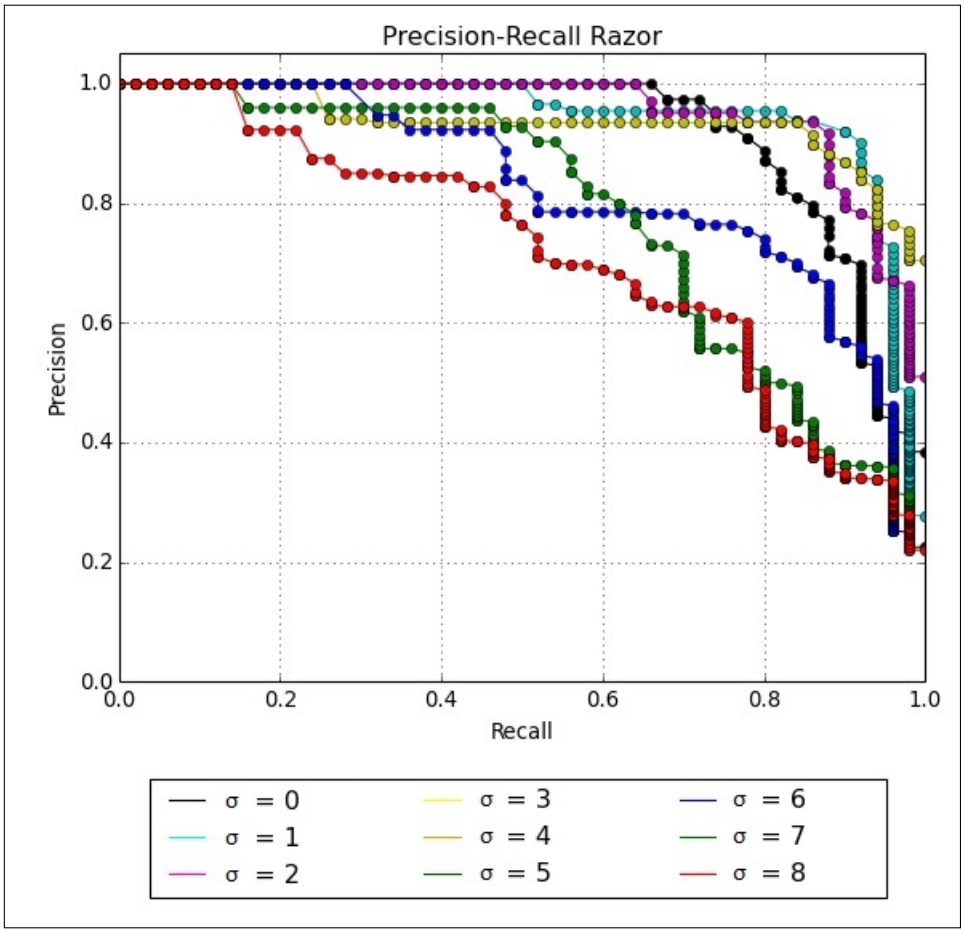

FIGURA 4.5. Gráfico de Precision-Recall para el clasificador de Razor, a diferentes valores de σ

El gráfico muestra que la mejor curva se encuentra entre la celeste ($\sigma = 1$) y la morada ($\sigma = 2$). El primer caso es el que muestra un mejor valor entre *precision* y *recall*, ya que contienen el punto más próximo a la esquina superior derecha del gráfico (que como se explicó, es el punto ideal). La curva morada es la que se mantiene por mucho más tiempo en valor de *precision* igual a 1, el cual recién comienza a decaer a un *recall* aproximado de 0.65. La

negra se descarta ya que posee un decaimineto bien pronunciado a partir de un valor de 0.7 en *recall*. Además, el gráfico confirma el hecho de que a niveles altos de σ, este clasificador decae su rendimiento.

Las 2 mejores curvas presentan un alto porcentaje de *precision* a medida que *recall* aumenta. Esto se debe principalmente al hecho de que existe una cantidad casi nula de FP, es decir, el algoritmo pocas veces se equivoca en confundir una hoja de afeitar con cualquier otro objeto. No obstante, la baja a *recall* alto se debe a que existe una cantidad considerable de hojas de afeitar que el algoritmo no logra detectar.

Para el clasificador de Shuriken, su gráfico de Precision-Recall se muestra en la figura 4.6, a diferentes valores de σ.

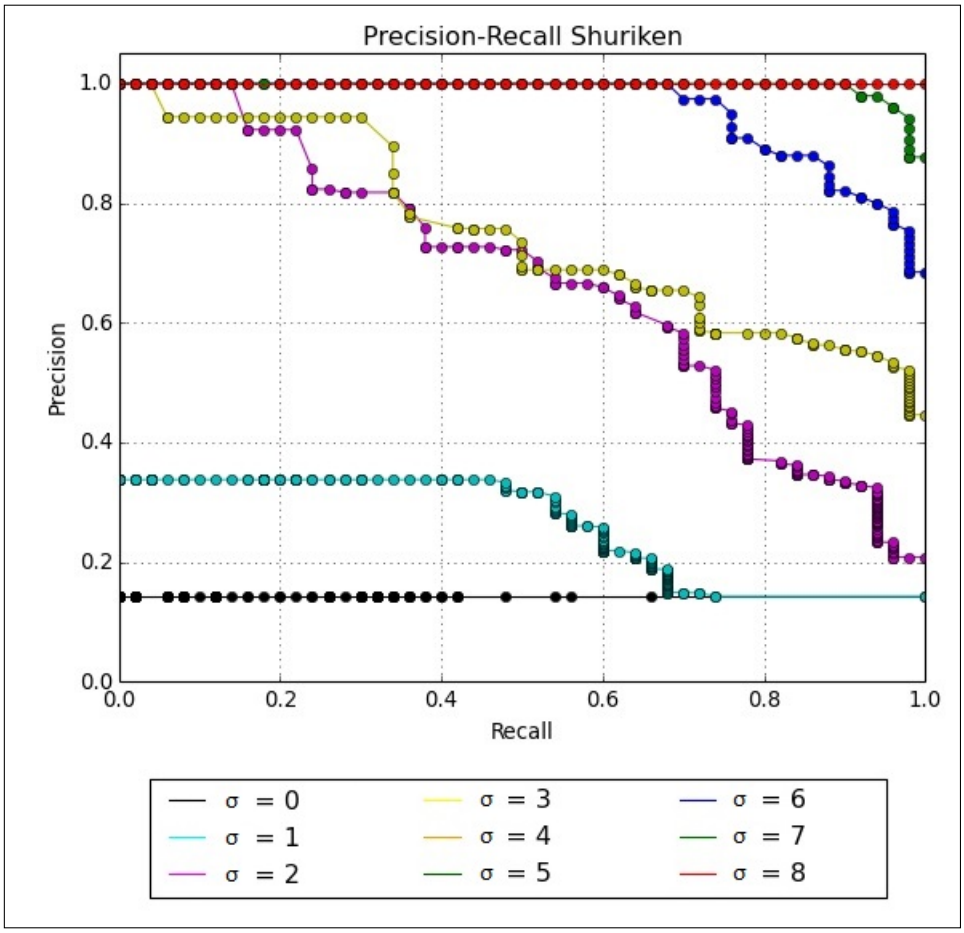

FIGURA 4.6. Gráfico de Precision-Recall para el clasificador de Shuriken, a diferentes valores de σ

Este gráfico presenta una clara tendencia al alza a medida que σ aumenta, llegando a tener una curva perfecta a valor $\sigma = 6$ (curva roja), siendo la mejor de los 3 objetos. Esta curva perfecta se debe a que prácticamente siempre hay una cantidad nula de FP, apoyado con una alta cantidad de TP. Si se observa el Anexo A, se aprecia que en la mayoría de los casos el valor de *precision* es 1, independiente de los parámetros del clasificador.

Luego, para el clasificador de Guns, su gráfico de Precision-Recall se muestra en la figura 4.7, a diferentes valores de σ.

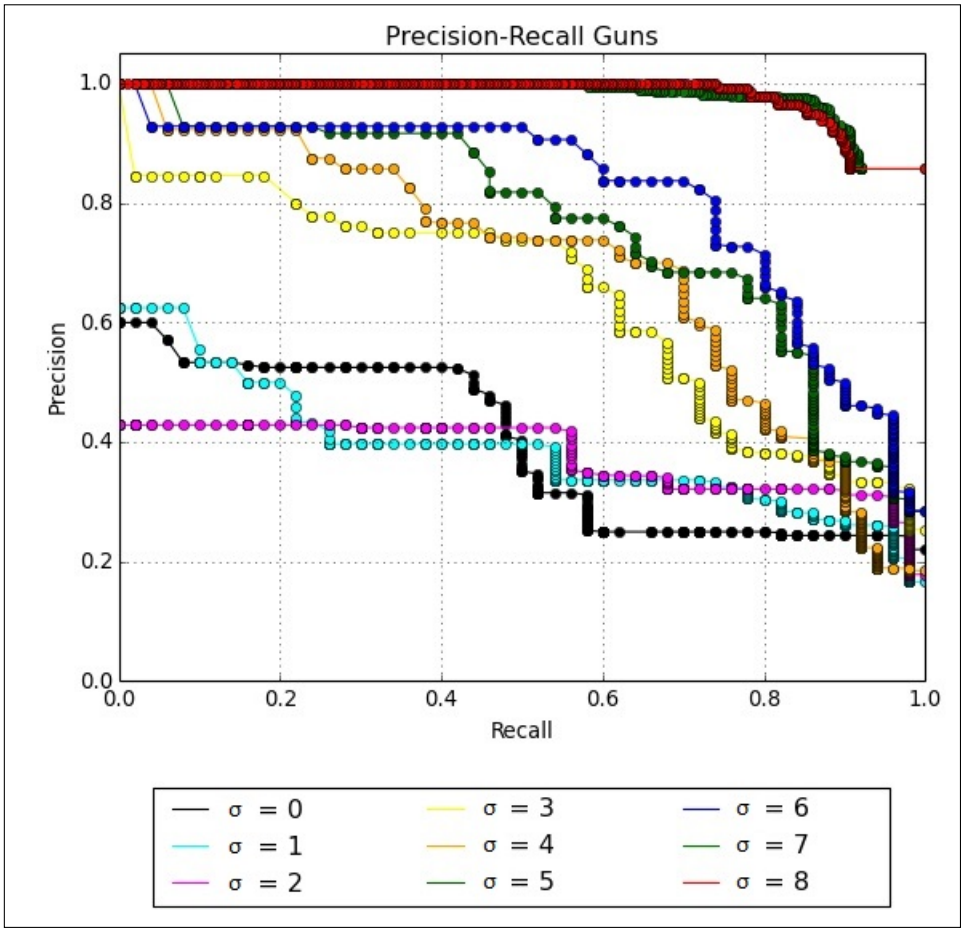

FIGURA 4.7. Gráfico de Precision-Recall para el clasificador de Guns, a dife-rentes valores de σ

Este gráfico tiene un comportamiento similar al clasificador de Shuriken, puesto que a medida que σ aumenta, la curva es mejor. No obstante, salvo los valores de $\sigma = 7$ y $\sigma = 8$, existe una clara tendencia a la baja del *precision* a bajos valores de *recall*, donde ya comienza

a descender a partir del valor 0.1 aprox. Esto se debe a que este clasificador es que presenta la mayor cantidad de FP de los 3 estudiados, puesto que las pistolas son un objeto más fácil de confundir con otros, dado que no posee simetría como los anteriores. De todas formas, a valores σ mayores que 6, el nivel de *precision* se mantiene alto (sobre 0.8 en las mejores curvas por un buen margen de *recall*), debido a la gran cantidad de TP, que es mayor al del clasificador de Razor.

Finalmente, el gráfico 4.8 muestra una comparación entre las 3 mejores curvas de cada clasificador, donde claramente se aprecia que el algoritmo se comporta mejor frente a shurikens, que frente a los otros 2 objetos, siendo el clasificador de Razor el de más bajo desempeño.

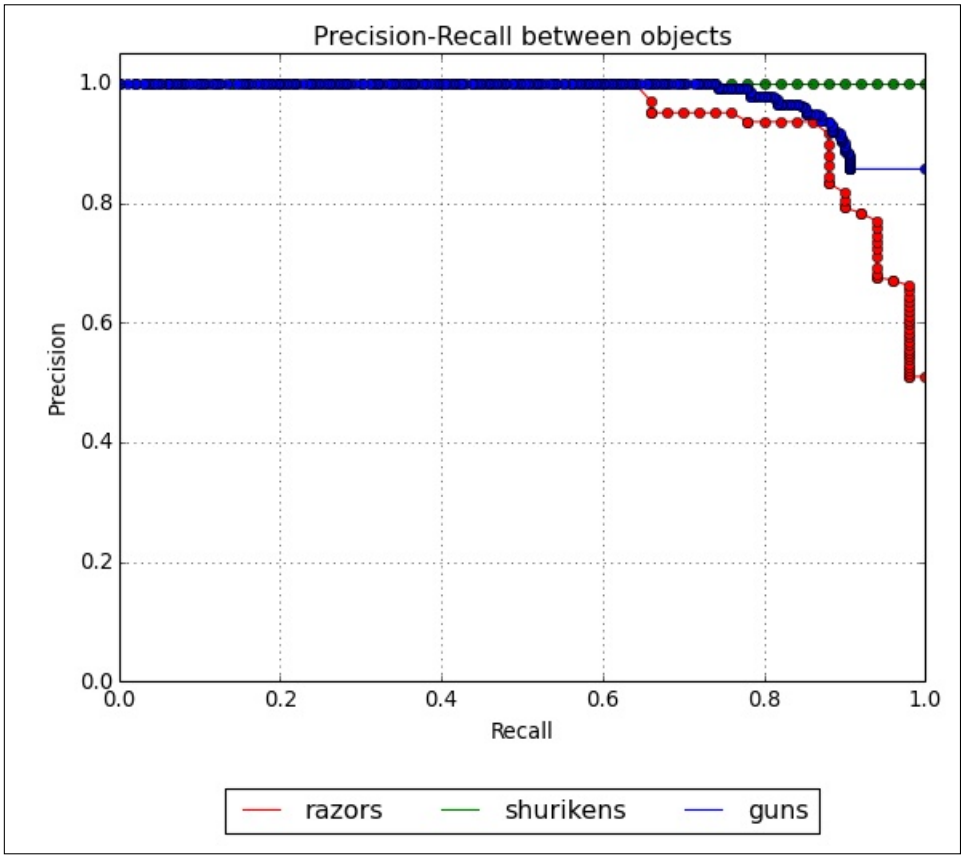

FIGURA 4.8. Gráfico de Precision-Recall mostrando las mejores curvas de los 3 clasificadores de forma simultánea.

4.4.1.2. Tablas del desempeño en TPR y TNR

La tabla 4.15 muestra los porcentajes promedio de TPR y TNR (como decimal) que se logró en cada clasificador. Esta tabla sólo considera los resultados de los clasificadores que utilizan los parámetros descritos en la tabla 4.13.

Como se puede apreciar, en promedio la clasificación de Shuriken es la más óptima para todos los parámetros utilizados, ya que, como se dedujo de los gráficos de Precision-Recall, la cantidad de TP y TN es bastante alta en todos los experimentos, donde el promedio para TNR es casi 1, lo que indica que la precisión para este objeto es muy buena.

TABLE 4.15. Promedio de rendimientos por clasificador

Clasificadores					
	TPR	**FNR**	**TNR**	**FPR**	**Rendimiento**
Razors	0,7750	0,2250	1,0000	0,0000	0,8875
Shurikens	0,9139	0,0861	0,9989	0,0011	0,9564
Guns	0,9761	0,0239	0,8250	0,1750	0,9006

Por otro lado, se observa que para la clase Razor el promedio de TPR es bastante bajo, no alcanzando el 80%, pero el promedio de TNR es el 100%. Esto último provoca que el *precision* sea alto (como se observa en el gráfico correspondiente) pero baja el promedio de la diagonal, por lo cual su desempeño es algo más bajo que el de la clase Gun, que presenta lo contrario al de Razor. Este comportamiento no se puede concluir solamente de los gráficos.

Por último, el clasificador de Guns posee el más alto promedio de TPR, pero un bajo porcentaje de TNR en comparación a los anteriores (lo que provoca que aumenten los FP), esto provoca que su diagonal sea similar al de Guns.

Luego, la tabla 4.16 muestra el mejor y peor clasificador, considerando como parámetro de decisión el promedio entre TPR y TNR del clasificador correspondiente.

TABLE 4.16. Mejores y peores clasificadores para cada objeto

	Mejor clasificador			Peor clasificador		
	TPR	**TNR**	**Promedio**	**TPR**	**TNR**	**Promedio**
Razors	0,8200	1,0000	0,9100	0,2800	0,9800	0,6300
Shurikens	0,9900	1,0000	0,9950	0,0000	1,0000	0,5000
Guns	0,9900	0,8600	0,9250	0,2400	0,7600	0,5000

Es curioso observar la clasificación de Shuriken, el cual presenta el mejor y peor desempeño de los 3, (dependiendo de los parámetros utilizados) lo que indica la importancia del ajuste de parámetros en el algoritmo desarrollado. Tanto las clases Shuriken como Gun obtienen un desempeño de casi 99% de TPR, aunque para el caso de este último su porcentaje de TNR es bajo. Razor no alcanza una detección mayor al 82%, lo que indica que es el objeto que el algoritmo considera más difícil de detectar. Esto es curioso, ya que para un ser humano, se podría pensar que el objeto más difícil es la pistola, debido a su alta variabilidad intraclase.

4.4.2. Análisis del desempeño por objeto

En lo que respecta a cada clasificador particular, el resumen de los resultados es el siguiente: para el caso del clasificador de Gun, el mejor clasificador obtuvo un desempeño de 99%, mientras que el caso de Shuriken obtuvo un desempeño de 99% y el de Razor fue de 82%. Tanto la clase Razor como la Shuriken presentaron un promedio de falsos positivos entre un 0% y un 1%, mientras que en el caso del de Guns fue sustancialmente mayor.

Con los datos anteriores se puede concluir que el algoritmo clasifica con mejor rendimiento las shurikens y con peor rendimiento las hojas de afeitar. No obstante, hay varios puntos a considerar en la clasificación de cada objeto, los cuales se detallarán a continuación.

(i) Los clasificadores de Razor y Shuriken presentaron un nivel de casi 100% de TNR, lo que indica que para el algoritmo le resulta más fácil distinguir estos objetos de cualquier otro. Esto puede explicarse debido a la simetría de ambos objetos, lo cual

le entrega una mayor información (ya que ésta se encuentra repetida) al momento de realizar el Bag of Words.

(ii) El clasificador de Gun tiene un porcentaje bajo de TNR en comparación a los otros clasificadores, lo que indica que el algoritmo confunde varios objetos y cree que son pistolas. Esto puede explicarse al hecho de que tantos los otros objetos como las pistolas no tienen estructura definida, siendo éstas últimas más difíciles de diferenciar de cualquier otro objeto.

(iii) El clasificador de Razor posee el porcentaje más bajo de TPR, lo que puede deberse al hecho de que las hojas de afeitar pueden ser ocluídas más fácilmente en una imagen con varios objetos, ya que su intensidad de gris es más tenue comparándola con las shurikens y pistolas, que son más oscuras. Además el hecho de ser plana puede ser otro factor que complica al algoritmo. Esto puede provocar que el algoritmo en ocasiones no encuentre la hoja de afeitar (sobre todo en casos en donde ésta se encuentra de lado), realizando una clasificación errónea de la imagen.

(iv) Los clasificadores de Gun y Shuriken presentan los más altos porcentajes de TPR, sobre todo éste último en el promedio. Esto se puede explicar por la misma razón anterior, es decir, dado que son imágenes más oscuras, son más reconocibles por el algoritmo.

Con esto se puede concluir que la clase Shuriken reúne las mejores características para que su desempeño sea el más alto, vale decir: su simetría y poca oclusión.

Con los puntos anteriores, no basta con decir que el algoritmo funciona "mejor" con algunos objetos que con otros, sino que para cada objeto tienen un comportamiento diferente. No obstante, al momento de juntar los 3 clasificadores en uno, los resultados cambian ligeramente, lo que se observará en la siguiente sección.

4.5. Desempeño general del clasificador

Para la evaluación general del clasificador se buscó maximizar los porcentajes de detección de la matriz de confusión, considerando la etapa de score validation descrita en la

sección 3.3.1. El objetivo principal es maximizar la diagonal del clasificador, ya que esto implica que la cantidad de TP y TN es alta (vale decir, muchos objetos han sido clasificados correctamente), y por ende, la cantidad de FP y FN es baja (pocos errores de clasificación). En el caso de que resultados tengan promedios similares en su diagonal, se prevalece el clasificador que tenga una menor cantidad de falsos positivos, lo que implica una mayor precisión en los clasificadores binarios, ya que en este contexto es preferible que clasifique como peligroso un objeto que no lo es, a que el algoritmo pase por alto un objeto que si es peligroso. También se buscó el mejor desempeño en cada clasificador particular, sin sesgarse a sólo un tipo de objeto y pese a que el rendimiento general podría ser mayor.

4.5.1. Resultados del desempeño general del clasificador

Como métrica de mejor desempeño se tomó el promedio de la diagonal obtenida en la matriz de confusión, además de considerar los porcentajes más altos de cada clasificador particular.

Para este experimento se tomaron los mejores parámetros de cada clasificador, descritos en la tabla 4.13, considerando todas las permutaciones posibles. Para buscar los mejores parámetros en conjunto (y disminuir en alguna forma la cantidad de experimentos posibles) se utilizó el set de validación, de los cuales se escogieron los parámetros que obtuvieron una diagonal mayor al 89%, que luego fueron testeados con el set de testing, mostrando el resultado final.

Para el parámetro de la cantidad de árboles n del random Forest, se decidió probar con 2000 y 4000, árboles para todos los clasificadores, evitando demasiadas combinaciones. Se debe considerar que cada experimento demoraba 1 hora en promedio, por lo cual disminuir la cantidad de experimentos era fundamental. Un resumen de los experimentos realizados aparece a continuación:

- $3 \cdot 3 \cdot 3 = 27$ experimentos variando LBP
- $2 \cdot 2 \cdot 2 = 8$ experimentos variando SIFT
- 2 experimentos variando cantidad n de árboles.

El total de experimentos realizados fue de $27 \cdot 8 \cdot 2 = 432$ experimentos con el set de validación, de los cuales sólo los mejores se probaron con el set de Testing.

Luego de realizar todos los experimentos se obtuvieron los resultados mostrados en las siguientes tablas: la tabla 4.17 muestra los aciertos de los 5 mejores resultados obtenidos; la tabla 4.18 y la tabla 4.19 muestran la matriz de confusión final con el mejor rendimiento logrado para cada una de las 4 clases: (Razor, Shuriken, Gun y Others) y los parámetros del experimento con mejor promedio respectivamente.

TABLE 4.17. Promedios de las 5 mejores clasificaciones

	BEST	**Best 2**	**Best 3**	**Best 4**	**Best 5**
Razor	0,8400	0,7900	0,7800	0,8000	0,8300
Shuriken	0,9200	0,9400	0,9100	0,9700	0,9600
Gun	0,9700	0,9500	0,9800	0,8900	0,9000
Otros	0,8700	0,9000	0,9000	0,9100	0,8800
Promedio	**0,9000**	0,8950	0,8925	0,8925	0,8925

TABLE 4.18. Matriz de confusión para el mejor clasificador

	Razors	Shurikens	Guns	NoClass
Razors	84	0	0	16
Shurikens	0	92	0	8
Guns	0	0	97	3
NoClass	0	0	52	348

TABLE 4.19. Parámetros del mejor clasificador general.

	Razor	**Shuriken**	**Gun**
N_Tree	2000	4000	2000
Sigma	2	6	8
K SIFT	400	200	100
K LBP	100	100	200

Como se aprecia en la tabla 4.17, el mejor clasificador obtuvo un rendimiento en la diagonal del 90%, en donde para cada clasificador el resultado fue el siguiente, luego de realizar el procedimiento de Score Validation:

- Razor: $TP = 84\%$ $TN = 100\%$
- Shuriken: $TP = 92\%$ $TN = 100\%$
- Gun: $TP = 97\%$ $TN = 87\%$

Estos resultados muestran coherencia con los obtenidos en cada clasificador particular, ya que si se observa el anexo A, los parámetros utilizados coinciden con los clasificadores que tuvieron un desempeño mayor al 89% en el promedio.

Por último, se observó que el parámetro de puntos SIFT era determinante al momento de unir los 3 clasificadores, ya que influía de forma considerable cuántas imágenes entraban en conflicto, y por ende, el desempeño final. Los puntos SIFT que obtuvieron un mejor desempeño se muestran en la tabla 4.20.

TABLE 4.20. Tabla con el mejor parámetro de K puntos SIFT tras la unión de clasificadores

	Razor	**Shuriken**	**Gun**
K SIFT	400	200	100

4.5.2. Análisis del desempeño general del clasificador

Los resultados obtenidos considerando la etapa de Score Validation mostraron una diferencia con respecto al uso de cada clasificador por separado. Como primera conclusión, se descubrió que existe otro parámetro que es influenciado por los parámetros de cada clasificador, el cual es el nivel de "confianza" que cada clasificador tiene con respecto al objeto que busca, o en otras palabras, el promedio de hojas del árbol que clasifican al objeto. Diferentes configuraciones aumentan o disminuyen el nivel de confianza por objeto, lo que causa que en los casos en que hay conflicto (vale decir, que una misma imagen sea clasificada como 2 objetos a la vez) la decisión final sea diferente.

Otra conclusión importante es que sólo ocurrieron 2 tipos de conflicto: la clase Gun con la clase Shuriken y la clase Gun con la clase Razor. Nunca se registró un conflicto entre las shurikens y hojas de afeitar, o que un objeto fuera clasificado como los 3 a la vez. Esto puede deberse al hecho de que las pistolas poseen una estructura más compleja que los otros 2 objetos, que son simétricos. Para el algoritmo es más fácil confundirse cuando uno de los objetos es más complicado de describir (como en este caso la pistola). Esto mismo es la causa, como ya se explicó, de que las pistolas tuvieran una mayor cantidad de FP, puesto que pueden confundirse más fácilmente con los no-objetos.

TABLE 4.21. Desempeño de cada algoritmo por clase, luego de la etapa de Score Validaton

	Gun	**Shuriken**	**Razor**	**Other**	**Total**	**Ranking**
Basico	54.0	0.0	30.0	64.0	37.0	7
Sin filtro	72.0	0.0	40.0	94.0	51.5	6
Mediana	80.0	2.0	48.0	91.0	55.3	5
Sin LBP	86.0	92.0	96.0	68.5	85.6	3
Sin SIFT	4.0	0.0	2.0	74.5	20.1	8
SVM	74.0	74.0	48.0	49.0	61.2	4
XASR+	91.0	99.0	71.0	88.0	87.5	2
Completo	84.0	92.0	97.0	87.0	90.0	1

Por otro lado, el desarrollo de esta metodología muestra que al agregar una forma más compacta de describir las imágenes (por medio del modelo BoW) sí presenta un alza en el desempeño total de clasificación. Esto se puede observar en la tabla 4.21, que muestra los resultados de la detección de cada una de las clases (luego de la etapa de Score Validation) para cada algoritmo descrito en la sección 4.3.

En este análisis se agregaron los resultados del algoritmo XASR+ (Mery, Svec, and Arias, 2015), testeado con la misma base de datos y las mismas condiciones que los algoritmos anteriores. Se puede apreciar que el algoritmo desarrollado es el que posee el mejor ranking, superando incluso al algoritmo XASR+ que ya ha sido utilizado en radiografías de equipajes. La tabla 4.22 muestra los precision y recall de la tabla anterior.

TABLE 4.22. Precision y Recall de cada algoritmo, luego de la etapa de Score Validation

	Gun		**Shuriken**		**Razor**		**Total**	**Ranking**
	Prec	**Rec**	**Prec**	**Rec**	**Prec**	**Rec**		
Basico	0,80	0,30	1,00	0	1,00	0,54	37,0	7
Sin filtro	0,96	0,40	1,00	0	1,00	0,72	51,5	6
Mediana	0,94	0,48	1,00	0,02	1,00	0,80	55,3	5
Sin LBP	0,83	0,96	1,00	0,92	1,00	0,86	85,6	3
Sin SIFT	0,90	0,02	0,97	0	0,97	0,04	20,1	8
SVM	0,77	0,48	0,98	0,74	0,98	0,74	61,2	4
XASR	0,90	0,87	0,74	1,00	0,64	0,97	88,5	2
Completo	0,88	0,99	1,00	0,92	1,00	0,82	90,0	1

5. CONCLUSIONES Y TRABAJOS FUTUROS

El aporte de esta investigación es un avance en el estudio de la detección de objetos en imágenes de rayos X. Los resultados muestran que la metodología utilizada obtiene un resultado general del 90% en la diagonal de la matriz de confusión, lo cual es aceptable en el estado del arte, ya que trabajos recientes que también utilizan BoW en equipajes han logrado un 97% de TPR en pistolas y 89% en botellas (Flitton et al., 2015). También, dentro de una comparación de diferentes algoritmos que utilizan la misma base de datos, esta propuesta se ubicó en el 5to lugar de 10 propuestas (Mery et al., 2016), estando en primer lugar variaciones de Deep Learning (Flitton et al., 2015) y siendo superado por una corta distancia por Sparse KNN (Mery, Svec, and Arias, 2015).

En lo que respecta al desempeño de cada clasificador, se observa que el objeto que presenta un mejor rendimiento general fueron las shurikens, debido principalmente a su morfología simétrica. Por otro lado, el objeto que presenta una menor cantidad de TP fueron las hojas de afeitar. Las razones de esto son 2 principalmente: debido a su tamaño es muy fácil que el objeto pueda estar ocluído sobre otro objeto en la imagen de test, lo que otorga una dificultad adicional a la clasificación, sobre todo a la obtención de descriptores SIFT. En lo que respecta a los falsos positivos, se observó una clara diferencia entre las pistolas y los otros objetos, lo que se concluye que se debe a la simetría de objetos: dado que los otros 2 objetos son simétricos, su clasificación es mas sencilla si se compara con la de las pistolas. Esto no puede corroborarse hasta tener una base de datos con otro objeto no simétrico con el cual se pueda comparar.

En los resultados finales se buscaron los parámetros que, además de maximizar la diagonal, mantuviera un promedio particular alto para cada clasificador, de modo de que el algoritmo completo mantuviera una homogeneidad en la detección de cada objeto.

Cada fase del algoritmo desarrollado puede separarse en varios procesos, en donde cada una posee su propia relevancia para obtener el resultado esperado, aportando de una u otra manera al resultado final. Dado que las etapas de Training y Testing son relativamente similares, se enfocarán las conclusiones en cada una de las fases nombradas a continuación.

- Pre-procesamiento de imágenes:

  Sin duda esta etapa corresponde a una de las más relevantes para el desempeño final del algoritmo, dado que permite una detección más eficiente de los puntos SIFT necesarios para representar cada imagen.

  Se debe destacar la relevancia del valor del σ en el filtro Gaussiano. Como se apreció anteriormente, este valor es muy importante para lograr un desempeño aceptable para cada objeto, además de ser un parámetro determinante al momento de la evaluación de resultados. Se debe considerar que para cada clasificador (y por ende, cada objeto), el valor de σ es diferente. Por ejemplo, para el caso de los clasificadores de Shuriken y Guns, valores de σ inferiores a 4 lograban un desempeño muy bajo, incluso llegando a casos en que el algoritmo no era capaz de detectar ninguna shuriken. Por otro lado, para el caso del clasificador de Razor, valores de sigma mayores a 3 disminuían considerablemente el desempeño de éste gradualmente. Este factor se debe tener en cuenta al momento de agregar un nuevo objeto al clasificador, considerando un estudio de ajuste de parámetros antes de probarlo en un ambiente más real.

- Extracción de puntos SIFT y creación de sus K representantes:

  Sin duda este es el núcleo principal del modelo Bag of Words que se crea posteriormente. La característica de ser invariantes a la rotación y a la ubicación es un punto clave, dado que permite detectar objetos independiente de la forma en que se encuentren (ya sea rotados, de lado, etc.)

  El ajuste de parámetros mostró que la cantidad K de puntos SIFT utilizados no posee gran relevancia en el desempeño particular de cada clasificador. No obstante, al momento del desempeño general, obtiene una relevancia más notoria, ya que influye directamente en el grado de certeza de cada clasificador.

  Por otro lado, se observó una leve alza de desempeño en la detección a medida que el valor de K aumentaba. No obstante, cuanto mayor sea la cantidad de puntos SIFT utilizados, el tiempo de ejecución del algoritmo aumentaba.

Los experimentos mostraron que el tiempo promedio de la ejecución del K-MEANS era de 20 minutos en el Training, si se utilizaba el pc más potente de los 2 usados en este trabajo. Para elegir los valores de K adecuados se consideró en qué punto el desempeño del algoritmo se mantenía. Debido a esto, se utilizaron valores no mayores a $K = 400$, ya que a partir de ese punto el desempeño se mantenía.

- Extracción de descriptores LBP y creación de sus K representantes:
  Pese a que la cantidad K de representantes LBP no posee la misma importancia que los casos anteriores, si se debe destacar que no bastan los puntos SIFT para tener un descriptor potente, sobre todo para detectar pistolas. Este descriptor permite obtener la textura del objeto, dato relevante para su representación.
  Experimentos del análisis de sensibilidad mostraron que existía una diferencia entre utilizar o no este descriptor, donde el primer caso muestra un leve mejor desempeño que el segundo. No obstante, la cantidad de K representantes es independiente del desempeño, ya que como muestran los resultados, no existe una notoria tendencia entre diferentes valores de LBP.

- Entrenamiento con Random Forest:
  Sin duda el entrenamiento es el *core* del algoritmo. Como se dijo anteriormente, valores de 2000 y 4000 fueron utilizados, dado que valores mayores aumentaban considerablemente el tiempo de ejecución del algoritmo, sin mejorar el porcentaje de desempeño.

Respecto al desempeño general del algoritmo, éste fue de un 90%, considerando los 4 objetos ya mencionados en un solo clasificador. Hay que considerar que, dependiendo de los parámetros utilizados, la tendencia era hacia uno u otro objeto, por ejemplo, si la cantidad de puntos SIFT en una pistola aumentaba con respecto a los de una hoja de afeitar, entonces una mayor cantidad de hojas de afeitar eran confundidas con pistolas, o en otras palabras, el

clasificador de Guns tenía más certeza de que la imagen es una pistola que el de Razor. Estos cambios no seguían una tendencia definida.

Este comportamiento nos da la conclusión de que cada clasificador depende indirectamente de los otros, aumentando o disminuyendo su grado de certeza con cada objeto. Se debe tener esto en cuenta, sobre todo si se agrega un nuevo objeto al algoritmo.

Una de las primeras conclusiones generales que se pueden obtener es que el uso del modelo Bag of Words utilizando descriptores SIFT y LBP es adecuado, ya que el rendimiento logrado es alto. Además, el algoritmo es eficiente, ya que los tiempos de ejecución, sobre todo en el Testing (que es el caso más importante), es bajo por cada imagen. También es versátil, ya que su implementación permite agregar nuevos clasificadores sin tener que modificar la estructura completa del código.

En lo que respecta a los descriptores utilizados, se puede concluir que éstos permiten representar de una forma adecuada la imagen, entregando la información suficiente para lograr una clasificación satisfactoria. Esto lleva a inferir que, para el caso de imágenes de rayos X, no es necesario obtener una mayor cantidad de información para problemas de esta naturaleza.

Sobre el modelo Bag of Words, éste muestra un alto grado de desempeño, por lo cual se puede concluir que no toda la información obtenida de las imágenes es relevante para obtener una clasificación decente, ya que hay mucha información repetida que puede empeorar o confundir al algoritmo. Sobre este mismo punto, el modelo permite disminuir la cantidad de *features* utilizadas, lo que disminuye el tiempo en que se demora en procesar cada imagen. Una de las ventajas de usar este modelo es que el tiempo de ejecución es considerablemente menor al caso de utilizar toda la información (puntos SIFT y LBP) que la imagen entrega.

El algoritmo Random Forest de clasificación utilizado mostró un desempeño importante y eficiente, por lo cual se concluye que este método trabaja bien en imágenes de rayos X de este estilo, no necesitando el uso de otro clasificador anexo o complementario. Como aporte a este punto, se probó el algoritmo SVM como otro clasificador, pero éste obtuvo un desempeño mediocre (como se observa en el análisis de sensibilidad) ya que prácticamente todas las imágenes llegan a ser clasificadas como no-objeto.

La limitación principal de este trabajo es el tipo de imágenes que necesita para ser utilizado, ya que sólo se dedica a clasificar objetos, y no a detectar la posición donde se encuentra el mismo, lo que sería más cercano a un caso real.

Hay que considerar que los resultados obtenidos podrían tener mejoras si se utiliza un sistema automático de calibración de parámetros o una métrica de unión de clasificadores más grande que sólo utilizar el porcentaje de puntuación del Random Forest. No obstante, se debe considerar el hecho de que los clasificadores interactúan entre sí, por lo cual el diseño de un sistema de calibración simple puede llegar a ser complejo.

5.1. Trabajos futuros

Con este trabajo se pretende formar un algoritmo que sirva de base sólida para desarrollar un detector completo de objetos peligrosos en radiografías. Los resultados de esta metodología son aceptables en el contexto estudiado, además de que es posible llevar esta metodología a la práctica en un software comercial. No obstante, como se mencionó anteriormente, este algoritmo sólo trabaja con la clasificación de objetos que ya han sido encontrados, sin considerar la detección de los mismos. En un caso real, por ejemplo en un aeropuerto, los equipajes vienen con una cantidad significativa de objetos dentro de la misma, los cuales muchas veces se confunden o mimetizan, incluso para el ojo humano, como se muestra en la radiografía de un equipaje mostrado en la figura 5.1.

Considerando lo anterior, se proponen 3 trabajos futuros a partir de esta tesis. El primero consiste en mejorar aún más el rendimiento del algoritmo desarrollado, esperando llegar a un rendimiento mayor al alcanzado (idealmente un 99%). El segundo trabajo pro-puesto es utilizar este algoritmo con una base de datos más grande, o agregar un nuevo objeto a detectar. Esto es posible ya que su construcción y estructura permite de forma fácil agregar un nuevo clasificador y compararlo con los estudiados.

El tercer trabajo, y más importante, es considerar el desarrollo de un algoritmo de **detección de objetos**, complementario al desarrollado en esta tesis. Este trabajo futuro podría utilizar el algoritmo desarrollado en esta tesis como segunda etapa, de tal forma de que, en

primera instancia, se detecten los posibles objetos dentro de una imagen como la anterior, para posteriormente clasificarlos como peligroso o no. De esta forma, el software final sería un programa capaz de trabajar en un ambiente más real.

FIGURA 5.1. Radiografía de una mochila que contiene múltiples objetos.

BIBLIOGRAFIA

Abidi, B. R., Zheng, Y., Gribok, A. V., Abidi, M., et al. (2006). Improving weapon detection in single energy x-ray images through pseudocoloring. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *36*(6), 784–796.

Abusaeeda, O., Evans, J., Downes, D., & Chan, J. (2011). View synthesis of kdex imagery for 3d security x-ray imaging.

Armistead, R. A. (1998, November 17). *Single beam photoneutron probe and x-ray imaging system for contraband detection and identification.* Google Patents. (US Patent 5,838,759)

Baniukiewicz, P. (2014). Automated defect recognition and identification in digital radiography. *Journal of Nondestructive Evaluation*, *33*(3), 327–334.

Bastan, M., Byeon, W., & Breuel, T. (2013). Object recognition in multi-view dual x-ray images. In *British machine vision conference bmvc.*

Baştan, M., Yousefi, M. R., & Breuel, T. M. (2011). Visual words on baggage x-ray images. In *Computer analysis of images and patterns* (pp. 360–368).

Bastian, C. C. von, Schwaninger, A., & Michel, S. (2008). Do multi-view x-ray systems improve x-ray image interpretation in airport security screening?

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *24*(4), 509–522.

Berclaz, J., Fleuret, F., Türetken, E., & Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *33*(9), 1806–1819.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Brosnan, T., & Sun, D.-W. (2002). Inspection and grading of agricultural and food products by computer vision systemsa review. *Computers and electronics in agriculture*, *36*(2), 193–213.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on machine learning* (pp. 161–168).

Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, eccv* (Vol. 1, pp. 1–2).

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer vision and pattern recognition, 2005. cvpr 2005. ieee computer society conference on* (Vol. 1, pp. 886–893).

Damashek, A., & Doherty, J. (n.d.). Detecting guns using parametric edge matching.

Eberhardt, J., Rainey, S., Stevens, R., Sowerby, B., & Tickner, J. (2005). Fast neutron radiography scanner for the detection of contraband in air cargo containers. *Applied Radiation and Isotopes*, *63*(2), 179–188.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *32*(9), 1627–1645.

Flitton, G., Mouton, A., & Breckon, T. P. (2015). Object classification in 3d baggage security computed tomography imagery using visual codebooks. *Pattern Recognition*, *48*(8), 2489–2499.

Franzel, T., Schmidt, U., & Roth, S. (2012). *Object detection in multi-view x-ray images*. Springer.

Gini, C. (1912). Variability and mutability, contribution to the study of statistical distribution and relaitons. *Studi Economico-Giuricici della R*.

Grill-Spector, K. (2003). The neural basis of object perception. *Current opinion in neurobiology*, *13*(2), 159–166.

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Heinrich, W. (1988). *Automated inspection of castings using x-ray testing*. Unpublished doctoral dissertation, PhD thesis, Institute for Measurement and Automation, Faculty of Electrical Engineering, Technical University of Berlin, 1988.(in German).

Huang, Q., Wu, Y., Baruch, J., Jiang, P., & Peng, Y. (2009). A template model for defect simulation for evaluating nondestructive testing in x-radiography. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *39*(2), 466–475.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *24*(7), 881–892.

Kullback, S. (1987). Letter to the editor: the kullback-leibler distance.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomforest. *R news*, *2*(3), 18–22.

Lim, J. S. (1990). Two-dimensional signal and image processing. *Englewood Cliffs, NJ, Prentice Hall, 1990, 710 p.*, *1*.

Liu, D., & Wang, Z. (2008). A united classification system of x-ray image based on fuzzy rule and neural networks. In *Intelligent system and knowledge engineering, 2008. iske 2008. 3rd international conference on* (Vol. 1, pp. 717–722).

Lopes, A. P., Avila, S. E. de, Peixoto, A. N., Oliveira, R. S., & A Araujo, A. de. (2009). A bag-of-features approach based on hue-sift descriptor for nude detection. In *Signal processing conference, 2009 17th european* (pp. 1552–1556).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91–110.

Maturana, D., Mery, D., & Soto, A. (2011). Face recognition with decision tree-based local binary patterns. In *Computer vision–accv 2010* (pp. 618–629). Springer.

Mazess, R. B., Barden, H. S., Bisek, J. P., & Hanson, J. (1990). Dual-energy x-ray absorptiometry for total-body and regional bone-mineral and soft-tissue composition. *The American journal of clinical nutrition*, *51*(6), 1106–1112.

Mery, D. (2011). Automated detection in complex objects using a tracking algorithm in multiple x-ray views. In *Computer vision and pattern recognition workshops (cvprw), 2011 ieee computer society conference on* (pp. 41–48).

Mery, D. (2015). Inspection of complex objects using multiple-x-ray views. *Mechatronics, IEEE/ASME Transactions on*, *20*(1), 338–347.

Mery, D., & Berti, M. A. (2003). Automatic detection of welding defects using texture features. *Insight-Non-Destructive Testing and Condition Monitoring*, *45*(10), 676–681.

Mery, D., & Filbert, D. (2002). Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. *Robotics and Automation, IEEE Transactions on*, *18*(6), 890–901.

Mery, D., Mondragon, G., Riffo, V., & Zuccar, I. (2013). Detection of regular objects in baggage using multiple x-ray views. *Insight-Non-Destructive Testing and Condition Monitoring*, *55*(1), 16–20.

Mery, D., Riffo, V., Zscherpel, U., Mondragn, G., Lillo, I., Zuccar, I., et al. (2015). *Gdxray - the grima database of x-ray images.* Disponible en `http://dmery.ing.puc.cl/index.php/material/gdxray/`. Departmento de Ciencias de la Computación, Universidad Católica de Chile, en colaboración con Institute for Materials Research and Testing (BAM), Berlin.

Mery, D., Riffo, V., Zuccar, I., & Pieringer, C. (2013). Automated x-ray object recognition using an efficient search algorithm in multiple views. In *Computer vision and pattern recognition workshops (cvprw), 2013 ieee conference on* (pp. 368–374).

Mery, D., Svec, E., & Arias, M. (2015). Object recognition in baggage inspection using adaptive sparse representations of x-ray images. In *Image and video technology* (pp. 709–720). Springer.

Mery, D., Svec, E., Arias, M., Riffo, V., Saavedra, J. M., & Banerjee, S. (2016). Modern computer vision techniques for x-ray testing in baggage inspection. *Enviado a IEEE Trans. on Cybernetics, Man, System (Marzo 2016, Primera revision Mayo 2016)*.

Michel, S., Koller, S. M., De Ruiter, J. C., Moerland, R., Hogervorst, M., & Schwaninger, A. (2007). Computer-based training increases efficiency in x-ray image interpretation

by aviation security screeners. In *Security technology, 2007 41st annual ieee international carnahan conference on* (pp. 201–206).

Mondragón González, G., et al. (2013). Metodología para el desarrollo de un clasificador de múltiples vistas de radiografías: aplicación para la búsqueda de armas pequeñas dentro de equipajes.

Mouton, A., & Breckon, T. P. (2015). A review of automated image understanding within 3d baggage computed tomography security screening. *Journal of X-ray science and technology*, *23*(5), 531–555.

Murphy, E. E. (1989). A rising war on terrorists. *Spectrum, IEEE*, *26*(11), 33–36.

Murray, N., & Riordan, K. (1995). Evaluation of automatic explosive detection systems. In *Security technology, 1995. proceedings. institute of electrical and electronics engineers 29th annual 1995 international carnahan conference on* (pp. 175–179).

Nercessian, S., Panetta, K., & Agaian, S. (2008). Automatic detection of potential threat objects in x-ray luggage scan images. In *Technologies for homeland security, 2008 ieee conference on* (pp. 504–509).

Nilsback, M.-E., & Zisserman, A. (2006). A visual vocabulary for flower classification. In *Computer vision and pattern recognition, 2006 ieee computer society conference on* (Vol. 2, pp. 1447–1454).

Oertel, C., & Bock, P. (2006). Identification of objects-of-interest in x-ray images. In *Applied imagery and pattern recognition workshop, 2006. aipr 2006. 35th ieee* (pp. 17–17).

Ojala, T., Pietikainen, M., & Harwood, D. (1994). Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Pattern recognition, 1994. vol. 1-conference a: Computer vision & image processing., proceedings of the 12th iapr international conference on* (pp. 582–585).

Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. (2012). Cats and dogs. In *Computer vision and pattern recognition (cvpr), 2012 ieee conference on* (pp. 3498–3505).

Pizarro, L., Mery, D., Delpiano, R., & Carrasco, M. (2008). Robust automated multiple view inspection. *Pattern Analysis and Applications*, *11*(1), 21–32.

Precision, C. O. P. R. (n.d.). Precision and recall.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Riffo, V., & Mery, D. (2012). Active x-ray testing of complex objects. *Insight-Non-Destructive Testing and Condition Monitoring*, *54*(1), 28–35.

Silva, R. Ricardo da, & Mery, D. (2007). The state of the art of weld seam radiographic testing: Part ii, pattern recognition. *Materials evaluation*, *65*(8), 833–838.

Singh, M., & Singh, S. (2005). Optimizing image enhancement for screening luggage at airports. In *Computational intelligence for homeland security and personal safety, 2005. cihsps 2005. proceedings of the 2005 ieee international conference on* (pp. 131–136).

Solem, J. E. (2015). *Sift python implementation - solem's vision blog.* Disponible en `http://www.janeriksolem.net/2009/02/sift-python-implementation.html`.

Strecker, H. (1998). Automatic detection of explosives in airline baggage using elastic x-ray scatter. *Medicamundi*, *42*, 30–33.

Tsai, C.-F. (2012). Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, *2012*.

Van De Sande, K. E., Gevers, T., & Snoek, C. G. (2010). Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *32*(9), 1582–1596.

Viola, P., & Jones, M. (2001a). Rapid object detection using a boosted cascade of simple features. In *Computer vision and pattern recognition, 2001. cvpr 2001. proceedings of the 2001 ieee computer society conference on* (Vol. 1, pp. I–511).

Viola, P., & Jones, M. (2001b). Robust real-time object detection. *International Journal of Computer Vision*, *4*, 51–52.

Vyborny, C. J., & Giger, M. L. (1994). Computer vision and artificial intelligence in mammography. *AJR. American journal of roentgenology*, *162*(3), 699–708.

Wales, A., Halbherr, T., & Schwaninger, A. (2009). Using speed measures to predict performance in x-ray luggage screening tasks. In *Security technology, 2009. 43rd annual 2009 international carnahan conference on* (pp. 212–215).

Wei, W., & Hanbo, F. (2011). Traffic accident automatic detection and remote alarm device. In *Electric information and control engineering (iceice), 2011 international conference on* (pp. 910–913).

Wikipedia. (2015a). *Airport security.* Disponible en `https://en.wikipedia.org/wiki/Airport_security#/media/File:VTBS-luggage_screening.JPG`.

Wikipedia. (2015b). *árbol de decisión.* Disponible en `https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n`.

Wikipedia. (2015c). *Precision and recall.* Disponible en `https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg`.

Zentai, G. (2010). X-ray imaging for homeland security. *International Journal of Signal and Imaging Systems Engineering*, *3*(1), 13–20.

Zhang, J., Marszałek, M., Lazebnik, S., & Schmid, C. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, *73*(2), 213–238.

Zhang, Z., Lipton, A. J., Brewer, P. C., Chosak, A. J., Haering, N., Myers, G. W., et al. (2005, May 31). *Human detection and tracking for security applications.* Google Patents. (US Patent App. 11/139,986)

ANEXOS

ANEXO A. TABLAS DE AJUSTE DE PARÁMETROS POR CLASIFICADOR

En esta sección se muestran las tablas con los experimentos particulares para los 3 clasificadores entrenados, buscando el ajuste de parámetros de cada uno. Estos experimentos fueron realizados sin el proceso de validación de Scores y utilizando el set de datos de Validación.

Las primeras 4 columnas muestran los valores de σ, K de SIFT, K de LBP y cantidad de árboles utilizados. Las siguientes 6 columnas muestran la cantidad de TP, TN, FP y FN obtenidos (los primeros en porcentajes). La siguiente columna muestra el promedio en porcentaje de TP y TN. La última muestra los Precision y Recall para cada experimento.

Tabla para Razors

TABLE A.1. Tabla de rendimiento para el clasificador de Razors

σ	**Tree**	**SIFT**	**LBP**	**TPR**	**TNR**	**TP**	**TN**	**FP**	**FN**	**Rendimiento**	**Recall**	**Precision**
0	2000	100	50	0,42	1,00	42	600	0	58	**0,710**	0,42	1,00
			100	0,44	1,00	44	600	0	56	**0,720**	0,44	1,00
			200	0,40	1,00	40	600	0	60	**0,700**	0,40	1,00
		200	50	0,44	1,00	44	600	0	56	**0,720**	0,44	1,00
			100	0,46	1,00	46	600	0	54	**0,730**	0,46	1,00
			200	0,48	1,00	48	600	0	52	**0,740**	0,48	1,00
		400	50	0,62	0,99	62	594	6	38	**0,805**	0,62	0,91
			100	0,56	0,99	56	594	6	44	**0,775**	0,56	0,90
			200	0,58	1,00	58	600	0	42	**0,790**	0,58	1,00
1	2000	100	50	0,52	1,00	52	600	0	48	**0,760**	0,52	1,00
			100	0,64	0,99	64	594	6	36	**0,815**	0,64	0,91
			200	0,46	1,00	46	600	0	54	**0,730**	0,46	1,00
		200	50	0,66	0,99	66	594	6	34	**0,825**	0,66	0,92
			100	0,74	1,00	74	600	0	26	**0,870**	0,74	1,00
			200	0,72	1,00	72	600	0	28	**0,860**	0,72	1,00
		400	50	0,80	0,99	80	594	6	20	**0,895**	0,80	0,93
			100	0,84	1,00	84	600	0	16	**0,920**	0,84	1,00
			200	0,80	1,00	80	600	0	20	**0,900**	0,80	1,00

Tabla para Razors (continuación)

TABLE A.2. Tabla de rendimiento para el clasificador de Razors (continuación)

σ	**Tree**	**SIFT**	**LBP**	**TPR**	**TNR**	**TP**	**TN**	**FP**	**FN**	**Rendimiento**	**Recall**	**Precision**
2	2000	100	50	0,72	0,99	72	594	6	28	**0,855**	0,72	0,92
			100	0,60	0,99	60	594	6	40	**0,795**	0,60	0,91
			200	0,66	0,99	66	594	6	34	**0,825**	0,66	0,92
		200	50	0,80	0,99	80	594	6	20	**0,895**	0,80	0,93
			100	0,78	0,99	78	594	6	22	**0,885**	0,78	0,93
			200	0,80	0,99	80	594	6	20	**0,895**	0,80	0,93
		400	50	0,80	0,99	80	594	6	20	**0,895**	0,80	0,93
			100	0,84	0,98	84	588	12	16	**0,910**	0,84	0,88
			200	0,86	0,99	86	594	6	14	**0,925**	0,86	0,93
3	2000	100	50	0,46	0,98	46	588	12	54	**0,720**	0,46	0,79
			100	0,54	0,99	54	594	6	46	**0,765**	0,54	0,90
			200	0,36	0,99	36	594	6	64	**0,675**	0,36	0,86
		200	50	0,70	1,00	70	600	0	30	**0,850**	0,70	1,00
			100	0,74	1,00	74	600	0	26	**0,870**	0,74	1,00
			200	0,66	1,00	66	600	0	34	**0,830**	0,66	1,00
		400	50	0,80	1,00	80	600	0	20	**0,900**	0,80	1,00
			100	0,82	1,00	82	600	0	18	**0,910**	0,82	1,00
			200	0,82	0,99	82	594	6	18	**0,905**	0,82	0,93
4	2000	100	50	0,32	0,98	32	588	12	68	**0,650**	0,32	0,73
			100	0,38	0,99	38	594	6	62	**0,685**	0,38	0,86
			200	0,36	1,00	36	600	0	64	**0,680**	0,36	1,00
		200	50	0,42	0,99	42	594	6	58	**0,705**	0,42	0,88
			100	0,42	1,00	42	600	0	58	**0,710**	0,42	1,00
			200	0,44	1,00	44	600	0	56	**0,720**	0,44	1,00
		400	50	0,38	1,00	38	600	0	62	**0,690**	0,38	1,00
			100	0,56	0,99	56	594	6	44	**0,775**	0,56	0,90
			200	0,44	0,99	44	594	6	56	**0,715**	0,44	0,88
5	2000	100	50	0,32	1,00	32	600	0	68	**0,660**	0,32	1,00
			100	0,32	0,98	32	588	12	68	**0,650**	0,32	0,73
			200	0,22	0,99	22	594	6	78	**0,605**	0,22	0,79
		200	50	0,42	1,00	42	600	0	58	**0,710**	0,42	1,00
			100	0,46	1,00	46	600	0	54	**0,730**	0,46	1,00
			200	0,40	1,00	40	600	0	60	**0,700**	0,40	1,00
		400	50	0,22	1,00	22	600	0	78	**0,610**	0,22	1,00
			100	0,34	1,00	34	600	0	66	**0,670**	0,34	1,00
			200	0,38	1,00	38	600	0	62	**0,690**	0,38	1,00

Tabla para Razors (continuación)

TABLE A.3. Tabla de rendimiento para el clasificador de Razors (continuación)

σ	**Tree**	**SIFT**	**LBP**	**TPR**	**TNR**	**TP**	**TN**	**FP**	**FN**	**Rendimiento**	**Recall**	**Precision**
6	2000	100	50	0,36	1,00	36	600	0	64	**0,680**	0,36	1,00
			100	0,28	0,99	28	594	6	72	**0,635**	0,28	0,82
			200	0,24	0,99	24	594	6	76	**0,615**	0,24	0,80
		200	50	0,46	1,00	46	600	0	54	**0,730**	0,46	1,00
			100	0,36	1,00	36	600	0	64	**0,680**	0,36	1,00
			200	0,36	1,00	36	600	0	64	**0,680**	0,36	1,00
		400	50	0,36	1,00	36	600	0	64	**0,680**	0,36	1,00
			100	0,44	1,00	44	600	0	56	**0,720**	0,44	1,00
			200	0,30	1,00	30	600	0	70	**0,650**	0,30	1,00
7	2000	100	50	0,40	0,99	40	594	6	60	**0,695**	0,40	0,87
			100	0,34	0,99	34	594	6	66	**0,665**	0,34	0,85
			200	0,40	1,00	40	600	0	60	**0,700**	0,40	1,00
		200	50	0,46	0,99	46	594	6	54	**0,725**	0,46	0,88
			100	0,42	1,00	42	600	0	58	**0,710**	0,42	1,00
			200	0,44	0,99	44	594	6	56	**0,715**	0,44	0,88
		400	50	0,32	0,99	32	594	6	68	**0,655**	0,32	0,84
			100	0,36	0,99	36	594	6	64	**0,675**	0,36	0,86
			200	0,40	0,99	40	594	6	60	**0,695**	0,40	0,87
8	2000	100	50	0,36	0,98	36	588	12	64	**0,670**	0,36	0,75
			100	0,40	0,99	40	594	6	60	**0,695**	0,40	0,87
			200	0,38	0,99	38	594	6	62	**0,685**	0,38	0,86
		200	50	0,36	1,00	36	600	0	64	**0,680**	0,36	1,00
			100	0,50	0,99	50	594	6	50	**0,745**	0,50	0,89
			200	0,44	1,00	44	600	0	56	**0,720**	0,44	1,00
		400	50	0,38	1,00	38	600	0	62	**0,690**	0,38	1,00
			100	0,30	1,00	30	600	0	70	**0,650**	0,30	1,00
			200	0,48	1,00	48	600	0	52	**0,740**	0,48	1,00

Tabla para Shurikens

TABLE A.4. Tabla de rendimiento para el clasificador de Shurikens

σ	Tree	SIFT	LBP	TPR	TNR	TP	TN	FP	FN	Rendimiento	Recall	Precision
0	2000	100	50	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			100	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			200	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
		200	50	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			100	0,00	0,99	0	594	6	100	**0,495**	0,00	1,00
			200	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
		400	50	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			100	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			200	0,00	0,99	0	594	6	100	**0,495**	0,00	1,00
1	2000	100	50	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			100	0,00	0,99	0	594	6	100	**0,495**	0,00	1,00
			200	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
		200	50	0,02	1,00	2	600	0	98	**0,510**	0,02	1,00
			100	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			200	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
		400	50	0,00	0,99	0	594	6	100	**0,495**	0,00	1,00
			100	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			200	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
2	2000	100	50	0,00	1,00	0	600	0	100	**0,500**	0,00	1,00
			100	0,00	0,99	0	594	6	100	**0,495**	0,00	1,00
			200	0,00	0,99	0	594	6	100	**0,495**	0,00	1,00
		200	50	0,08	0,99	8	594	6	92	**0,535**	0,08	0,57
			100	0,06	0,99	6	594	6	94	**0,525**	0,06	0,50
			200	0,12	1,00	12	600	0	88	**0,560**	0,12	1,00
		400	50	0,10	1,00	10	600	0	90	**0,550**	0,10	1,00
			100	0,04	1,00	4	600	0	96	**0,520**	0,04	1,00
			200	0,08	1,00	8	600	0	92	**0,540**	0,08	1,00
3	2000	100	50	0,04	0,99	4	594	6	96	**0,515**	0,04	0,40
			100	0,12	1,00	12	600	0	88	**0,560**	0,12	1,00
			200	0,04	1,00	4	600	0	96	**0,520**	0,04	1,00
		200	50	0,18	1,00	18	600	0	82	**0,590**	0,18	1,00
			100	0,20	0,99	20	594	6	80	**0,595**	0,20	0,77
			200	0,32	1,00	32	600	0	68	**0,660**	0,32	1,00
		400	50	0,20	1,00	20	600	0	80	**0,600**	0,20	1,00
			100	0,14	1,00	14	600	0	86	**0,570**	0,14	1,00
			200	0,30	1,00	30	600	0	70	**0,650**	0,30	1,00
4	2000	100	50	0,54	1,00	54	600	0	46	**0,770**	0,54	1,00
			100	0,52	1,00	52	600	0	48	**0,760**	0,52	1,00

Tabla para Shurikens (continuación)

TABLE A.5. Tabla de rendimiento para el clasificador de Shurikens (continuación)

σ	Tree	SIFT	LBP	TPR	TNR	TP	TN	FP	FN	Rendimiento	Recall	Precision
			200	0,38	1,00	38	600	0	62	**0,690**	0,38	1,00
		200	50	0,54	1,00	54	600	0	46	**0,770**	0,54	1,00
			100	0,6	1,00	60	600	0	40	**0,800**	0,6	1,00
			200	0,76	1,00	76	600	0	24	**0,880**	0,76	1,00
		400	50	0,58	1,00	58	600	0	42	**0,790**	0,58	1,00
			100	0,38	1,00	38	600	0	62	**0,690**	0,38	1,00
			200	0,48	1,00	48	600	0	52	**0,740**	0,48	1,00
5	2000	100	50	0,70	1,00	70	600	0	30	**0,850**	0,70	1,00
			100	0,86	1,00	86	600	0	14	**0,930**	0,86	1,00
			200	0,72	1,00	72	600	0	28	**0,860**	0,72	1,00
		200	50	0,88	1,00	88	600	0	12	**0,940**	0,88	1,00
			100	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
			200	0,78	1,00	78	600	0	22	**0,890**	0,78	1,00
		400	50	0,74	1,00	74	600	0	26	**0,870**	0,74	1,00
			100	0,60	1,00	60	600	0	40	**0,800**	0,60	1,00
			200	0,68	1,00	68	600	0	32	**0,840**	0,68	1,00
6	2000	100	50	0,94	1,00	94	600	0	6	**0,970**	0,94	1,00
			100	0,88	1,00	88	600	0	12	**0,940**	0,88	1,00
			200	0,88	1,00	88	600	0	12	**0,940**	0,88	1,00
		200	50	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
			100	0,98	1,00	98	600	0	2	**0,990**	0,98	1,00
			200	1,00	1,00	100	600	0	0	**1,000**	1,00	1,00
		400	50	0,94	0,99	94	594	6	6	**0,965**	0,94	0,94
			100	0,90	1,00	90	600	0	10	**0,950**	0,90	1,00
			200	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
7	2000	100	50	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
			100	0,82	1,00	82	600	0	18	**0,910**	0,82	1,00
			200	0,88	1,00	88	600	0	12	**0,940**	0,88	1,00
		200	50	0,94	1,00	94	600	0	6	**0,970**	0,94	1,00
			100	1,00	0,99	100	594	6	0	**0,995**	1,00	0,94
			200	0,96	1,00	96	600	0	4	**0,980**	0,96	1,00
		400	50	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
			100	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
			200	0,78	1,00	78	600	0	22	**0,890**	0,78	1,00
8	2000	100	50	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
			100	0,84	1,00	84	600	0	16	**0,920**	0,84	1,00
			200	0,88	1,00	88	600	0	12	**0,940**	0,88	1,00
		200	50	0,88	1,00	88	600	0	12	**0,940**	0,88	1,00
			100	0,88	1,00	88	600	0	12	**0,940**	0,88	1,00
			200	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
		400	50	0,82	1,00	82	600	0	18	**0,910**	0,82	1,00
			100	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
			200	0,86	1,00	86	600	0	14	**0,930**	0,86	1,00

Tabla para Guns

TABLE A.6. Tabla de rendimiento para el clasificador de Guns

σ	**Tree**	**SIFT**	**LBP**	**TPR**	**TNR**	**TP**	**TN**	**FP**	**FN**	**Rendimiento**	**Recall**	**Precision**
0	2000	100	50	0,42	0,95	42	570	30	58	**0,685**	0,42	0,58
			100	0,34	0,95	34	570	30	66	**0,645**	0,34	0,53
			200	0,42	0,95	42	570	30	58	**0,685**	0,42	0,58
		200	50	0,28	0,96	28	576	24	72	**0,620**	0,28	0,54
			100	0,24	0,96	24	576	24	76	**0,600**	0,24	0,50
			200	0,28	0,96	28	576	24	72	**0,620**	0,28	0,54
		400	50	0,18	0,96	18	576	24	82	**0,570**	0,18	0,43
			100	0,14	0,96	14	576	24	86	**0,550**	0,14	0,37
			200	0,18	0,95	18	570	30	82	**0,565**	0,18	0,38
1	2000	100	50	0,44	0,92	44	552	48	56	**0,680**	0,44	0,48
			100	0,44	0,92	44	552	48	56	**0,680**	0,44	0,48
			200	0,46	0,91	46	546	54	54	**0,685**	0,46	0,46
		200	50	0,26	0,94	26	564	36	74	**0,600**	0,26	0,42
			100	0,30	0,94	30	564	36	70	**0,620**	0,30	0,45
			200	0,34	0,93	34	558	42	66	**0,635**	0,34	0,45
		400	50	0,14	0,94	14	564	36	86	**0,540**	0,14	0,28
			100	0,16	0,92	16	552	48	84	**0,540**	0,16	0,25
			200	0,20	0,92	20	552	48	80	**0,560**	0,20	0,29
2	2000	100	50	0,44	0,95	44	570	30	56	**0,695**	0,44	0,59
			100	0,66	0,94	66	564	36	34	**0,800**	0,66	0,65
			200	0,56	0,92	56	552	48	44	**0,740**	0,56	0,54
		200	50	0,28	0,94	28	564	36	72	**0,610**	0,28	0,44
			100	0,46	0,95	46	570	30	54	**0,705**	0,46	0,61
			200	0,54	0,93	54	558	42	46	**0,735**	0,54	0,56
		400	50	0,46	0,92	46	552	48	54	**0,690**	0,46	0,49
			100	0,38	0,93	38	558	42	62	**0,655**	0,38	0,48
			200	0,32	0,92	32	552	48	68	**0,620**	0,32	0,40
3	2000	100	50	0,82	0,90	82	540	60	18	**0,860**	0,82	0,58
			100	0,86	0,90	86	540	60	14	**0,880**	0,86	0,59
			200	0,90	0,86	90	516	84	10	**0,880**	0,90	0,52
		200	50	0,84	0,82	84	492	108	16	**0,830**	0,84	0,44
			100	0,80	0,87	80	522	78	20	**0,835**	0,80	0,51
			200	0,82	0,86	82	516	84	18	**0,840**	0,82	0,49
		400	50	0,70	0,90	70	540	60	30	**0,800**	0,70	0,54
			100	0,72	0,87	72	522	78	28	**0,795**	0,72	0,48
			200	0,72	0,87	72	522	78	28	**0,795**	0,72	0,48
4	2000	100	50	0,88	0,81	88	486	114	12	**0,845**	0,88	0,44
			100	0,92	0,79	92	474	126	8	**0,855**	0,92	0,42

Tabla para Guns (continuación)

TABLE A.7. Tabla de rendimiento para el clasificador de Guns (continuación)

σ	Tree	SIFT	LBP	TPR	TNR	TP	TN	FP	FN	Rendimiento	Recall	Precision
			200	0,88	0,81	88	486	114	12	**0,845**	0,88	0,44
		200	50	0,82	0,84	82	504	96	18	**0,830**	0,82	0,46
			100	0,82	0,77	82	462	138	18	**0,795**	0,82	0,37
			200	0,88	0,85	88	510	90	12	**0,865**	0,88	0,49
		400	50	0,80	0,85	80	510	90	20	**0,825**	0,80	0,47
			100	0,82	0,85	82	510	90	18	**0,835**	0,82	0,48
			200	0,86	0,86	86	516	84	14	**0,860**	0,86	0,51
5	2000	100	50	0,92	0,82	92	492	108	8	**0,870**	0,92	0,46
			100	0,98	0,85	98	510	90	2	**0,915**	0,98	0,52
			200	0,96	0,81	96	486	114	4	**0,885**	0,96	0,46
		200	50	0,84	0,83	84	498	102	16	**0,835**	0,84	0,45
			100	0,86	0,86	86	516	84	14	**0,860**	0,86	0,51
			200	0,82	0,88	82	528	72	18	**0,850**	0,82	0,53
		400	50	0,74	0,85	74	510	90	26	**0,795**	0,74	0,45
			100	0,76	0,89	76	534	66	24	**0,825**	0,76	0,54
			200	0,80	0,88	80	528	72	20	**0,840**	0,80	0,53
6	2000	100	50	0,92	0,82	92	492	108	8	**0,870**	0,92	0,46
			100	1,00	0,82	100	492	108	0	**0,910**	1,00	0,48
			200	0,94	0,83	94	498	102	6	**0,885**	0,94	0,48
		200	50	0,84	0,87	84	522	78	16	**0,855**	0,84	0,52
			100	0,94	0,84	94	504	96	6	**0,890**	0,94	0,49
			200	0,92	0,85	92	510	90	8	**0,885**	0,92	0,51
		400	50	0,86	0,89	86	534	66	14	**0,875**	0,86	0,57
			100	0,88	0,86	88	516	84	12	**0,870**	0,88	0,51
			200	0,88	0,89	88	534	66	12	**0,885**	0,88	0,57
7	2000	100	50	1,00	0,83	100	498	102	0	**0,915**	1,00	0,50
			100	0,98	0,82	98	492	108	2	**0,900**	0,98	0,48
			200	0,96	0,85	96	510	90	4	**0,905**	0,96	0,52
		200	50	0,88	0,87	88	522	78	12	**0,875**	0,88	0,53
			100	0,92	0,88	92	528	72	8	**0,900**	0,92	0,56
			200	0,96	0,83	96	498	102	4	**0,895**	0,96	0,48
		400	50	0,84	0,88	84	528	72	16	**0,860**	0,84	0,54
			100	0,86	0,90	86	540	60	14	**0,880**	0,86	0,59
			200	0,90	0,89	90	534	66	10	**0,895**	0,90	0,58
8	2000	100	50	1,00	0,87	100	522	78	0	**0,935**	1,00	0,56
			100	0,92	0,86	92	516	84	8	**0,890**	0,92	0,52
			200	1,00	0,83	100	498	102	0	**0,915**	1,00	0,50
		200	50	0,96	0,88	96	528	72	4	**0,920**	0,96	0,57
			100	0,94	0,87	94	522	78	6	**0,905**	0,94	0,55
			200	0,92	0,87	92	522	78	8	**0,895**	0,92	0,54
		400	50	0,94	0,89	94	534	66	6	**0,915**	0,94	0,59
			100	0,90	0,89	90	534	66	10	**0,895**	0,90	0,58
			200	0,92	0,93	92	558	42	8	**0,925**	0,92	0,69

ANEXO B. TABLAS DE EXPERIMENTOS POR CLASIFICADOR

En esta sección se muestran las tablas de los experimentos más importantes, donde se utilizó el set de Testing. En esta sección se muestran sólo los parámetros de la tabla 4.19 de la sección 4.13.

Tabla para Razors

TABLE B.1. Tabla de evaluación para el clasificador de Razors

σ	**Tree**	**SIFT**	**LBP**	**TPR**	**TNR**	**TP**	**TN**	**FP**	**FN**	**Rendimiento**	**Recall**	**Precision**
2	1000	200	50	0,75	1,00	75	600	0	25	**0,875**	0,75	1,00
			100	0,81	1,00	81	600	0	19	**0,905**	0,81	1,00
			200	0,72	1,00	72	600	0	28	**0,860**	0,72	1,00
		400	50	0,78	1,00	78	600	0	22	**0,890**	0,78	1,00
			100	0,83	1,00	83	600	0	17	**0,915**	0,83	1,00
			200	0,83	1,00	83	600	0	17	**0,915**	0,83	1,00
2	2000	200	50	0,72	1,00	72	600	0	28	**0,860**	0,72	1,00
			100	0,74	1,00	74	600	0	26	**0,870**	0,74	1,00
			200	0,73	1,00	73	600	0	27	**0,865**	0,73	1,00
		400	50	0,84	1,00	84	600	0	16	**0,920**	0,84	1,00
			100	0,81	1,00	81	600	0	19	**0,905**	0,81	1,00
			200	0,81	1,00	81	600	0	19	**0,905**	0,81	1,00
2	4000	200	50	0,68	1,00	68	600	0	32	**0,840**	0,68	1,00
			100	0,71	1,00	71	600	0	29	**0,855**	0,71	1,00
			200	0,78	1,00	78	600	0	22	**0,890**	0,78	1,00
		400	50	0,84	1,00	84	600	0	16	**0,920**	0,84	1,00
			100	0,75	1,00	75	600	0	25	**0,875**	0,75	1,00
			200	0,82	1,00	82	600	0	18	**0,910**	0,82	1,00

Tabla para Shurikens

TABLE B.2. Tabla de evaluación para el clasificador de Shurikens

σ	**Tree**	**SIFT**	**LBP**	**TPR**	**TNR**	**TP**	**TN**	**FP**	**FN**	**Rendimiento**	**Recall**	**Precision**
6	1000	200	50	0,97	1,00	97	600	0	3	**0,985**	0,97	1,00
			100	0,97	1,00	97	600	0	3	**0,985**	0,97	1,00
			200	0,94	1,00	94	600	0	6	**0,970**	0,94	1,00
		400	50	0,84	1,00	84	600	0	16	**0,920**	0,84	1,00
			100	0,95	1,00	95	600	0	5	**0,975**	0,95	1,00
			200	0,83	0,99	83	594	6	17	**0,910**	0,83	0,93
6	2000	200	50	0,87	1,00	87	600	0	13	**0,935**	0,87	1,00
			100	0,96	1,00	96	600	0	4	**0,980**	0,96	1,00
			200	0,92	1,00	92	600	0	8	**0,960**	0,92	1,00
		400	50	0,83	1,00	83	600	0	17	**0,915**	0,83	1,00
			100	0,79	1,00	79	600	0	21	**0,895**	0,79	1,00
			200	0,92	0,99	92	594	6	8	**0,955**	0,92	0,94
6	4000	200	50	0,97	1,00	97	600	0	3	**0,985**	0,97	1,00
			100	0,93	1,00	93	600	0	7	**0,965**	0,93	1,00
			200	0,94	1,00	94	600	0	6	**0,970**	0,94	1,00
		400	50	0,99	1,00	99	600	0	1	**0,995**	0,99	1,00
			100	0,93	1,00	93	600	0	7	**0,965**	0,93	1,00
			200	0,90	1,00	90	600	0	10	**0,950**	0,90	1,00

Tabla para Guns

TABLE B.3. Tabla de rendimiento para el clasificador de Guns

σ	Tree	SIFT	LBP	TPR	TNR	TP	TN	FP	FN	Rendimiento	Recall	Precision
8	1000	100	50	0,99	0,74	99	444	156	1	**0,865**	0,99	0,39
			100	0,97	0,82	97	492	108	3	**0,895**	0,97	0,47
			200	0,99	0,84	99	504	96	1	**0,915**	0,99	0,51
		200	50	0,99	0,83	99	498	102	1	**0,910**	0,99	0,49
			100	0,99	0,79	99	474	126	1	**0,890**	0,99	0,44
			200	0,96	0,84	96	504	96	4	**0,900**	0,96	0,50
8	2000	100	50	0,98	0,82	98	492	108	2	**0,900**	0,98	0,48
			100	1,00	0,84	100	504	96	0	**0,920**	1,00	0,51
			200	0,98	0,85	98	510	90	2	**0,915**	0,98	0,52
		200	50	0,94	0,83	94	498	102	6	**0,885**	0,94	0,48
			100	0,96	0,80	96	480	120	4	**0,880**	0,96	0,44
			200	0,96	0,85	96	510	90	4	**0,905**	0,96	0,52
8	4000	100	50	0,98	0,83	98	498	102	2	**0,905**	0,98	0,49
			100	0,99	0,79	99	474	126	1	**0,890**	0,99	0,44
			200	0,98	0,82	98	492	108	2	**0,900**	0,98	0,48
		200	50	0,96	0,85	96	510	90	4	**0,905**	0,96	0,52
			100	0,99	0,84	99	504	96	1	**0,915**	0,99	0,51
			200	0,96	0,87	96	522	78	4	**0,915**	0,96	0,55