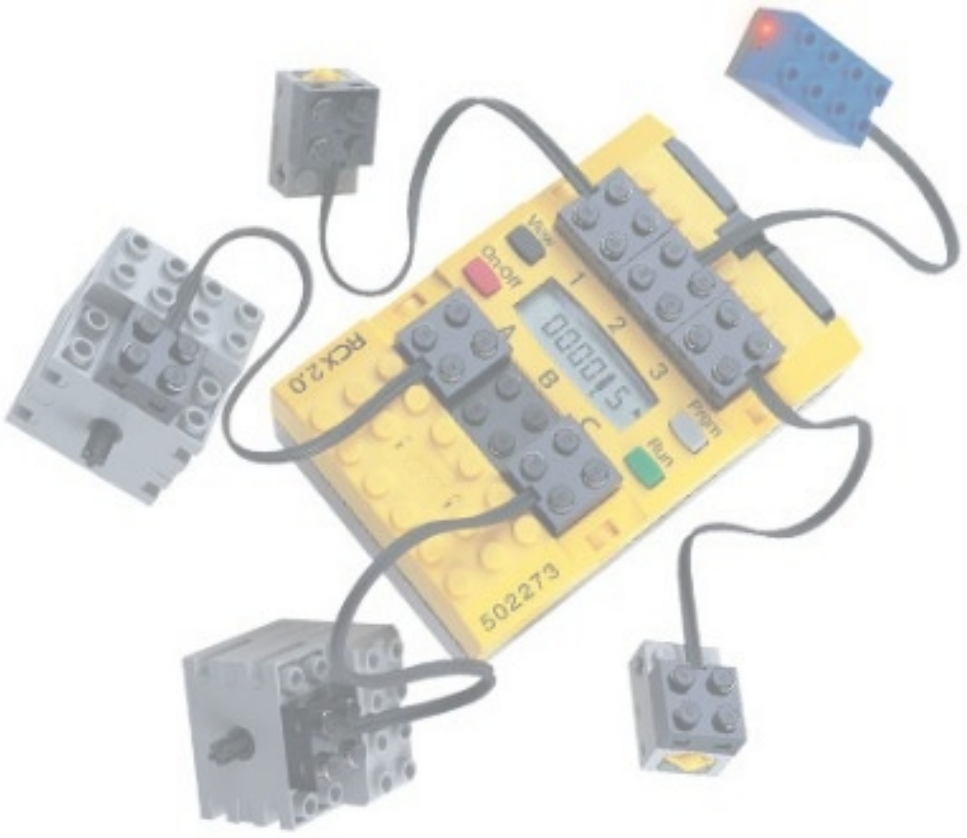
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Trabajo Fin de Grado:

Prototipo de Herramienta para el Inventariado Automático de Piezas de Lego basado en Visión Artificial



Diciembre de 2014
Las Palmas de Gran Canaria

Alumno:
Alberto Jesús Rodríguez Garrido

Tutores:
Dr. Francisco Alexis Quesada Arencibia. Escuela de Ingeniería Informática.
Dr. José Carlos Rodríguez Rodríguez. Teleformación

Resumen

Se presenta un prototipo de herramienta para la detección, segmentación, clasificación y conteo de piezas Lego basado en la librería de visión artificial OpenCV. Este prototipo surge ante la necesidad de automatizar la compleja y tediosa tarea del inventariado de kits Lego de la serie MindStorm.

En el proceso de detección y segmentación se han utilizado técnicas de umbralizado y el algoritmo de segmentación Watershed. Para el proceso de clasificación y conteo se han empleado dos aproximaciones diferentes en la obtención del vector de características de la imagen: BOW y Naive; así como máquinas de vector soporte (SVM) para la clasificación.

Entre otras aportaciones, dicho prototipo permite:

- guardar y recuperar los datos de sesión del clasificador (datos de configuración y entrenamiento),
- usar descriptores de imágenes no soportados aún por OpenCV, p. ej. KAZE, y
- utilizar una técnica de clusterización binaria optimizada por el autor que posibilita operar con descriptores de imágenes binarios.

Las pruebas demuestran que el prototipo es capaz de alcanzar tasas de acierto (piezas correctamente identificadas) de hasta el 98 %.

Abstract

A prototype tool based on computer vision library OpenCV for detection, segmentation, classification and counting of Lego pieces is presented. This prototype arises from the need to automate the complex and tedious task that is the inventory of Lego Mindstorm kits series. For the detection and segmentation process, thresholding techniques and the segmentation algorithm Watershed have been used. For the classification and counting process two different approaches have been used in order to get the image features vector: BOW and Naive; we have also used support vector machines (SVM) for the classification process.

Among other contributions, the prototype allows:

- storing and retrieving classifier session data (configuration and training data),
- using image descriptors not supported by OpenCV, eg . KAZE, and
- using a binary clustering technique optimized by the author that enable to operate with binary descriptors images.

Tests show that the prototype is able to achieve accuracy rates (correctly identified pieces) until 98%.

Índice

Índice de Tablas

Índice de figuras

1 Introducción

Si le preguntase cuántos objetos hay en esta imagen:

Figura 1 Número de objetos en la imagen

O cuántos objetos por categoría hay en esta otra:

Figura 2 Número de objetos por categoría

Y si además realizaran esta tarea de clasificación y conteo en kits Lego de laserie MindStorm, en el que el conjunto de categorías y piezas sea mayor:

Figura 3 Kit Lego serie MindStorm

Seguramente le llevaría un tiempo y un grado de esfuerzo y concentración alto, ya que, por ejemplo, un despiste o distracción, podría hacer que perdiera la cuenta y tuviera que comenzar de nuevo.

Si esto lo tuviera que realizar diariamente, entendería la necesidad de automatizar si no todo, al menos parte del proceso, el cual podemos dividir en los siguientes subprocesos: desmontar el kit, comprobar su completitud o auditarlo(clasificar y contar las piezas que lo integran) y posteriormente colocar en su bandeja cada categoría o grupo de piezas en el espacio reservado.

Es aquí donde surge este trabajo fin de grado (TFG), con la vocación, al tratarse de un prototipo, de sentar las bases de la futura herramienta auditora,ya que en términos de consumo de recursos (tiempo y capital humano) es el subproceso más crítico y susceptible de ser automatizado.

Para ello, se enfocó la solución al problema desde el punto de vista de la Visión Artificial o Visión por Computador. Una disciplina experimental,que en el marco de la ingenieríatiene por objeto ayudar a construir sistemas autónomos, que puedan llevar a cabo alguna de las tareas que el sistema visual humano es capaz de realizar, superándolo inclusos en ocasiones.

Por tanto, nos apoyaremos en el uso de diversas técnicas, procedimientos y algoritmos de dicha disciplina para poder llevar a cabo la automatización de este proceso de auditoría de los kits Lego de la serie MindStorm, esto es, dado una escena o imagen que contiene los objetos de interés (piezas Lego del kit a auditar), proceder a su detección y segmentación (extracción de los objetos o piezas Lego de la imagen), clasificación y conteo.

2 Estructura del documento

Se ha procurado seguir un flujo de desarrollo lineal y conforme al Manual Operativo de Trabajos de fin de Título aprobado en la Junta de Centro de la Escuela de Ingeniería Informática el 14 de Julio de 2014. A continuación indicamos los apartados en los que se ha estructurado:

- **Estado actual**: haremos una breve exposición por analogía de soluciones de visión artificial relacionadas con nuestro problema.
- **Objetivos**: lo que pretendemos lograr con la realización de este TFG.
- Aportaciones: dada la naturaleza de este TFG indicaremos sus contribuciones al ámbito científico-técnico.
- **Competencias**: generales como específicas cubiertas en este TFG.
- Normativa y legislación: analizaremos la normativa y legislación aplicable a este TFG, tanto desde un punto de vista general como específico.
- **Análisis del Problema**: una vez identificado el problema, analizaremos los distintos enfoques o aproximaciones con los que contamos para solucionarlo.
- **Recursos disponibles**: se detalla los recursos claves con los que hemos contado para poder ofrecer una solución a nuestro problema.
- **Diseño de la Solución**: a partir del análisis del problema propondremos el diseño del sistema de visión artificial que lo resolverá.
- **Implementación de la Solución**: refleja el detalle de implementación de la parte *software* de nuestro sistema de visión artificial.
- **Pruebas**: destinadas a comprobar nuestro sistema de visión artificial, así como a obtener la configuración que mejor se adapta a nuestro problema.
- **Resultados**: se indicaran los resultados de las pruebas desde distintos enfoques o perspectivas.
- **Conclusiones**: derivadas tanto de los resultados de las pruebas como de la realización de este TFG.
- **Trabajo futuro**: mejoras propuestas de cara a obtener la futura herramienta de auditoría de los kits Lego.
- **Fuentes de información**: fuentes y referencias consultadas en la realización de este TFG.
- **Anexo AManual de usuario**: manual en el que se explica la instalación y funcionamiento de la librería desarrollada.

Así mismo, el CD en el que se incluye este documento presenta la siguiente estructura de directorio:

- Memoria: contiene este documento, así como su abstract y resumen.
- Inventariador: copia del proyecto realizado en Visual Studio 2013.
- Librerías: copia de las librerías de los descriptores de imágenes KAZE y AKAZE generadas. No se incluyen por motivos de espacio las otras dos librerías que utiliza nuestra solución: OpenCV y Boost.

- Conjunto de Imágenes de entrenamiento: conjunto de imágenes empleadas para llevar a caboel entrenamiento en función del modo de clasificación: BOW-SVM (subdirectorio homónimo) y Naive-SVM (subdirectorio homónimo).
- Conjunto de imágenes deprueba: conjunto de imágenes utilizadas para testear los módulos Detección y segmentación (subdirectorio homónimo) y Clasificación y conteo (subdirectorio homónimo).
- Resultados: contiene el resultado de la prueba efectuada a los módulos Detección y Segmentación (subdirectorio homónimo) y Clasificación y conteo (subdirectorio homónimo).
- Datos de sesión del clasificador: se incluyen copia de los datos de sesión de las configuraciones de prueba del clasificador.

3 Estado actual

Si bien como tal no encontramos ninguna herramienta específica que realizara el proceso de detección, clasificación y conteo de piezas Lego. Sin embargo, se encontró un contador de objetos en contacto desarrollado por la empresa Apliquality Engineering para el sector metalúrgico. Según la información publicada[1]*el programa permite aprender las referencias deseadas y controlar si en el contaje de una determinada referencia hay contaminación. Esta aplicación es ideal para el embolsado de tuercas, tornillos, arandelas, pernos y otros. Al igual que para el reconocimiento de objetos en contacto. Para ello se han utilizado sistemas de visión artificial de INFAIMON, y combina métodos de reconocimiento por gradientes y superficie desarrollados por APLIQUALITY.*

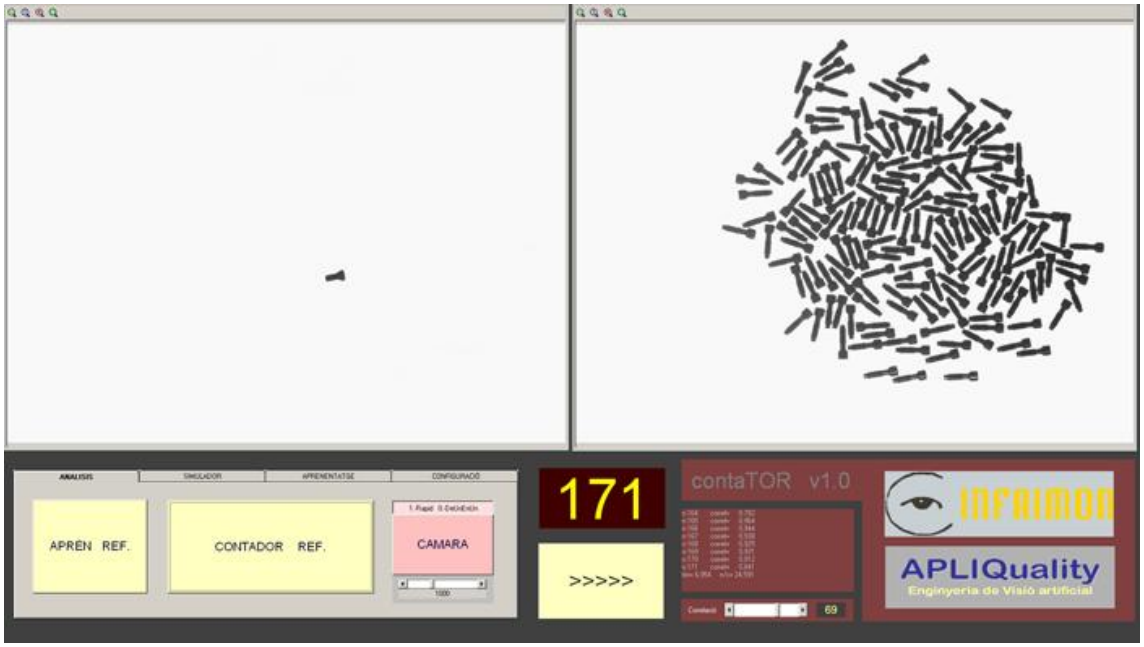


Figura 4 Interfaz del contador de objetos en contacto desarrollado por Apliquality Engineering

Con respecto a las librerías de visión artificial encontradas tenemos entre otras:

- **MATLAB**[2]: librería comercial desarrollada por MathWorks que posee su propio lenguaje de programación de alto nivel. Precisamente esta característica si bien facilita la portabilidad a otros sistemas, produce una merma en el rendimiento y aumento en el consumo de recursos. En cuanto a algoritmos de clasificación contamos con máquinas de vector soporte (SVM), redes neuronales, clasificadores bayesanos ingenuos… En cuanto a algoritmos de segmentación Watershed, clusterización k-means…
- **HALCON**[3]: librería comercial con IDE propio desarrollada por MVTech. Compatible con los lenguajes de programación C, C++ y .Net. Orientada al ámbito industrial, dispone de técnicas de análisis de blob (objetos extraídos de una imagen). En cuanto a algoritmos de clasificación disponemos de máquinas de vector soporte (SVM), redes neuronales, *Gaussian Model Mixtures*(GMM)…
- **EasyObject**[4]: librería comercial que forma parte de la suite Open eVision. Desarrollada por Euresys ofrece capacidad de integración con los lenguajes de programación C++ y .Net, así como con aplicaciones ActiveX. Dispone de funciones de codificación de imagen, de extracción de características de los objetos segmentados, así como de selección y ordenación de estos en función del valor de dichas características.

No obstante, no presentan las ventajas de la librería utilizada en este TFG (OpenCV) para realizar la implementación de nuestro sistema de visión artificial:

- ✓ ser de código abierto y libre,
- ✓ marco de referencia en su área, y
- ✓ contar con una importante comunidad de desarrolladores, lo que le permite estar a la vanguardia, así como disponer de un amplio conjunto de herramientas.

4 Objetivos

Como se comentó en el apartado 1. Introducción, el objetivo principal de este TFG es el diseño y construcción de un prototipo de herramienta que permita la automatización del proceso de auditoría de los kits de la serie Lego MindStorm.

Ello implica despiezar el robot construido, realizar la auditoría en sí, es decir, identificar, clasificar y contar dichas piezas y a continuación guardarlas en su kit.

Si bien, la automatización total del proceso no es posible, al no disponer de un sistema automático de manipulación de piezas, esto no debería verse como un impedimento para la consecución de nuestro objetivo, ya que el problema más significativo no estriba en la parte mecánica o manual del proceso, es decir, en el despiece y posterior colocación de las piezas en el kit, sino en el proceso propiamente de auditoría (identificación, clasificación y conteo).

Para ello y haciendo un símil, se plantearon distintos hitos que nuestro prototipo de herramienta auditora debe satisfacer, en creciente grado de dificultad y bajo ciertas premisas o reglas:

- las piezas Lego se encuentran en un entorno controlado,
- y no puede haber contacto ni oclusión entre ellas.

Así inicialmente, tiene que ser capaz de detectar y contar las piezas Lego presentes en una imagen. De esta forma, la dotamos de las capacidades de detección y segmentación.

Posteriormente, tiene que ser capaz de identificar cuántos tipos o clases diferentes de piezas Lego hay en una imagen. Por tanto, la dotamos de la capacidad de clasificación.

Finalmente y por combinación de las capacidades adquiridas anteriormente (detección, segmentación y clasificación), es capaz de realizar el proceso de auditoría completo, es decir, dada una imagen o escena conteniendo las piezas Lego del kit a auditar, indicar el número de piezas de cada categoría o clase del kit.

5 Competencias

G1. Poseer y comprender conocimientos en un área de estudio (Ingeniería Informática) que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.

Los conocimientos plasmados en la elaboración y redacción de esta memoria.

G2. Aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

Véase apartados 8. Análisis del problema, 9. Diseño de la solución y 11. Implementación de la solución.

G3. Reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.

Véase los apartados 13. Pruebas y 14. Resultados.

G4. Transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.

Mediante el empleode un lenguaje sencillo y claro, o el uso de analogías en la redacción de esta memoria. A modo de ejemplo véase el apartado 8. Análisis del problema.

G5. Desarrollar aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

Al tener que realizar un trabajo de investigación previo para poder dar solución al problema de visión artificial planteado en este TFG.

N1. Comunicarse de forma adecuada y respetuosa con diferentes audiencias (clientes, colaboradores, promotores, agentes sociales, etc.), utilizando los soportes y vías de comunicación más apropiados (especialmente las nuevas tecnologías de la información y la comunicación) de modo que pueda llegar a comprender los intereses, necesidades y preocupaciones de las personas y organizaciones, así como expresar claramente el sentido de la misión que tiene encomendada y la forma en que puede contribuir, con sus competencias y conocimientos profesionales, a la satisfacción de esos intereses, necesidades y preocupaciones.

Al mantener por ejemplo intercambio de correos con desarrolladores, comunicaciones con los tutores, conversaciones telefónicas con el soporte técnico de la empresa INFAIMON para resolver dudas o aclarar cuestiones técnicas relacionadas con la adquisición de imágenes.

N2. Cooperar con otras personas y organizaciones en la realización eficaz de funciones y tareas propias de su perfil profesional, desarrollando una actitud reflexiva sobre sus propias competencias y conocimientos profesionales y una actitud comprensiva y empática hacia las competencias y conocimientos de otros profesionales.

Colaboración con otros profesionales de la visión artificial, concretamente con Pablo Alcantarilla autor de los descriptores de imágenes KAZE y AKAZE (véase apartado 6. Aportaciones).

N3. Contribuir a la mejora continua de su profesión así como de las organizaciones en las que desarrolla sus prácticas a través de la participación activa en procesos de investigación, desarrollo e innovación.

Véase apartado 6. Aportaciones.

N4. Comprometerse activamente en el desarrollo de prácticas profesionales respetuosas con los derechos humanos así como con las normas éticas propias de su ámbito profesional para generar confianza en los beneficiarios de su profesión y obtener la legitimidad y la autoridad que la sociedad le reconoce.

Al emplear técnicas o prácticas orientadas a obtener un *software* de calidad (véase 10. Metodología de desarrollo) y por tanto un alto grado de satisfacción por parte del cliente, así como el respeto de la normativa y legislación vigente (véase subapartado 7.3 Licenciamiento *software* aplicable a este TFG).

N5. Participar activamente en la integración multicultural que favorezca el pleno desarrollo humano, la convivencia y la justicia social.

Como es el hecho de utilizar y producir *software*de código abierto(véase subapartado 7.3 Licenciamiento *software* aplicable a este TFG) y en inglés.

T1. Capacidad para concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada, la

concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas. (G1, G2)

Al concebir y realizar este TFG.

T2. Capacidad para dirigir las actividades objeto de los proyectos del ámbito de la informática, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada. (G1, G2)

Como lo demuestra la realización de este TFG.

T5. Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada. (G1, G2)

Véase los apartados 10. Metodología de desarrollo, 11. Diseño de la solución y 12. Implementación de la solución.

T6. Capacidad para concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada. (G1, G2)

Véase el subapartado 11.1 Módulo Adquisición de imágenes.

T7. Capacidad para conocer, comprender y aplicar la legislación necesaria durante el desarrollo de la profesión de Ingeniero Técnico en Informática y manejar especificaciones, reglamentos y normas de obligado cumplimiento. (N4)

Véase apartado 7. Normativa y Legislación.

T8. Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.

Véase los apartados 8. Análisis del problema, 11. Diseño de la solución y 12. Implementación de la solución.

CII01. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

Véase los apartados 11. Diseño de la solución, 12. Implementación de la solución y 7. Normativa y Legislación.

CII18. Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

Véase apartado 7. Normativa y Legislación.

TFG01. Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

La realización de este TFG.

IS01. Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

Véase apartados Véase los apartados 10. Metodología de desarrollo, 11. Diseño de la solución y 12. Implementación de la solución.

IS02. Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.

Véase apartados 4. Objetivos, 8. Análisis del problema y 11. Diseño de la solución.

IS03. Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.

Véase apartados 8. Análisis del problema y 11. Diseño de la solución.

IS04. Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

Véase apartados 8. Análisis del problema, 10. Metodología de desarrollo, 11. Diseño de la solución y 12. Implementación de la solución.

IS05. Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.

Véase subapartado 11.1 Módulo Adquisición de imágenes.

IS06. Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos.

Al desarrollar la solución software conforme a los principios y prácticas descritos en el apartado Metodología de desarrollo, así como aplicando un tipo de licenciamiento libre (véase subapartado 7.3 Licenciamiento software aplicable a este TFG).

IC07. Capacidad para analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos.

Véase subapartado 11.1 Módulo Adquisición de imágenes.

CP05. Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.

Véase los apartados 8. Análisis del problema y 11. Diseño de la solución.

CP07. Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

Véase los apartados 8. Análisis del problema, 11. Diseño de la solución y 12. Implementación de la solución.

6 Aportaciones

La principal aportación radica en haber desarrollado un prototipo de herramienta que permite identificar, clasificar y contar piezas Lego de la serie MindStorm.

Como consecuencia de lo anterior, hemos podido ofrecer a la comunidad una librería de detección, segmentación, clasificación y conteo basada en OpenCV que:

- Permite la normalización de las imágenes segmentadas.
- Adapta el modelo BOW de esta para permitir el uso de descriptores binarios.
- Admite el uso de descriptores de imagen no soportados por esta siempre que se respete su interfaz.
- Provee de un mecanismo de clusterización binaria por medio de la técnica *k-medoids*.
- Permite guardar y recuperar los datos de sesión del clasificador (datos de configuración y entrenamiento).
- Dispone de un mecanismo para la generación automática de los datos de entrenamiento sin requerir, a diferencia de otras soluciones, de un gran esfuerzo por parte del usuario.

Así como contribuir al:

- Desarrollo de OpenCV. Parte de las funcionalidades de la librería desarrollada en este TFG participarán en el OpenCV Vision Challenge, que tiene como objetivo la actualización y extensión de OpenCV.
- Mantenimiento y difusión de los descriptores de imagen KAZE y AKAZE. Se ha mantenido contacto con el autor y comunicado y resueltos algunos fallos. Así mismo, los resultados de las pruebas ayudarán a los fines ya descritos.

7 Normativa y Legislación

Si bien debido a la naturaleza de este proyecto no es aplicable la Ley Orgánica de Protección de Datos (Ley Orgánica 15/1999), ya que no tratamos o procesamos datos de carácter personal, sí es aplicable todo lo relacionado en materia de propiedad intelectual al hacer uso de *software* de terceros.

Por tanto, en este apartado nos centraremos en ofrecer una visión general sobre la propiedad intelectual y su aplicación tanto al desarrollo de *software* (Licenciamiento *software*)como a este TFG (Licenciamiento *software* aplicable a este TFG).

7.1 Propiedad intelectual

Haciendo un poco de historia, los derechos de autor han regulado durante siglos los derechos morales y patrimoniales de la creación de obras literarias, artísticas y científicas. Se distingue dos tradiciones: la anglosajona, que reclama los derechos sobre la propiedad creativa comercializable (*Copyright*), plasmándose jurídicamente por primera vez en el Estatuto de la Reina Ana de 1710, otorgando al autor la explotación de sus obras publicadas por un período de 14 años, prorrogables mientras viviese; y por otra parte la tradición de la Europa continental, que desarrollará una idea moral de identidad entre autor y obra que dará como resultado la primera ley de derechos de autor tras la Revolución Francesa.

Estas dos ideas componen un híbrido conceptual en la actual definición de Propiedad Intelectual, definida por la Organización Mundial de la Propiedad Intelectual (OMPI)[a]como:

> "La propiedad intelectual tiene que ver con las creaciones de la mente: las invenciones, las obras literarias y artísticas, los símbolos, los nombres, las imágenes y los dibujos y modelos utilizados en el comercio."

Dentro de la propiedad intelectual se distinguen:

- Propiedad Industrial (inventos y patentes, marcas, diseños y nombres).
- Derechos de Autor (literatura, pintura, fotografía, escultura, diseño arquitectónico, *software*, vegetales genéticamente modificados).
- Derechos Conexos (intérpretes, productores y organismos de radiodifusión).

En España la Ley de Propiedad Intelectual (Real Decreto Legislativo 1/1996, de 12 de abril) entiende el derecho de autor como algo único, con potestad de reproducción, comunicación, distribución o transformación. En su artículo 31 permite copia privada siempre que no exista ánimo de lucro y el duplicado se realice a través de una "copia legal". Para compensar a los autores, introduce el pago de un canon compensatorio asociado a algunos soportes de grabación. Así mismo, y al contrario que el sistema anglosajón, hace una mayor defensa de los derechos morales[b]frente a los derechos de carácter patrimonial, en los cuales hay que

[a] Organización fundada en el seno de las Naciones Unidas en 1967.

[b] Derechos más relacionados con el reconocimiento de la condición de autor en las obras, así como a exigir el respeto a la integridad de las obras y la no alteración de las mismas.

distinguir entre los derechos relacionados con la explotación de la obra (derechos exclusivos y de remuneración) y los derechos compensatorios (p. ej. derecho por copia privada).

De este modo, y en el marco que nos ataña, el de *software* o programa de computación, definido por la OMPI como:

> "un conjunto de instrucciones que, cuando se incorpora en un soporte legible por máquina, puede hacer que una máquina con capacidad para el tratamiento de la información indique, realice o consiga una función, tarea o resultados determinados"

Identificamos los derechos morales con los que permiten reivindicar al autor (programador, equipo de desarrollo,...) la paternidad de su obra, oponerse a su deformación, mantenerla inédita, autorizar a terceros para terminar la obra inconclusa y exigir que se mantenga anónima o seudónima, si es su deseo.

Por otra parte, identificamos los derechos patrimoniales con los que permiten al autor o titular de los derechos (desarrollador, empresa de desarrollo, cliente que encarga el desarrollo,…) explotar comercialmente la obra, es decir, los derechos de reproducción, transformación, distribución o puesta a disposición y publicación de la obra.

Por tanto, será en el contrato de licencia del software donde el titular de los derechos indique al usuario que derechos autoriza ejercer.

7.2 Licenciamiento *software*

Como se mencionó en el subapartado anterior, a través del contrato de licencia *software* el titular de los derechos morales y patrimoniales especifica que derechos autoriza a ejercer al usuario.

A nivel general y en base al tipo de licenciamiento *software* podemos distinguir:

- ***Software* de dominio público**: carece de licencia y por tanto puede ser usado, copiado, modificado y distribuido con o sin ánimo de lucro. Algunos tipos de copia o versiones modificadas pueden no ser libres si el autor impone restricciones adicionales en la redistribución del original o de trabajos derivados.
- ***Software* de código cerrado o también llamado *software* propietario o privativo**: por lo general, no permiten que el *software* sea modificado, desensamblado, copiado o distribuido de formas no especificadas en la propia licencia, regula el número de copias que pueden ser instaladas e incluso los fines concretos para los cuales puede ser utilizado. La mayoría de estas licencias limitan fuertemente la responsabilidad derivada de fallos en el programa. Ejemplos: Contrato de Licencia para el Usuario Final (CLUF) o Acuerdo de Licencia del Usuario Final (EULA).
- ***Software* de código abierto**: el objetivo es permitir el acceso al código ya sea por motivos de índole filosóficos (*Free Software Foundation*) o bien de índole técnico (*Open Source Iniciative*). Todas ellas se basan en la práctica del*copyleft*. Esta surge en oposición al *copyright* yconsiste en el ejercicio del derecho de autor con el objetivo de permitir la libre distribución de copias y versiones modificadas de una obra u otro

trabajo, exigiendo que los mismos derechos sean preservados en las versiones modificadas (fuerte) o no (débil o permisivo). Ejemplos: *GNU General Public License*, *Berkeley Software Distribution* (BSD)…

Finalmente, si bien la mayoría de licencias de código abierto ni cuentan con una traducción al castellano, niestán adaptadas al marco jurídico español, existen alternativas como las *Creative Common*. Estas son un conjunto de seis licencias obtenidas por combinación de las siguientes condiciones[5]:

- *Reconocimiento (Attribution): En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.*
- *No Comercial (Non commercial): La explotación de la obra queda limitada a usos no comerciales.*
- *Sin obras derivadas (No Derivate Works): La autorización para explotar la obra no incluye la transformación para crear una obra derivada.*
- *Compartir Igual (Share alike): La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.*

7.3 Licenciamiento *software* aplicable a este TFG

Se optó por aplicar un licenciamiento *software* BSD modificado (de 3 cláusulas).

Las razones que justifican la elección:

- Compatible con las librerías y algoritmos utilizados en este TFG:
  - Librería OpenCV y algoritmos KAZE y AKAZE utilizan este tipo de licencia
  - Librería Boost posee una licencia propia compatible
- Permitir un mayor grado de difusión y uso. Al ser una licencia con *copyleft* permisivo, admite que se pueda combinar con *software* que emplea otro tipo de licencias (libres o propietarias), así como aplicar al *software* derivado cualquier otro tipo de licencia.

Así mismo, no se permite sin autorización previa el uso con fines comerciales del *software* producido o derivado de este TFG.

8 Análisis del Problema

Tal y como mencionábamos en los apartados 1. Introducción y 4. Objetivos, nos enfrentamos al problema de poder automatizar el proceso de auditoría de los kits de la serie Lego MindStorm.

Es un problema perteneciente al campo de Visión Artificial, en el que a partir de una escena o imagen conteniendo los objetos de interés, en nuestro caso las piezas Lego del kit, debemos ser capaces de detectarlas, clasificarlas y contarlas.

Para ello, comparándolo con nuestro proceso de visión, tendremos que contar con algún tipo de dispositivo, que al igual que el ojo humano, nos permita capturar las imágenes (piezas Lego del kit) que queremos clasificar y contar.

Posteriormente y siguiendo con dicha comparación, estas imágenes capturadas por nuestros ojosson enviadas a nuestro cerebro, concretamente a nuestra corteza visual, que es el encargado de realizar el proceso de percepción visual.

Este proceso de percepción visualaplicado a nuestro problema, consistirá en detectar y extraer cada uno de los objetos (piezas Lego) de la imagen y realizar su clasificación y conteo.

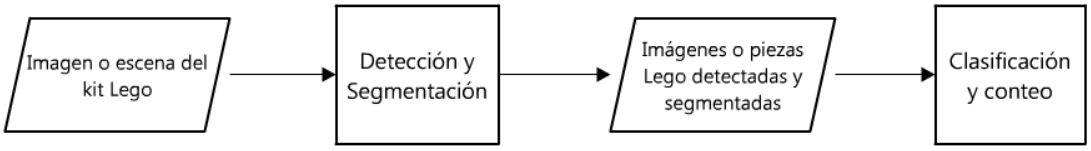


Figura 5 Proceso de percepción visual

A continuación,analizaremos dentro del contexto de la visión artificial lasdiferentes etapas en las que se divide nuestro problema: adquisición de imágenes, detección y segmentación (extracción de los objetos de interés en una escena) y clasificación y conteo.

8.1 Adquisición de imágenes

Como indicábamos al comienzo de este apartado, nuestro sistema de visión artificial necesita contar con “ojos” que le permita capturar la información del exterior (imágenes de las piezas Lego).

Si bien suponemos que las piezas Lego serán colocadas en un espacio habilitado para este propósito (véase Laboratorio de Visión), será necesario contar con algún dispositivo que permita realizar dichas capturas. Ejemplos de dispositivos ópticos: cámara fotográfica, videocámara, cámara Ethernet, webcam…

A este respecto, el dispositivo óptico deberá reunir unas determinadas características con el fin de garantizar la calidad de dichas capturas. En otras palabras y siguiendo con la comparación ya iniciada al comienzo de este apartado, tenemos que asegurarnos de contar con los mejores “ojos”, evitando problemas de astigmatismo (distorsión, baja resolución), o de hipermetropía o miopía (distancia focal no apropiada para nuestra área de trabajo).

Por tanto, a la hora de decantarnos por una opción u otra tendremos que valorar los siguientes aspectos:

- Espacio de color que vamos a requerir: grises o de color.
- Resolución (nivel de detalle o número de píxeles que es capaz de obtener el sensor)
- Tipo de sensor: si bien lo más habitual es encontrar sensores CMOS debido almenor consumo de energía, ser más económico respecto a los de tipo CCD y no presentar efecto *blooming*, fenómeno por el cual cuando un pixel ha alcanzado su nivel de

saturación empieza a “contagiar” a sus vecinos creando efectos indeseados, los sensores CCD a diferencia de los CMOSpresentan una mayor calidad en situaciones de poca luminosidad. Apareciendo recientemente como alternativa a estos problemas de iluminación el BSI-CMOS (*backside ilumination CMOS*), que logra incrementar la sensibilidad de capturar un fotón de entrada del 60% en un sensor CMOS convencional a más del 90%.

- Distancia focal: debemos seleccionar una lente conforme a nuestra área de trabajo, es decir, superficie a capturar y ubicación de la lente.
- Capacidad de integración (comunicación con otros sistemas): aislado o comunicado, p. ej. con nuestro sistema de percepción visual mediante algún tipo de interfaz.
- Flujo de datos: bajo petición (estático) o continuo (dinámico) y en este último caso a qué velocidad o tasa de entrega.

8.2 Detección y segmentación

Una vez hemos obtenido por alguno de los dispositivos descritos en el apartado anterior el conjunto de imágenes a procesar, el siguiente paso es poder detectar y segmentar (extraer) cada uno de los objetos de interés (piezas Lego del kit) presentes en dichas imágenes.

Para ello, es necesario realizar previamente un preproceso de la imagen, dependiente del contexto en el que se ha realizado las capturas y que nos permita adecuarla a los diferentes algoritmos o técnicas de segmentación. Operaciones típicas de preprocesado de imagen: morfológicas, de conversión de espacio de color, ecualización de su histograma…

En cuanto a algoritmos o técnicas empleadas en el proceso de segmentación, es decir, la extracción y por tanto detección deuna o más regiones u objetos de interés de la imagen(piezas Lego) basada en un criterio de discontinuidad o similitud, la visión artificial nos ofrece las siguientes aproximaciones:

- **Segmentación basada a nivel de pixel o técnica de umbralizado**.Empleado en los casos en el que el fondo de la imagen (todo aquello que no es objeto de interés) es claramente diferenciable y conocido. Se utiliza un umbral como criterio para determinar qué es objeto y qué no lo es (fondo). El método comúnmente utilizado **para el cálculo de dicho umbral es el de Otsu**[6].
- **Segmentación basada en detección de bordes**. Un ejemplo representativo es el algoritmo de Canny[7], un algoritmo óptimo de detección de bordes. Por óptimo queremos decir buena detección, buena localización (los bordes deben estar lo más cerca posible del borde la imagen real) y respuesta mínima (los bordes deben ser marcados una vez y con cierta tolerancia a ruido).
- **Segmentación basada en regiones**.Tiene en cuenta los niveles de gris de los píxeles vecinos, bien incluyendo píxeles de vecinos similares (*growing region*), bien separando la imagen en regiones basada en un criterio de similitud para posteriormente fusionar o unir aquellas regiones con mismo criterio (*“quad tree” division*) o bien mediante la aplicación de Watershed[8], un algoritmo que considera una imagen en escala de grises como un relieve topográfico, donde el nivel de gris de un píxel se interpreta como su altitud en el relieve.

8.3 Clasificación y conteo

Una vez hemos obtenido el conjunto de imágenes de interés (piezas Lego del kit) por medio de la aplicación y/o combinación de los procedimientos o técnicas descritas en los subapartados anteriores, la última fase del proceso es realizar la clasificacióny conteo de los mismos.

Para ello, necesitamos contar con artefactos que nos permitan tanto describir dichas imágenes (descriptores), así como con artefactos que a partir de los anteriores nos permitan clasificarlos (clasificadores). Siendo el proceso de conteo trivial una vez hayamos identificado cada uno de los objetos del conjunto de imágenes.

Por tanto, a continuación veremos las aproximaciones más empleadas en el campo de la visión artificial para los artefactos mencionados, es decir, descriptores de imágenes y clasificadores.

8.3.1 Descriptores de imágenes

Son artefactos matemáticos que nos permiten identificar de forma unívoca una imagen, a pesar de presentar transformaciones debido a cambios de iluminación, ruido, escala, rotación,...

Por tanto una imagen se descompone en pequeñas regiones que son caracterizadas por estos descriptores.

Posteriormente estos descriptores son utilizados en la aplicación para llevar a cabo procesos de comparación, entrenamiento de clasificadores, etc.

Principales problemas:

- A la hora de detectar estas características, ¿en qué regiones de la imagen realizarlas?
- ¿Cómo computar dichas características o descriptores?
- ¿Cómo comparar dos descriptores?

Respecto al primer problema, ¿en qué regiones?,las soluciones pasan por descomponer la imagen como una cuadrícula, en *key-points*, o bien tratarla de forma global.

El segundo problema, ¿cómo computar dichas características?, dependerá de la descomposición elegida, centrándonos por ser la solución más estudiada en la relativa a los *key-points*, esto es, puntos o regiones (de diferente orientación, escala,…) obtenidos a través de la aplicación del método que los describen.

Respecto al tercer problema, ¿cómo comparar dos descriptores?, los descriptores suelen venir con sus propias funciones de distancias que nos indicaría el grado de similitud, empleando la mayoría de ellos la distancia euclídea. No obstante, cabe destacar la existencia de otro tipo de descriptores denominados binarios que emplean distancia Hamming (BRISK[9], ORB[10],…).

Se emplean distintos descriptores en base a las invariancias antes efectos visuales que tenga que soportar la imagen (iluminación, escala, rotación…).

Tipos de descriptores:

- Basados en intensidad (nivel de gris de cada pixel): No capturan información geométrica y no es invariante frente a cambios de iluminación. Ejemplo: Histograma de Intensidad.
- Histograma de gradientes de imagen (dirección y magnitud de cada pixel): Invariante frente a cambios de iluminación. No captura información geométrica. Ejemplo: los descriptores SIFT[11], GLOH[12], SURF[13].
- Basados en la similitud de la forma. Ejemplo: los descriptores Shape Context, Local Self-Similarity y Geometric Blur.
- Detectan el color: En base al espacio de color utilizado tenemos:
    - Espacio de color RGB: por ejemplo los descriptores Histograma de Color, RGB-SIFT y RG-SIFT (emplea solo los canales R y G).
    - Espacio de color HSV: por ejemplo el descriptor HSV-SIFT.
    - Espacio de color Opponent: por ejemplo los descriptores OpponentSift y C-SIFT (emplea O1/O3 y O2/O3 del espacio).

Así mismo, ya que el descriptor o vector de características de la imagen se obtiene a partir de los *key-points*, hay algoritmos que:

- Resuelven únicamentela obtención de *key-points* (detector). P. ej. Dense.
- Obtienen el vector de características a partir de los *key-points* suministrados (extractor). P. ej. FREAK.
- Presentan métodos para calcular tanto sus *key-points* como su vector de características. P. ej. SIFT.

Por último, si bien hasta ahora ha predominado descriptores que para generar dicho vector de características emplean espacios de escala gaussianos, recientemente han aparecido descriptores que emplean espacios de escala no lineales (KAZE[14], AKAZE[15]), los cuales respetan a diferencia de los primeros el contorno natural de los objetos.

8.3.2 Clasificadores

Como ya se mencionó al comienzo de este subapartado, una vez que disponemos de algún mecanismo para describir las imágenes,necesitamos de algún tipo de artefacto que a partir de estos nos permita categorizarlas y a partir de ahí clasificarlas.

Por tanto y siguiendo con la analogía o comparación que mencionábamos al inicio de este apartado, nuestro sistema de percepción estaría formado por clasificadores o algoritmos de aprendizaje que por razonamiento estadístico infieren el resultado, esto es, a que categoría pertenece dicha imagen (pieza Lego del kit).

Conforme al modo de aprendizaje podemos encontrar los siguientes **tipos de clasificadores o algoritmos de aprendizaje:**

- **Supervisado**: es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro,

los resultados deseados. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación). El objetivo del aprendizaje supervisado es el de crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos supervisados, los datos de entrenamiento. Para ello, tiene que generalizar a partir de los datos presentados a las situaciones no vistas previamente. Por tanto los datos de entrenamiento a partir de los cuales se construye el modelo de predicción deben estar etiquetados.Empleantanto modelos generativos, p. ej. *Bag of Words* (BOW)[16], Redes Neuronales Artificiales, Clasificador Bayesano Ingenuo…; como modelosdescriptivos, p. ej. *Support Vector Machine* (SVM)[17],*Linear discriminant analysis* (LDA)…; o bien una combinación de ambos.

- **No supervisado**:El modelo es ajustado a las observaciones y no hay un conocimiento a priori. Los datos de entrada son tratados como un conjunto de variables aleatorias sobre las cuales se construye el modelo de densidad. Por tanto y a diferencia de los algoritmos supervisados, los datos de entrenamiento no están etiquetados. Se emplean normalmente en la compresión de datosy en clustering, p. ej.K-Means[18].
- **Semi-supervisado**: Es una combinación de los anteriores. De esta forma, se reduce el coste asociado al etiquetado, solo se requiere una pequeña cantidad de datos, y se mejora notablemente la exactitud de aprendizaje en comparación con un algoritmo no supervisado. Preserva la capacidad de recibir *feedback* del entorno propia de los algoritmos de aprendizaje supervisado. Ejemplos:*Transductive Support Vector Machine* (TSVM),la técnica de co-entrenamiento (*co-training*)…

9 Recursos

En este apartado detallaremos a continuación los principales recursos *hardware* y *software*con los que hemos contado para realizar este TFG.

9.1 Recursos *hardware*

- **Plataforma Kaiser**: plataforma que permite fijar de una forma precisa el campo visual de un dispositivo óptico. Esto es debido a la posibilidad de regular tanto su altura como el ángulo de la lente con respecto a la superficie de trabajo.

Figura 6 Plataforma Kaiser

- **Estación de trabajo donde se realizaron las capturas de imágenes**: procesador Intel Core i7 2.80 GHZ, 4GB Memoria con sistema operativo Windows 8.1 Enterprise 64 Bits.
- **Estación de trabajo donde se realizaron las pruebas**: procesadorIntel Core i7 920 2.67 GHZ, 12 GB Memoria con sistema operativo Windows 7 Enterprise SP1 64 Bits.
- **Webcam Logitech modelo Orbit**: sensor de tipo CMOS, con una resolución nativa de 2 mega píxeles, pudiendo alcanzar los 8 mega píxeles por interpolación.

Figura 7 Webcam Logitech modelo Orbit

- **Cámara Ethernet a color Mako modelo 419C**: dispone de un sensor tipo CCD, de tamaño 4/3 de pulgada y resolución nativa de 4 mega píxeles.

Figura 8 Cámara Ethernet Mako modelo 419C

- **Ópticas para sensores de ½ pulgada y resolución máxima de 2 mega píxeles**.

Figura 9 Ópticas con montura tipo C

9.2 Recursos *software*

- **Suite ofimática Microsoft Office 2010**[19]: Para la redacción y generación de este documento.
- **Lovely Charts**[20]: Herramienta online de dibujo utilizada para generar los diagramas de flujos de este documento.
- **Microsoft Visual Studio 2013 Ultimate**[21]: IDE empleado para la implementación *software* de nuestra solución.
- **FTP Filezilla 3.7.3**[22]: servidor FTP libre que utilizamos para facilitar la distribución de las capturas de imágenes.
- **Librería OpenCV 2.4.9**[23]: biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas. Es una librería multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, comoreconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estereoscópica y visión robótica. Pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multi núcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para dichos procesadores.
- **LibreríaBoost versión 1.55**[24]: es un conjunto de bibliotecas de *software* libre y revisión por pares preparadas para extender las capacidades del lenguaje de programación C++. Su licencia, de tipo BSD, permite que sea utilizada en cualquier tipo de proyectos, ya sean comerciales o no.

10 Metodología de desarrollo

Dada la naturaleza investigadora o experimental de este TFG no se llegó a seguir estrictamente una metodología de desarrollo específica. No obstante, al estar nuestro proceso de desarrollo

guiado por una serie de hitos o etapas a cubrir (véase apartado 4. Objetivos), se optó por combinarun enfoque iterativo incrementalcon un enfoque ágil (los cambios son bienvenidos, entrega de funcionalidad sobre documentación, implicación del cliente en el proceso de desarrollo).

Así mismo, debido a la necesidad de que nuestra solución sea modular,altamente configurable y abierto a evolución, se utilizaron tanto los principios de diseño de orientación objetos SOLID (acrónimo del inglés *Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion*)[25]:

- **Principio de Única Responsabilidad** (*Single responsibility principle*): un objeto solo debería tener una única responsabilidad.
- **Principio Abierto/Cerrado** (*Open Close Principle*): la noción de que las "entidades de *software*… deben estar abiertas para su extensión, pero cerradas para su modificación".
- **Principio de sustitución de Liskov** (*Liskov substitution*): la noción de que los "objetos de un programa deberían ser reemplazables por instancias de sus subtipos sin alterar el correcto funcionamiento del programa".
- **Principio de Segregación de la Interface** (*Interface segregation principle*): la noción de que "muchas interfaces cliente específicas son mejores que una interfaz de propósito general."
- **Principio de Inversión de Dependencia** (*Dependency inversion principle*): la noción de que uno debería "Depender de Abstracciones. No depender de concreciones." La Inyección de Dependencias es uno de los métodos que siguen este principio.

Como un conjunto de buenas prácticas recogidas por Robert C. Martin en su libro Clean Code:el uso de buenos nombres que reflejen la funcionalidad o el concepto, funciones de tamaño reducido, limitar salvo razón justificada el número de argumentos de un método a no más de 3 parámetros, no repetirse,uso apropiado de los comentarios…

Todo ello con el objetivo de producir un código expresivo, mantenible, extensible y de calidad.

11 Diseño de la solución

Diseñada conforme a lo expuesto en el apartado8. Análisis del Problema, se caracteriza por una estructura modular, en el que nuestros "ojos" están representados por el módulo "Adquisición de Imágenes" y nuestro proceso de percepción visuales llevado a cabo por los módulos "Detección y segmentación" y "Clasificación y conteo", todos ellos independientes pero relacionados entre sí.

De esta forma, dada la naturaleza y complejidad de nuestro problema, nos permite adaptar cada etapa en la que se divide, esto es, de adquisición de imágenes, de detección y segmentación (extracción de cada uno de los objetos o piezas Lego) y posteriormentede clasificación y conteo de las piezas Lego detectadas y extraídas, a los requerimientos y necesidades presentes y futuros.

Por tanto, nuestra propuesta de solución está formada por un módulo de naturaleza más física o *hardware* (módulo“Adquisición de imágenes”) y por dos módulos de una naturaleza claramente *software*que conforman una biblioteca o librería (módulos “Detección y segmentación” y “Clasificación y conteo”).

11.1 Módulo Adquisición de imágenes

Ubicado en el espacio de trabajo que hemos denominado Laboratorio de Visión, es el encargado de realizar las capturas de las imágenes que contienen las piezas Lego de los kit que queremos auditar.

Está integrado por los siguientes elementos:

- Plataforma Kaiser en la cual se colocan las piezas Lego del kit a auditar. Condiciones ideales de trabajo: superficie de 35 cm^2 y distancia focal de 50 cm.
- Dispositivo óptico conectado a un terminal tipo PC en donde se almacenan las capturas. Contando dicho PC con un servidor FTP, Filezilla en nuestro caso, para facilitar el acceso y distribución remoto de las mismas.

Respecto a la elección del dispositivo óptico, inicialmente se optó por la cámara Ethernet de alta resoluciónMako. No obstante, al no contar con ópticas adecuadas para poder trabajar con dicha cámara, fue necesario emplear un dispositivo óptico de peor resolución, la webcam de Logitech(véase subapartado 9.1 Recursos *hardware*). Este hecho y a fin de asegurar el máximo nivel de detalle de la imagen, nos obligó a establecer las siguientes condiciones de trabajo:

- Imágenes en escala de grises (mayor contraste).
- Distancia focal de 43,9 cm.
- Superficie de trabajo de 8,5 cm^2.

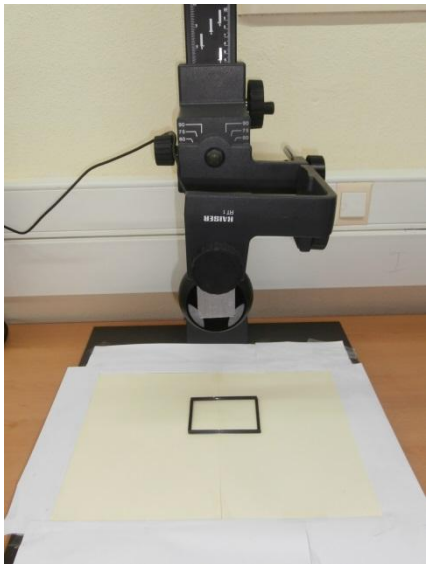


Figura 10 Área de trabajo delimitada por el rectángulo

Por tanto, cumpliendo con lo indicado en el apartado 4. Objetivos, las capturas de las imágenes se realizan en un entorno controlado, en el que procuramos garantizar:

- Las restricciones impuestas de no contacto ni oclusión entre las piezas Lego
- Un alto contraste entre las piezas Lego y el fondo
- Obtención de imágenes de alta calidad y nivel de detalle (iluminación adecuada, reducción de sombras…)

Condiciones claves que influirán directa e indirectamente en los procesos llevados a cabo por los otros dos módulos que veremos a continuación.

11.2 Módulo Detección y segmentación

Como su propio nombre indica, es el encargado de realizar el proceso de detección y segmentación conforme a lo expuesto en el apartado 8. Análisis del problema.

Parámetro de entrada: imagen o escena conteniendo las piezas lego del kit a auditar.

Parámetro de salida:conjunto de objetos o piezas Lego detectados yextraídos (segmentados) en dicha imagen.

Dicho proceso se divide en:

- **Etapa de preproceso de la imagen**: íntimamente ligada al entorno o contexto en el que se realice la adquisición de las imágenes. Es un procedimiento específico, creativo, de ajuste manual y clave, ya que interviene tanto en la obtención de los marcadores que emplea nuestro algoritmo de segmentación, como en la adecuación de nuestra imagen a los requerimientos de este. Dado que nuestro proceso de segmentación se realiza en un entorno controlado (véase subapartado 11.1 Módulo Adquisición de imágenes), se utilizan las siguientes técnicas de preproceso de imagen:
    - conversión a escala de grises (espacio de color requerido por el algoritmo de segmentación elegido)
    - ecualización del histograma de la imagen (mejora el contraste de la imagen)
    - operación morfológica de dilatación (elimina huecos o espacios en el interior de los objetos)
    - umbralizado binario inverso,usando como valor de umbral el obtenido por el método de Otsu.
- **Obtención de los marcadores de nuestro algoritmo de segmentación**
- **Aplicación de nuestro algoritmo de segmentación a la imageno escena**

Como algoritmo de segmentación emplearemos el algoritmo de segmentación basado en regiones Watershed. La elección de este algoritmo se debe a su robustez contrastada y los resultados obtenidos en nuestras pruebas preliminares.

Por último, si bien el proceso de segmentación concluye con la obtención del conjunto de imágenes detectadas y extraídas, se ha incluido la opción de normalizar dicho conjunto de imágenes, es decir, que todas las imágenes resultantes sean del mismo tamaño. Esta decisión

se justifica al ser un aspecto que influye en el comportamiento y rendimiento del proceso llevado a cabo por el módulo Clasificación y conteo.

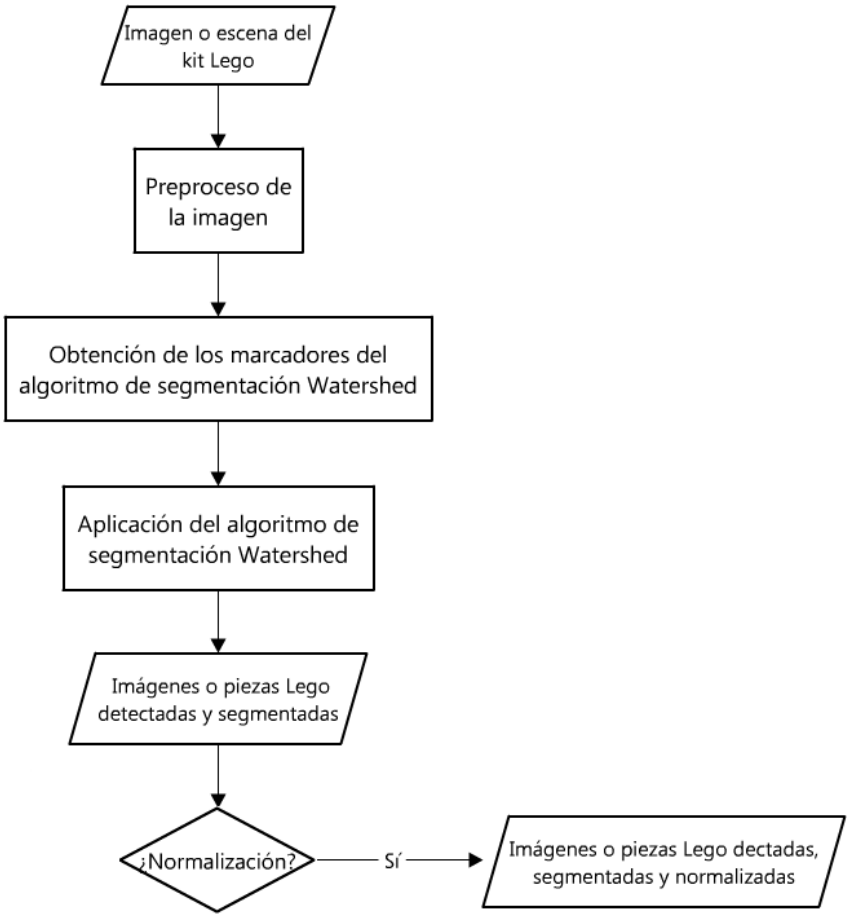


Figura 11Flujo del proceso llevado a cabo por el módulo Detección y segmentación

11.3 Módulo Clasificación y conteo

Como su propio nombre indica, es el encargado de realizar el proceso de clasificación y conteo de las piezas Lego del kit a auditar.

Parámetros de entrada:

- Conjunto de imágenes normalizadas (deigual tamaño) o piezas Lego del kit a auditar.
- Conjunto de imágenes de entrenamiento de los clasificadores, es decir, el conjunto de imágenes de muestra de cada clase o categoría que conforma dicho kit.

Ambos conjuntos de imágenes se obtienen por la detección, segmentación y normalización de una o varias escenas conteniendo los objetos o piezas Lego de interés.

Parámetro de salida:

- Listado conteniendo el resultado de dicho proceso, esto es, el número de piezas por categoríadel kitreconocidas, desconocidas y aquellas que no fueron posibles de clasificar al asignárselesmás de una categoría.

Dicho procesose divide para cada una de las imágenes del conjunto de piezas Lego del kit a auditar en las siguientes fases:

- Obtención de su vector de características.
- Clasificación y conteo de la misma.

11.3.1 Fase de obtención del vector de características

Dado el carácter experimental de este TFGemplea dos aproximaciones:

- **Aproximación estándar contrastada**: empleo del modelo generativo BOW para generar el vector de características. Al emplear un diccionario de palabras, conseguimos una mayor normalización de los resultados y reducimos la redundancia en los datos de entrenamiento. La construcción del diccionario se realiza a partir de los descriptores de imágenesobtenidos de un subconjunto de imágenes de entrenamiento.
- **Aproximación "naive" experimental**: El vector de características se obtiene a partir de un clusterizado del descriptor de imagen. Los principales motivos de emplear esta aproximación son tanto su simplicidad como en teoría un menor tiempo de entrenamiento con respecto a la aproximación anterior.

Teniendo en cuenta la necesidad de invariancia visual frente a rotaciones (piezas esparcidas en posición arbitraria) nuestra solución soporta el uso de los siguientes descriptores de imágenes:

- ✓ Detector Dense y como extractor bien SIFT o SURF.
- ✓ KAZE.
- ✓ AKAZE.
- ✓ Momentos de HU.

De esta forma podremos estudiar el descriptor que mejor se adapta nuestro problema.

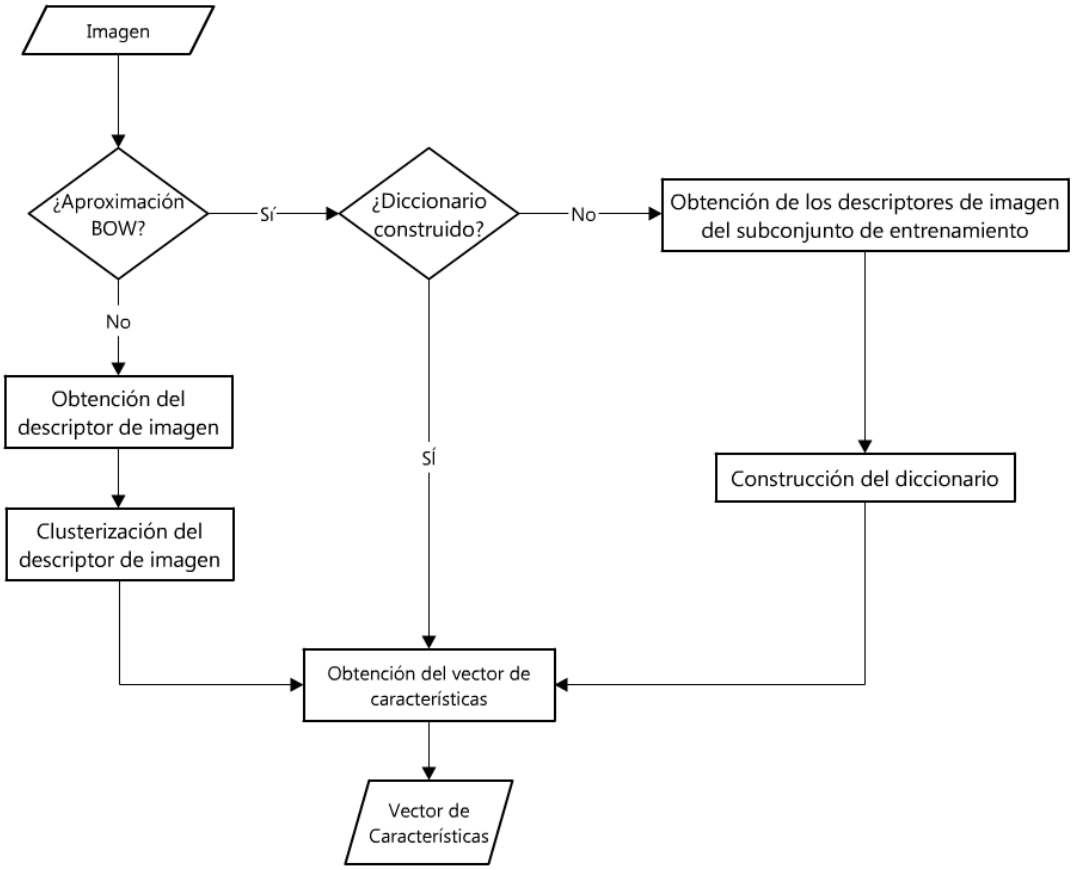


Figura 12Flujo del proceso llevado a cabo por la fase Obtención del vector de características

11.3.2 Fase de clasificación y conteo

Al ser el dominio de nuestro problema conocido, finito y de tamaño reducido, empleamos como clasificador el algoritmo de aprendizaje supervisado SVM. Por tanto, se requiere una fase previa de entrenamiento (véase subapartado 8.3.2 Clasificadores). Los datos de entrenamiento se obtendrán a partir del conjunto de vectores de características asociado al conjunto de imágenes de entrenamiento normalizado de cada SVM o clasificador.

De esta forma, una vez entrenadoel conjunto de clasificadores (tantos como categorías del kit Lego), procedemos a la clasificación y conteo de las piezas Lego del kit a auditar. Para ello, a partir del vector de características asociado a cada pieza, obtenemos el resultado de los clasificadores. En base al resultado obtenido procederemos a incrementar en una unidad el valor de:

- la categoría que lo representa sí solo lo identifica un clasificador,
- la categoría especial “dudoso” si lo identifica más de un clasificador, o
- la categoría especial “desconocido” en caso de no ser identificado.

Es importante que ambos conjuntos de imágenes (entrenamiento y clasificación) estén normalizados, es decir, las imágenes que lo integren sean del mismo tamaño. En caso

contrario, podríamos obtener vectores de características que no representen de forma fidedigna a dichos conjuntos.

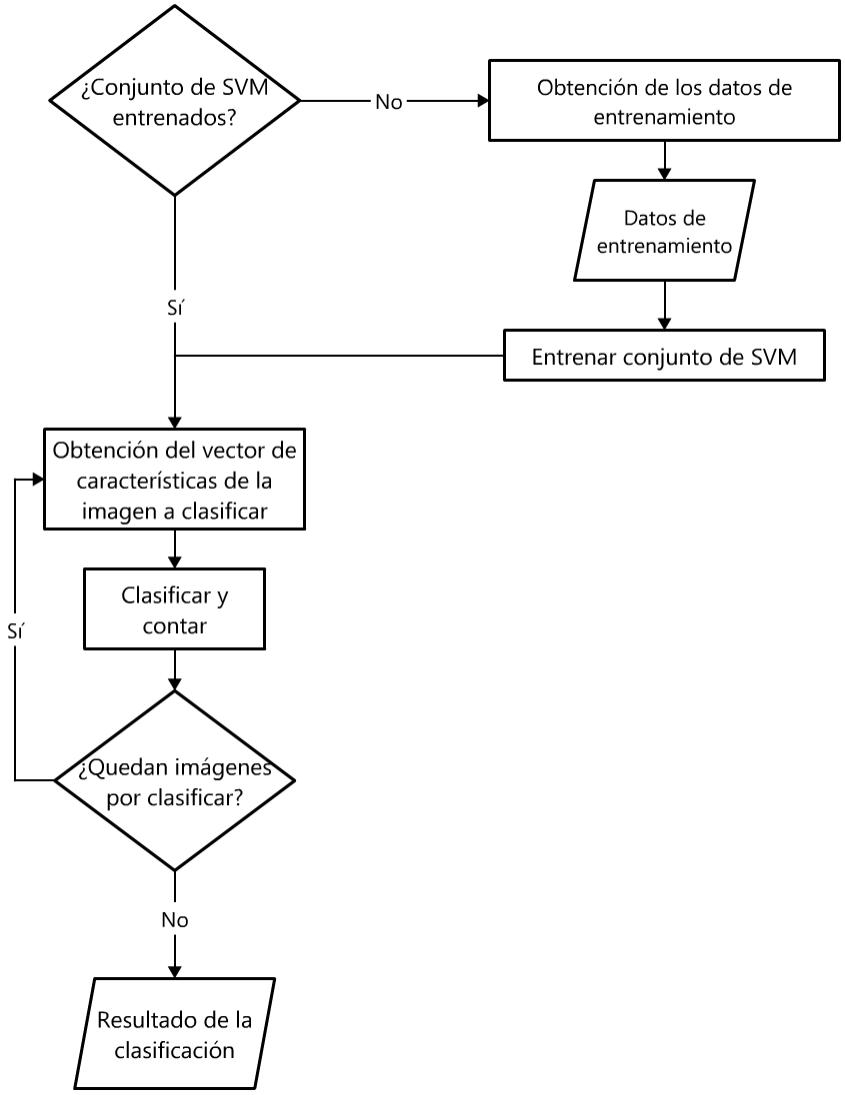


Figura 13 Flujo del proceso llevado a cabo por la fase Clasificación y conteo

12 Implementación de la solución

Como se vio en el apartado anterior, nuestra solución está formada por tres módulos independientes pero relacionados entre sí.

Aunque uno de ellos es principalmente de naturaleza física o *hardware*(“Adquisición de Imágenes”), los otros dos son claramente de naturaleza *software*y conforman la biblioteca o librería encargada de realizar nuestro proceso de “percepción visual”, es decir, el proceso de auditoría de los kits Lego.

Por tanto, en este apartado nos centraremos en los detalles de implementación dedicha biblioteca o librería que hemos denominado Inventariador.

12.1 Inventariador

Es la biblioteca o librería formada por los módulos “Detección y segmentación” y“Clasificación y conteo” de nuestra solución.

Implementadoconforme a lo expuesto en los apartados 10.Metodologíadedesarrolloy11. Diseñode la solución, se apoya en la librería OpenCV (véase subapartado 9.2 Recursos *software*) para realizar los procesos de visión artificial llevados a cabo por dichos módulos.

Características principales de nuestra librería:

- Escrita en el lenguaje de programación C++ y en Inglés.
- Módulo Detección y segmentación:
    - Normalización de las imágenes segmentadas.
    - Persistencia en disco de las imágenes segmentadas.
- Módulo Clasificación y conteo:
    - Dos modos de clasificación: “BOW-SVM” y “Naive-SVM”.
    - Admite descriptores de imágenes no soportados por la librería OpenCV[c]. P. ej. KAZE y AKAZE.
    - Permite el uso de descriptores de imágenes binarios.
    - Obtención de datos estadísticos de la sesión.
    - Persistencia en disco tanto de los resultados de la clasificación como de los datos estadísticos.
    - Guardar y recuperar los datos de sesión del clasificador (datos de entrenamiento y configuración).

A continuación indicaremos los detalles de implementación más relevantes de dichos módulos. Para ejemplos de su uso por parte del usuario puede consultar el Manual de usuario.

12.1.1 Módulo Detección y segmentación

Clase *`LegoImageSegmentation`*:

- Realiza el proceso de detección y segmentación de una escena o imagen que contiene piezas Lego del kit de la serie MindStorm.
- Para el preproceso de la imagen o escena utiliza las funcionesmorfológicas de apertura y cierre, de conversión del espacio de color, de equalización yde umbralizadoofrecidas por la librería OpenCV.
- Se apoya para obtener los marcadores del algoritmo de segmentación enlos métodos*`findContours()`*[26]y *`drawContours()`*[27]de dicha librería. El método*`findCountours()`* nos permite extraer en una imagen binaria en la queel fondo es representado por el valor 0los contornos de los objetos presentes en esta. Por otra parte, el método*`drawContours()`* nos permite dibujar los contornos o

[c] La interfaz del descriptor de imagen no soportado tiene que ser igual a la utilizada por dicha librería.

siluetas de dichos objetos, esto es, los marcadores del algoritmo de segmentación Watershed.

- Realiza la segmentación conforme a la implementación del algoritmo Watershed ofrecido por la mencionada librería.

- Nos permite:
    - Obtener el:
        - Listado de las imágenes o piezas Lego detectadas y segmentadas. Para ello, una vez aplicado el algoritmo de segmentación es necesario realizar un proceso de extracción de los objetos detectados y segmentados en la escena.
        - Número de imágenes detectadas y extraídas.
    - Normalizar los resultados, es decir, que todo el conjunto de imágenes extraídas sea conforme al tamaño indicado en píxeles. El nuevo conjunto de imágenes conserva su relación de aspecto original y los objetos son centrados en el marco que los contiene.
    - Persistir en disco el conjunto de imágenes extraídas. Para ello es necesario proporcionar tanto la ruta del directorio como el nombre de la categoría con el que se generarán los nombres de las imágenes de dicho conjunto.

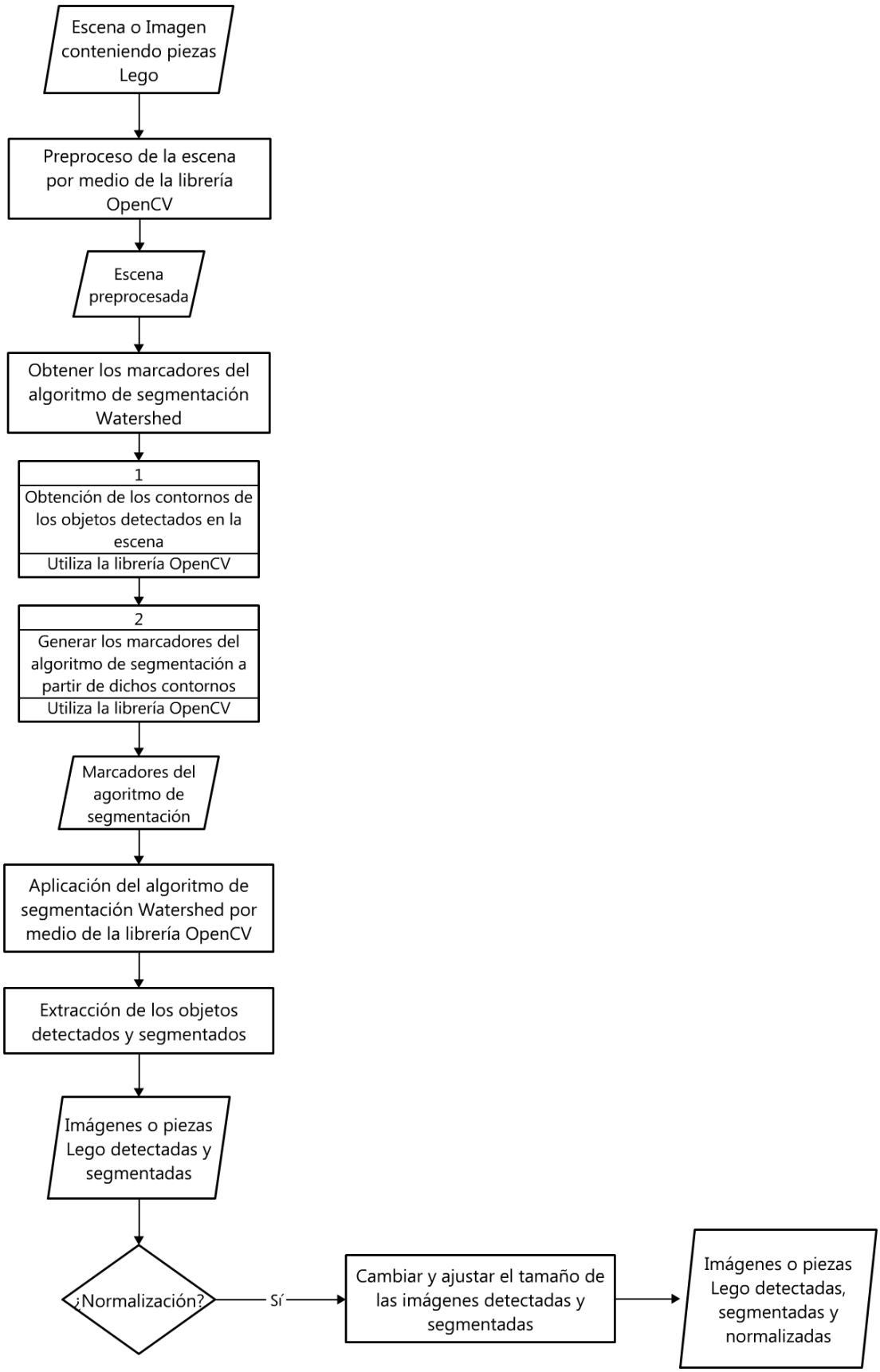


Figura 14 Flujo de la implementación del proceso llevado a cabo por la clase LegoImageSegmentation

12.1.2 Módulo Clasificación y conteo

A continuación, al igual que hicimos en el apartado 11. Diseño de la solución, indicaremos los detalles de implementación más relevantes de cada una de las fases en las que se divide dicho proceso: obtención del vector de características y clasificación y conteo.

12.1.2.1 Fase de obtención del vector de características

Aproximación estándar contrastada:

- Llevada a cabo por la clase *`BOWFeatureExtractor`*.
- Adapta el modelo BOW implementado por la librería OpenCV para poder usar descriptores de imágenes binario.
- Para la construcción del diccionario emplea la clase auxiliar *`BOWVocabulary`*.
- Utiliza la clase *`TrainImageSet`* para obtener el subconjunto de imágenes de entrenamiento del diccionario.
- Permite:
    - Establecer el tamaño de clúster del vocabulario.
    - Guardar y recuperar los distintos modelos generados en base a la configuración seleccionada.

Aproximación "naive" o experimental:

- Llevada a cabo por la clase*`NaiveFeatureExtractor`*.
- Conforme al tipo de descriptor de imagen, para realizar el clusterizado utiliza:
    - Descriptores de imagen no binarios: el método de la librería OpenCV *`kmeans`*().
    - Descriptores de imagen binarios: el método*`binary_cluster()`* de la clase *`KMedoid`*. Clase desarrollada por nosotros al no disponer la versión utilizada de la librería OpenCV de dicha característica.

Respecto a los descriptores de imágenes, ambas aproximaciones utilizan la clase abstracta FeatureExtractor:

- Diseñada conforme a los principios de diseño de orientación a objetos SOLID (véase apartado 10. Metodología de desarrollo).
- Hace uso de la herencia y del polimorfismo (aplicación del principio *Open Close*) para garantizar la extensión de la clase. Esto nos permitirá incorporar nuevos descriptores.
- Permite el uso de descriptores de imágenes no soportados por la librería OpenCV siempre que respeten la interfaz proporcionada por esta. En nuestro caso KAZE y AKAZE.
- Descriptores de imágenes soportados:
    - No binarios: KAZE, Dense + SIFT, Dense + SURF, SURF y Momentos de HU
    - Binarios: AKAZE

En el caso de los descriptores de imágenes KAZE y AKAZEmerecen una mención especial al representarcomo ya mencionamos en el apartado 8. Análisis del Problema un nuevo enfoque. A diferencia de los descriptores de imagen que para detectar y describir las características

construyen o emplean una aproximación del espacio de escala gausiano de la imagen,este tipo de descriptores realizan un blur adaptativo y local a los datos de la imagen. De esta forma, reducen el ruido pero respetan el contorno de los objetos, lo que se traduce en un aumento de la precisión enladescripción de dichos objetos. En cuanto a sus diferencias, en KAZE el espacio de escala mantiene la misma resolución, el descriptor de imagen no es binario y utiliza información de gradiente, mientras que en el caso de AKAZE el espacio de escala es piramidal y el descriptor de imagen es binario y utiliza información de intensidad y gradiente. Se integraron en nuestra solución como dos librerías que generamos a partir de su código, y en cuanto a su configuración, como gridempleamos para KAZE GSURF Extended y para AKAZE MLDB, ambos con invariancia frente a rotaciones.

Así mismo, fue necesario proveer un mecanismo que permitiera a nuestro lenguaje de programación (C++) instanciar en tiempo de ejecución objetos de la clase *FeatureExtractor*. Para ello empleamos una clase factoría (*FeatureExtractorFactory*), que por medio del uso de macros, al iniciarse el programa registra automáticamente las clases genéricas asociada a cada uno de los descriptores de imágenes soportados.Esta característica es requerida para poder cargar los datos de configuración del clasificador.

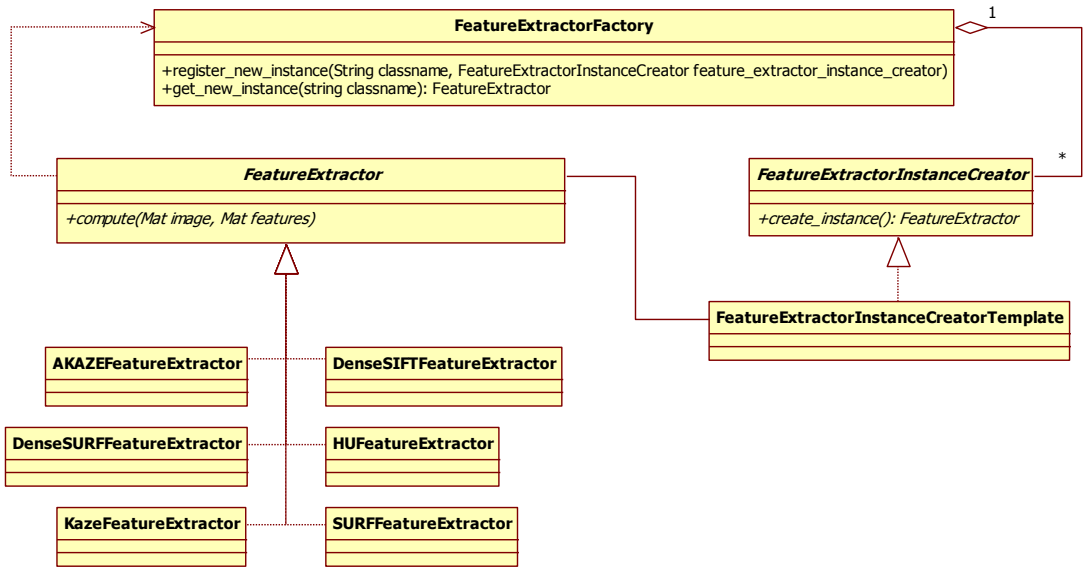


Figura 15 Diagrama de la clase FeatureExtractorFactory

Finalmente y como ya mencionamos previamente, la clase *KMedoid* nos permitió llevar a cabo el proceso de clusterización binario requerido por ambas aproximaciones. Esta es una funcionalidad claramente diferenciadora de nuestra solución al no ser soportada por la versión de librería OpenCV utilizada en este TFG.Características de la clase *KMedoid*:

- Implementa la técnica k-medoids conforme al algoritmo propuesto por H.S. Park [28]
- Criterios de parada:
    - Criterio de convergencia: diferencia de las sumas nulo.

- Criterio de no convergencia: bucle descubierto en la fase de testeo del algoritmo, en el que en cada iteración el valor de la diferencia de las sumas permanecía constante pero no nulo. Para evitar casos fortuitos o excepcionales se añadió un contador de número máximo de iteraciones consecutivas (margen de error). De esta forma, cuando se alcance dicho límite termina el algoritmo.El contador siempre se inicializa.
- Número máximo de iteraciones: añadimos este criterio ya que en ocasiones el tiempo que llevaría encontrar la solución puede ser no admisible. De esta forma, nos quedaríamos con una aproximación a la solución.

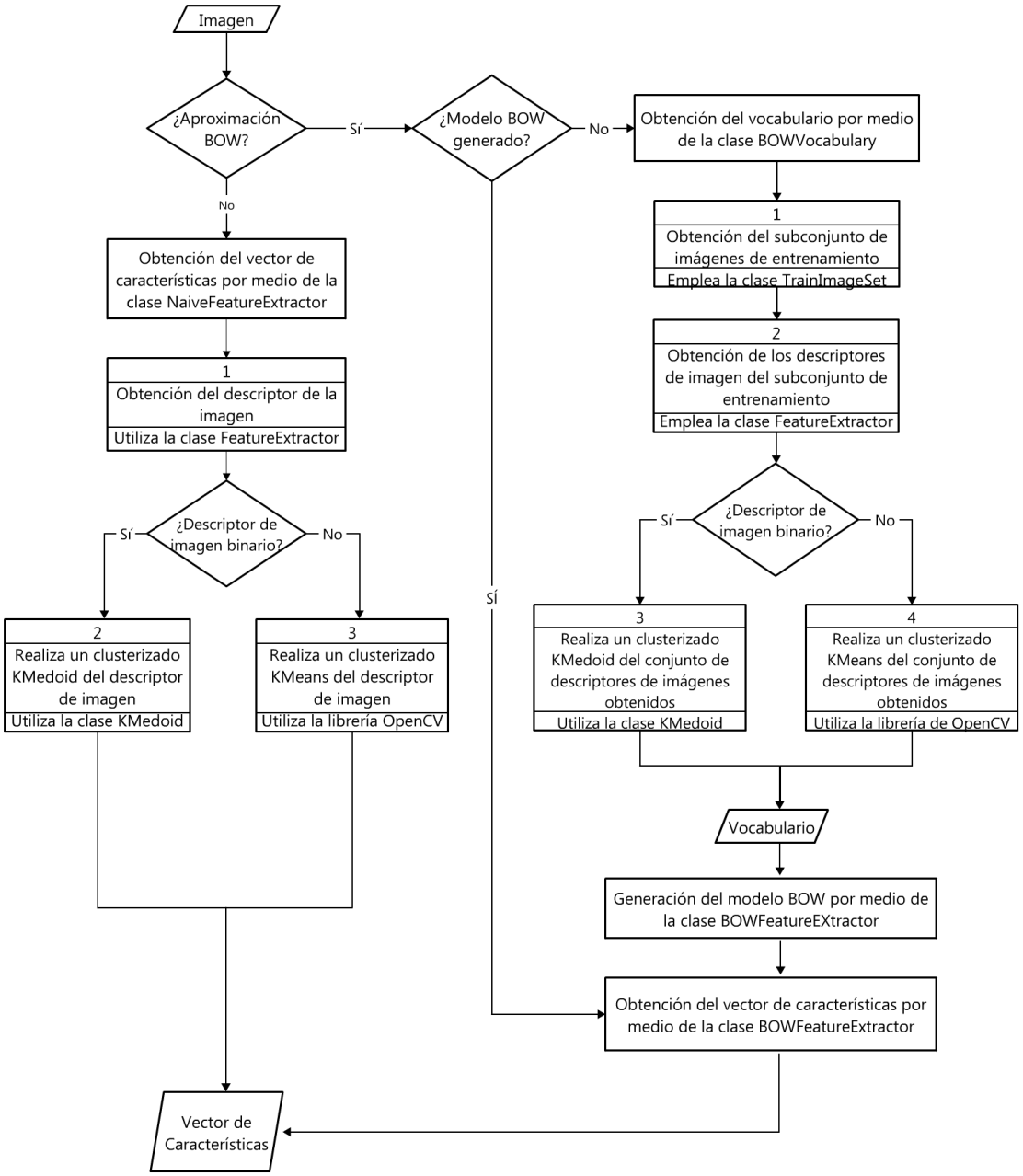


Figura 16 Flujo de la implementación de la fase de obtención del vector de características llevada a cabo por la clase Classifier

12.1.2.2 Fase de Clasificación y conteo

Clase *Classifier*:

- Responsable de controlar y realizar el proceso de clasificación y conteo.
- Emplea la clase *SVMSet* para llevar a cabo el proceso de clasificación.
- Por medio de la clase *TrainImageSet* obtiene el conjunto de imágenes de entrenamiento.
- Utiliza la clase *SVMTrainDataSet* para construir las muestras positivas y negativas del conjunto de SVM.
- Nos permite:
    - Configurar los siguientes parámetros:
        - El modo de clasificación en función de la aproximación elegida para la obtención del vector de características: "BOW-SVM" y "Naive-SVM".
        - El tipo de descriptor de imagen (DENSE+SIFT, DENSE+SURF, SURF, KAZE, AKAZE, Momentos de HU).
        - Tamaño del clúster del vocabulario para el modo de clasificación "BOW-SVM".
        - El tipo de núcleo del conjunto de SVM.
        - Los directorios del conjunto de imágenes de entrenamiento tanto para los clasificadores como para la construcción del diccionario en la aproximación BOW.
    - Obtener y guardar el resultado de la clasificación en disco mediante la generación de un archivo XML en el que se indica el número de piezas Lego por categoría detectada.
    - Guardar y recuperar los datos de sesión en disco. Por datos de sesión nos referimos a:modo de clasificación, tipo de descriptor de imagen, conjunto de SVM entrenado y el modelo BOW generado en caso de utilizar esta aproximación. Para ellogeneramos un archivo XML utilizando el modelo de persistencia ofrecido por la librería OpenCV.
    - Obtener y guardar datos estadísticos de la sesión, esto es, el tiempo de entrenamiento de los clasificadores y el tiempo de clasificación.Hace uso de la estructura *ClassifierStatisticsTime*. Los tiempos se expresan en segundos.

Estructura *ClassifierStatisticsTime*:

- Es utilizada por la clase *Classifier* para guardar los datos estadísticos de la sesión.
- Contiene dos campos: *training_time* (tiempo de entrenamiento) y *classification_time* (tiempo de clasificación).

Clase *SVMSet*:

- Representa el conjunto de SVM.
- Admite todos los núcleos soportados por la librería OpenCV (Lineal, Radial, Sigmoide y Polinomial).

- Usa el método de autocalibrado ofrecida por dicha librería para una configuración óptima de los núcleos.
- Emplea la clase *SVMTrainDataSet* para obtener el conjunto de muestras positivas y negativas a partir de los cuales construir el modelo de predicción. Para ello, generará el vector de etiquetas asociado a dicho conjunto. Ya que cada vector de característica se almacena en una fila de la matriz, por medio de dicho vector se especifica si dicha fila (índice en el vector) identifica a la categoría (valor 1) o no (valor -1).
- Nos permite:
    - Configurar el conjunto de SVM
    - Obtener:
        - El conjunto de SVM entrenados
        - Su estado (entrenado, no entrenado)

Clase *TrainImageSet*:

- Nos permite cargar y agrupar las imágenes por categoría.
- Cada categoría es representada por un subdirectorio del directorio principal. Como nombre de categoría usa el de dicho subdirectorio.
- Emplea el módulo *FileSystem* de la librería Boost para leer de forma recursiva las imágenes almacenadas en todos los subdirectorios de un directorio.
- Permite obtener:
    - El conjunto de imágenes agrupadas por categoría detectada
    - El listado de las categorías detectadas
    - Su estado (construido, no construido)

Clase *SVMTrainDataSet*:

- Construye las muestras positivas y negativas del conjunto de SVM en base al modo de clasificación seleccionado. Por muestras positivas, nos referimos al conjunto de vectores de característica asociado al conjunto de imágenes de entrenamiento que representa dicha categoría. El conjunto de muestras negativas de dicha categoría será las muestras positivas de las otras categorías.
- Utiliza la clase *TrainImageSet* para obtener el conjunto de imágenes de entrenamiento.
- Permite obtener:
    - El conjunto de muestras positivas de cada una de las categorías detectadas
    - El conjunto de muestras negativas de cada una de las categorías detectadas
    - El listado de categorías detectadas
    - Su estado (vacío, conjunto de muestras para entrenamiento generada)

Para garantizar el correcto funcionamiento de todo el proceso de clasificación y conteo, fue necesario implementar una lógica de control de estados en todas las clases intervinientes en dicho proceso. A modo ilustrativo en la figura podemos observar el diagrama de estados de la clase *Classifier*.

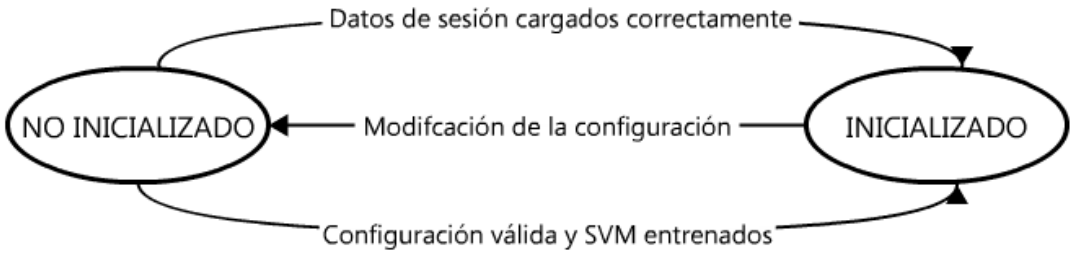


Figura 17 Diagrama de estado de la clase Classifier

Precisamente para facilitar el uso de esta característica, a excepción del clasificador que lo implementa en su lógica de control de estado, el resto de clases intervinientes en dicho proceso poseen un mecanismo de `clear`.

Finalmente, las categorías especiales (véase subapartado 11.3.2 Fase de clasificación y conteo) se implementaron con las siguientes traducciones: *Unkown* (desconocido) y *Wrong* (errónea). En este último caso, si bien la traducción *Wrong* no corresponde con dudosa, a efectos prácticos consideramos que el hecho de no poder determinar a qué categoría corresponde una determinada pieza es un fallo.

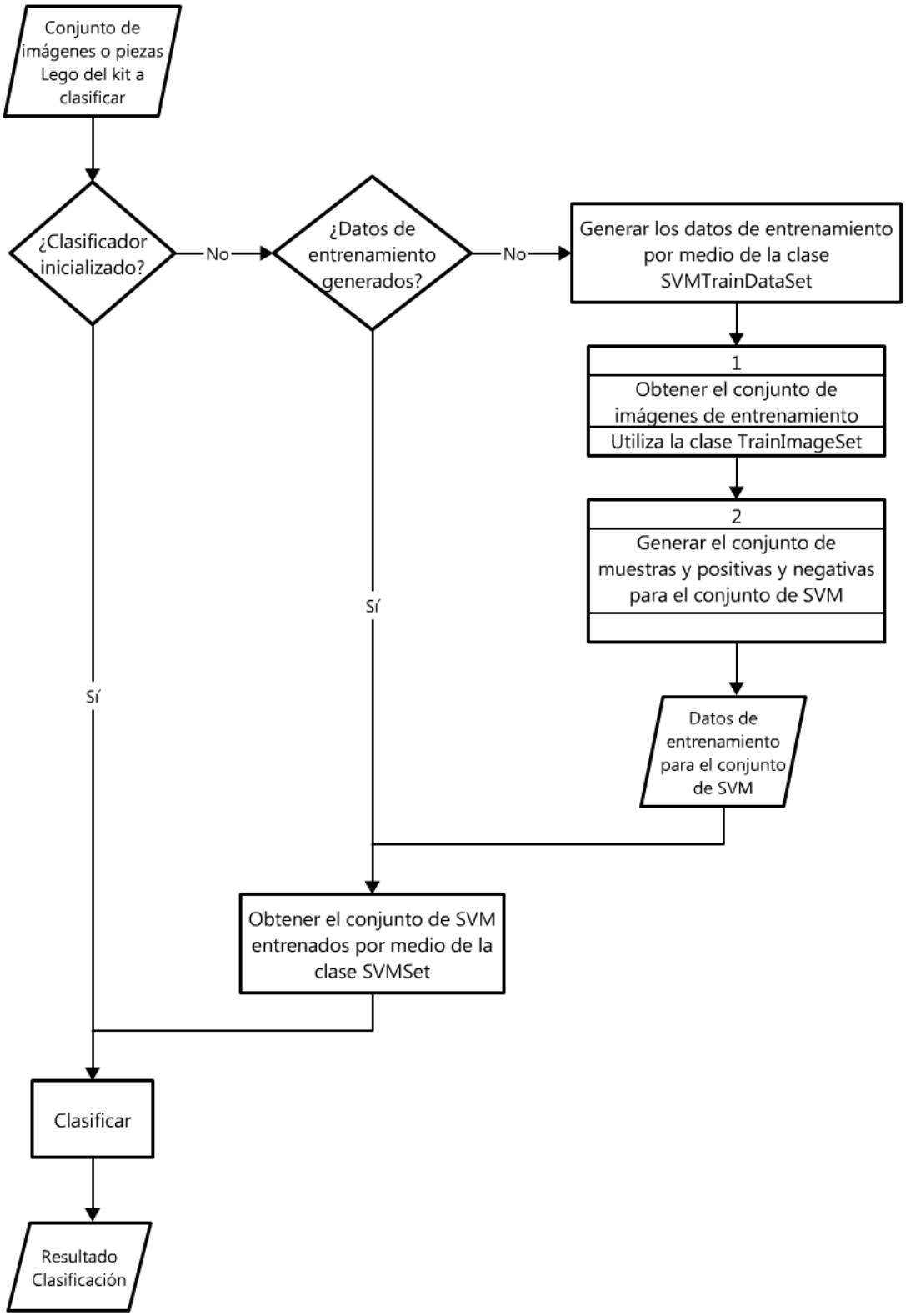


Figura 18Flujo de la implementación dela fase de clasificación y conteo llevado a cabo por la clase Classifier

13 Pruebas

De tipo funcional, se definieron para la solución *software*y en base al tipo de módulo a comprobar (“Detección y segmentación” y “Clasificación y conteo”). Además, en el caso del módulo de “Detección y clasificación”, dichas pruebas se realizaron con el objetivo de descubrir la configuración más óptima de nuestro prototipo de herramienta auditora. A este respecto, y debido a la naturaleza experimental de nuestro TFG, en dichas pruebas se procuró analizar y estudiar la influencia y relación de los distintos parámetros que intervienen en el proceso de clasificación y conteo, esto es, tipo de descriptor de imagen utilizado (AKAZE, KAZE, Dense-SIFT, Dense-SURF, SURF y Momentos de HU), modo de clasificación (BOW-SVMo Naive-SVM), tipo de núcleo usado a la hora de generar el modelo predictivo en los SVM (Lineal, Radial, Sigmoide, Polinomial), así como la influencia del tamaño del diccionario en el modo de clasificación BOW-SVM.

Finalmente, a pesar de queel conjunto de datos de pruebas se limitó considerablemente por las condiciones de trabajo impuestas al no disponer del dispositivo óptico apropiado (véase subapartado 11.1 Módulo Adquisición de imágenes), se mantuvieron lasdos categoríascuya auditoría, dado el enorme esfuerzo y dificultad que entraña para las personas que lo llevan a cabo, condujo a la realización de este TFG. Dichas categorías corresponden a las piezas Lego de tipo conector corto y largo respectivamente, y su dificultad radica en ser piezas muy similares y de elevado número(64 por kit para el conector corto).Por tanto, nuestro conjunto de datos de pruebas está formado por tres categorías: conector corto, conector largo y eje 2. Esta última nos servirá como parámetro de control.

13.1.1 Prueba del módulo Detección y segmentación

Datos de prueba:7 imágenes o escenas por cada una de las categorías de prueba. Cada escena o imagen contiene 15 objetos o piezas Lego en posición arbitraria y respetando las restricciones de no contacto ni oclusión mencionadas en apartados previos.

Caso de prueba: comprobar los procesos de detección y segmentación, así como el de normalización.

Diseño de la prueba: para cada escena o imagen, se registró tanto el número de piezas obtenidas como el tiempo consumido por ambos procesos. Para el proceso de normalizaciónse comprobó que el tamaño de la imagen normalizada correspondía con el especificado, en nuestro caso 150 píxeles.

Criterio de aceptación: el número de imágenes obtenidas coincide con el número de objetos o piezas Lego presentes en la escena o imagen origina, y el tamaño de la imagen normalizada coincide con el indicado.

13.1.2 Prueba del módulo Clasificación y conteo

Datos de prueba: se usa el conjunto de imágenes normalizadas obtenidas en las pruebasdel módulo de segmentación. En base al modo de clasificación, dicho conjunto de imágenes se estructura de la siguiente forma:

- BOW-SVM:
    - Diccionario: 75 imágenes (25 imágenes representativas de cada categoría).
    - Entrenamiento: para cada SVM (uno por categoría), 25 imágenes positivas (imágenes representativas de su categoría) y 50 negativas (el conjunto total de imágenes positivas de las otras dos categorías).
    - Datos de prueba: 150 imágenes (50 imágenes por cada categoría).
- Naive-SVM:
    - Entrenamiento: para cada SVM (uno por categoría), 25 imágenes positivas (imágenes representativas de su categoría) y 50 negativas (el conjunto total de imágenes positivas de las otras dos categorías).
    - Datos de prueba: 150 imágenes (50 imágenes por cada categoría).

Caso de prueba: comprobar para cada una de las configuraciones de prueba, la tasa de acierto, esto es, el número de objetos o piezas Lego correctamente identificadas, así como medir el tiempo empleado en las fases de entrenamiento y de clasificación y conteo.

Configuraciones de prueba: combinación de cada uno de los 6 descriptores disponibles (AKAZE, KAZE, Dense-SIFT, Dense-SURF, SURF y Momentos de HU) con cada uno de los tipos de núcleos disponibles para generar el modelo predictivo del SVM (Lineal, Radial, Sigmoide, Polinomial), y en el caso del modo de clasificación BOW-SVM además para diccionario con tamaño de clúster 500, 1000 y 1250. Se obtuvieron 96 configuraciones distintas (véase tablas 1 al 5).

Diseño de la prueba: para cada una de las configuraciones de prueba, se registró el resultado del clasificador, así como los tiempos llevado a cabo por las etapas de entrenamiento y de clasificación y conteo.

Criterio de aceptación: no se registran piezas en la categoría de "*unknown*"o "*wrong*" (véase subapartado 12.1.2 Módulo Clasificación y conteo) y el número de piezas registradas por categoría es de 50.

Configuración de prueba del clasificador
Modo de clasificación Naive-SVM, Núcleo Lineal, Descriptor de imagen AKAZE
Modo de clasificación Naive-SVM, Núcleo Lineal, Descriptor de imagen KAZE
Modo de clasificación Naive-SVM, Núcleo Lineal, Descriptor de imagen Dense-SIFT
Modo de clasificación Naive-SVM, Núcleo Lineal, Descriptor de imagen Dense-SURF
Modo de clasificación Naive-SVM, Núcleo Lineal, Descriptor de imagen SURF
Modo de clasificación Naive-SVM, Núcleo Lineal, Descriptor de imagen Momentos de HU
Modo de clasificación Naive-SVM, Núcleo Radial, Descriptor de imagen AKAZE
Modo de clasificación Naive-SVM, Núcleo Radial, Descriptor de imagen KAZE
Modo de clasificación Naive-SVM, Núcleo Radial, Descriptor de imagen Dense-SIFT
Modo de clasificación Naive-SVM, Núcleo Radial, Descriptor de imagen Dense-SURF
Modo de clasificación Naive-SVM, Núcleo Radial, Descriptor de imagen SURF
Modo de clasificación Naive-SVM, Núcleo Radial, Descriptor de imagen Momentos de HU
Modo de clasificación Naive-SVM, Núcleo Sigmoide, Descriptor de imagen AKAZE
Modo de clasificación Naive-SVM, Núcleo Sigmoide, Descriptor de imagen KAZE
Modo de clasificación Naive-SVM, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT
Modo de clasificación Naive-SVM, Núcleo Sigmoide, Descriptor de imagen Dense-SURF
Modo de clasificación Naive-SVM, Núcleo Sigmoide, Descriptor de imagen SURF
Modo de clasificación Naive-SVM, Núcleo Sigmoide, Descriptor de imagen Momentos de HU
Modo de clasificación Naive-SVM, Núcleo Polinomial, Descriptor de imagen AKAZE
Modo de clasificación Naive-SVM, Núcleo Polinomial, Descriptor de imagen KAZE
Modo de clasificación Naive-SVM, Núcleo Polinomial, Descriptor de imagen Dense-SIFT
Modo de clasificación Naive-SVM, Núcleo Polinomial, Descriptor de imagen Dense-SURF
Modo de clasificación Naive-SVM, Núcleo Polinomial, Descriptor de imagen SURF
Modo de clasificación Naive-SVM, Núcleo Polinomial, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen AKAZE

Tabla 1 Configuraciones de prueba del clasificador - 1

Configuración de prueba del clasificador
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen Dense-SIFT

Tabla 2Configuraciones de prueba del clasificador - 2

Configuración de prueba del clasificador
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen SURF

Tabla 3Configuraciones de prueba del clasificador - 3

Configuración de prueba del clasificador
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen AKAZE

Tabla 4Configuraciones de prueba del clasificador - 4

Configuración de prueba del clasificador
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen Momentos de HU
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen AKAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen KAZE
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen Dense-SIFT
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen Dense-SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen SURF
Modo de clasificación BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen Momentos de HU

Tabla 5Configuraciones de prueba del clasificador - 5

14 Resultados

En este apartado indicaremos los resultados de las pruebas del apartado anterior.

Si bien por detalles de implementación todos los resultados del conjunto de pruebas han sido generados en inglés, en aras de mantener una coherencia con el estilo de redacción seguido hasta ahora, los datos serán presentados en castellano y desde distintos enfoques.

En el CD se encuentran tanto el conjunto de datos de pruebas, como los resultados de las mismas. Así mismo se han incluido los datos de sesión para cada una de las configuraciones de prueba del clasificador. De esta forma y dado el componente investigativo que también tiene este TFG logramos que este experimento sea reproducible.

14.1 Resultado de las prueba del módulo Detección y segmentación

Indicaremos a continuación y a modo ilustrativo, el resultado de realizar para cada una de las categorías de prueba tanto el proceso de detección y segmentación como el de su posterior

normalización. Como se mencionó al comienzo de este apartado, el total del conjunto de imágenes generadas puede ser consultado en el CD.

Figura 19 Escena de la categoría Conector corto

Figura 20 Pieza conector corto extraída

Figura 21 Pieza conector corto extraída y normalizada

Figura 22 Escena categoría Conector largo

Figura 23Pieza conector largo extraída

Figura 24 Pieza conector largo extraída y normalizada

Figura 25 Escena categoría Eje 2

Figura 26 Pieza conector eje 2 extraída

Figura 27 Pieza conector eje 2 extraída y normalizada

Finalmente, en la tabla 6 se detalla tanto el número total de piezas extraídas como los tiempos promedio de detección y segmentación y de normalización obtenidos para cada una de las categorías de prueba.

Categoría	Tiempo promedio de segmentación (s)	Tiempo promedio de normalización (s)	Total piezas extraídas
Conector corto	19,28	0,05	105
Conector largo	28,11	0,04	105
Eje 2	26,70	0,04	105

Tabla 6 Resultado de la prueba del módulo Detección y segmentación

14.2 Resultado de la prueba del módulo Clasificación y conteo

Tal y como se mencionó al comienzo de este apartado, en las tablas 7 al 26 se detalla el resultado obtenido para cada una de las configuraciones de prueba de los clasificadores, ordenados en función de su tasa de acierto (número de piezas correctamente identificadas en cada una de las categorías de prueba) y en orden decreciente.

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen KAZE	472	465	49	50	48	0	3	**98,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen KAZE	465	464	50	50	46	2	2	**97,33%**
Modo Naive-SVM, Núcleo Polinomial, Descriptor de imagen Dense-SURF	145	60	49	49	46	5	1	**96,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen KAZE	475	462	47	51	48	4	0	**96,00%**
Modo Naive-SVM, Núcleo Radial, Descriptor de imagen KAZE	462	453	47	49	46	2	6	**94,67%**

Tabla 7 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 1

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo Naive-SVM, Núcleo Polinomial, Descriptor de imagen KAZE	537	453	45	48	49	3	5	**94,67%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen KAZE	475	462	47	49	46	4	4	**94,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen KAZE	460	462	47	50	45	2	6	**94,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen KAZE	494	466	52	47	47	1	3	**94,67%**
Modo Naive-SVM, Núcleo Lineal, Descriptor de imagen KAZE	455	454	45	50	46	3	6	**94,00%**

Tabla 8 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 2

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo Naive-SVM, Núcleo Radial, Descriptor de imagen Dense-SIFT	**1523**	**1510**	**46**	**45**	**50**	**4**	**5**	**94,00%**
Modo Naive-SVM, Núcleo Lineal, Descriptor de imagen Dense-SURF	61	60	50	41	49	3	7	**93,33%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen Dense-SURF	225	298	48	49	43	9	1	**93,33%**
Modo Naive-SVM, Núcleo Radial, Descriptor de imagen Dense-SURF	68	60	49	41	49	3	8	**92,67%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen KAZE	461	462	44	50	45	3	8	**92,67%**

Tabla 9 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 3

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen KAZE	461	462	46	51	44	6	3	**92,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen KAZE	474	462	45	48	45	6	6	**92,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen AKAZE	155	42	50	46	38	8	8	**89,33%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen AKAZE	155	42	50	46	38	8	8	**89,33%**
Modo Naive-SVM, Núcleo Radial, Descriptor de imagen SURF	12	4	46	39	48	8	9	**88,67%**

Tabla 10 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 4

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen Dense-SIFT	2280	2047	49	49	35	4	13	**88,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen AKAZE	867	43	49	47	37	4	13	**88,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen AKAZE	170	42	50	47	36	7	10	**88,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen Dense-SURF	227	306	47	43	43	10	7	**88,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen Dense-SIFT	4277	2145	50	50	32	5	13	**88,00%**

Tabla 11 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 5

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen AKAZE	155	42	50	47	35	9	9	**88,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen Dense-SURF	211	299	52	39	43	9	7	**86,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen Dense-SURF	225	299	47	39	44	9	11	**86,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen Dense-SIFT	2779	2076	50	48	31	5	16	**86,00%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen Dense-SURF	212	302	53	39	43	8	7	**86,00%**

Tabla 12 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 6

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen Dense-SURF	209	295	52	39	42	11	6	**86,00%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen SURF	9	10	43	40	44	10	13	**84,67%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen AKAZE	506	43	51	47	30	9	13	**84,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen AKAZE	171	42	47	44	35	5	19	**84,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen SURF	24	10	51	31	46	5	17	**84,00%**

Tabla 13 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 7

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen SURF	12	11	44	40	40	5	21	**82,67%**
Modo Naive-SVM, Núcleo Polinomial, Descriptor de imagen SURF	90	4	38	39	46	15	12	**82,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen SURF	32	12	36	48	38	1	27	**81,33%**
Modo Naive-SVM, Núcleo Polinomial, Descriptor de imagen Dense-SIFT	1603	1515	43	41	37	20	9	**80,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen SURF	18	11	36	46	39	12	17	**80,67%**

Tabla 14 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 8

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen SURF	9	10	36	45	39	18	12	**80,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen AKAZE	624	43	49	44	26	0	31	**79,33%**
Modo Naive-SVM, Núcleo Lineal, Descriptor de imagen SURF	5	4	43	32	43	15	17	**78,67%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen AKAZE	170	42	45	42	31	12	20	**78,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen SURF	9	9	48	30	40	8	24	**78,67%**

Tabla 15 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 9

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen SURF	24	9	48	39	30	15	18	**78,00%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen SURF	25	10	36	39	42	21	12	**78,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen Dense-SURF	397	337	50	36	30	18	16	**77,33%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen Dense-SIFT	1794	1888	38	37	32	22	21	**71,33%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen Dense-SURF	604	350	44	38	23	18	27	**70,00%**

Tabla 16 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 10

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo Naive-SVM, Núcleo Polinomial, Descriptor de imagen AKAZE	129	44	31	40	33	13	33	**69,33%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen Dense-SIFT	1770	1868	36	39	25	15	35	**66,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen Momentos de HU	2	0	36	35	28	0	51	**66,00%**
Modo Naive-SVM, Núcleo Lineal, Descriptor de imagen Dense-SIFT	1524	1512	38	20	40	22	30	**65,33%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen Momentos de HU	18	0	37	29	32	0	52	**65,33%**

Tabla 17 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 11

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo Naive-SVM, Núcleo Lineal, Descriptor de imagen AKAZE	47	45	31	35	31	24	29	**64,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen Dense-SIFT	1792	1871	48	15	34	8	45	**64,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen Dense-SURF	1233	409	39	54	9	2	46	**62,67%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Lineal, Descriptor de imagen Momentos de HU	0	0	30	32	28	0	60	**60,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Radial, Descriptor de imagen Momentos de HU	2	0	33	27	26	0	64	**57,33%**

Tabla 18 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 12

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen Momentos de HU	2	0	28	32	25	0	65	**56,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Polinomial, Descriptor de imagen Dense-SIFT	1799	1903	37	23	25	18	47	**56,67%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Polinomial, Descriptor de imagen Momentos de HU	18	0	34	25	24	0	67	**55,33%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Radial, Descriptor de imagen Dense-SIFT	1777	1882	50	0	32	0	68	**54,67%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Lineal, Descriptor de imagen Momentos de HU	0	0	27	27	25	0	71	**52,67%**

Tabla 19 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 13

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo Naive-SVM, Núcleo Lineal, Descriptor de imagen Momentos de HU	0	0	39	62	0	0	49	**51,33%**
Modo Naive-SVM, Núcleo Radial, Descriptor de imagen Momentos de HU	2	0	39	62	0	0	49	**51,33%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Polinomial, Descriptor de imagen Momentos de HU	18	0	30	30	5	0	85	**43,33%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Lineal, Descriptor de imagen Momentos de HU	0	0	35	25	0	0	90	**40,00%**
Modo Naive-SVM, Núcleo Polinomial, Descriptor de imagen Momentos de HU	25	0	37	8	8	39	58	**35,33%**

Tabla 20 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 14

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Radial, Descriptor de imagen Dense-SIFT	1779	1885	50	0	0	0	100	**33,33%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen Dense-SURF	216	303	0	42	0	0	108	**28,00%**
Modo Naive-SVM, Núcleo Radial, Descriptor de imagen AKAZE	53	45	0	13	12	0	125	**16,67%**
Modo Naive-SVM, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT	1528	1508	0	0	24	3	123	**16,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT	1779	1888	0	0	21	0	129	**14,00%**

Tabla 21 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 15

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo Naive-SVM, Núcleo Sigmoide, Descriptor de imagen AKAZE	59	44	0	0	0	0	150	**0,00%**
Modo Naive-SVM, Núcleo Sigmoide, Descriptor de imagen KAZE	475	454	0	0	0	0	150	**0,00%**
Modo Naive-SVM, Núcleo Sigmoide, Descriptor de imagen Dense-SURF	81	60	0	0	0	0	150	**0,00%**
Modo Naive-SVM, Núcleo Sigmoide, Descriptor de imagen SURF	25	4	0	0	0	0	150	**0,00%**
Modo Naive-SVM, Núcleo Sigmoide, Descriptor de imagen Momentos de HU	5	0	0	0	0	0	150	**0,00%**

Tabla 22 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 16

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen AKAZE	157	42	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen KAZE	462	462	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT	1779	1881	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen Dense-SURF	215	303	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen SURF	11	9	0	0	0	0	150	**0,00%**

Tabla 23 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 17

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 500, Núcleo Sigmoide, Descriptor de imagen Momentos de HU	3	0	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen AKAZE	157	42	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen KAZE	463	462	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen Dense-SIFT	1773	1874	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen SURF	11	9	0	0	0	0	150	**0,00%**

Tabla 24 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 18

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasificación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 1000, Núcleo Sigmoide, Descriptor de imagen Momentos de HU	3	0	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen AKAZE	157	43	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen KAZE	463	462	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen Dense-SURF	214	302	0	0	0	0	150	**0,00%**
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen SURF	11	10	0	0	0	0	150	**0,00%**

Tabla 25 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 19

Configuración de prueba del clasificador	Tiempo de entrenamiento (s)	Tiempo de clasficación (s)	Conector Corto	Conector Largo	Eje 2	Wrong	Unknown	Tasa de acierto
Modo BOW-SVM, Tamaño del clúster 2500, Núcleo Sigmoide, Descriptor de imagen Momentos de HU	3	0	0	0	0	0	150	**0,00%**

Tabla 26 Resultado de la prueba del módulo Clasificación y conteo ordenado en función de su tasa de acierto – 20

En las figuras 28 al 31 podemos observar datos estadísticos de los resultados referentes a los tiempos de entrenamiento y de clasificación.

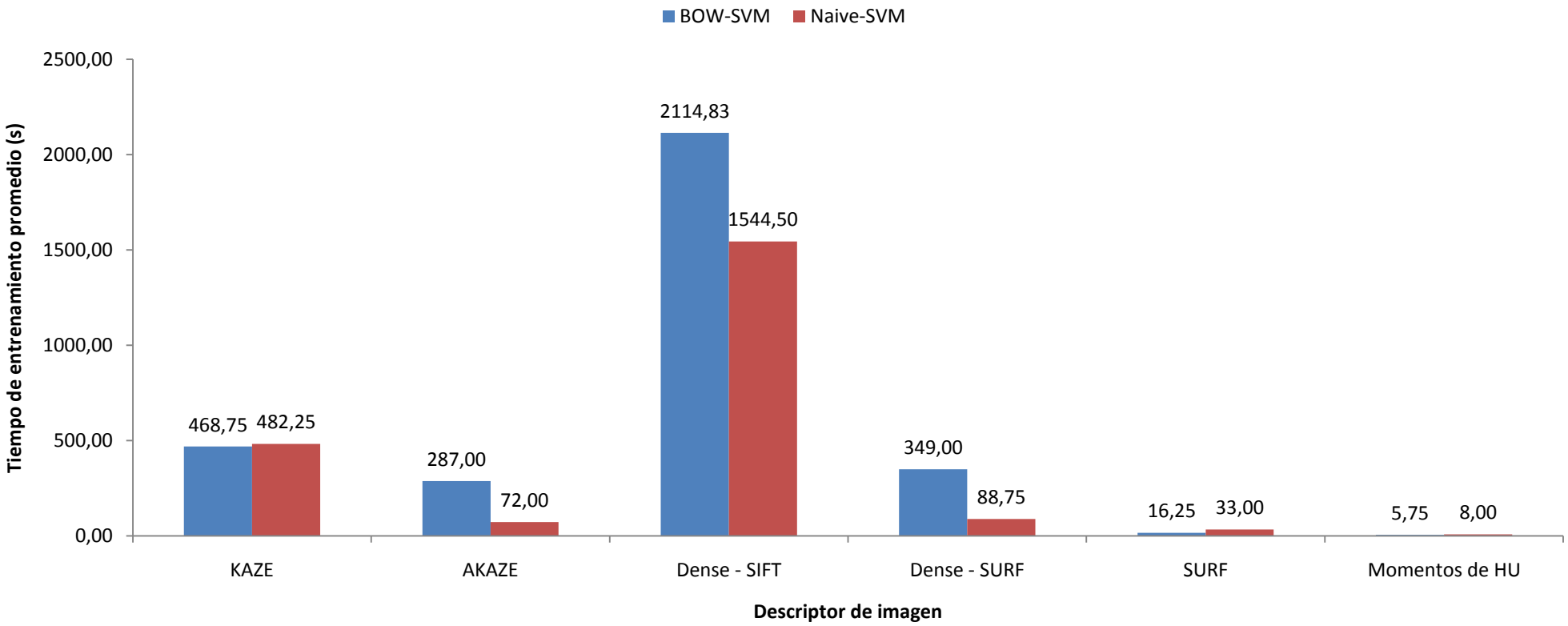


Figura 28 Tiempo de entrenamiento promedio de los modos de clasificación en función del descriptor de imagen utilizado

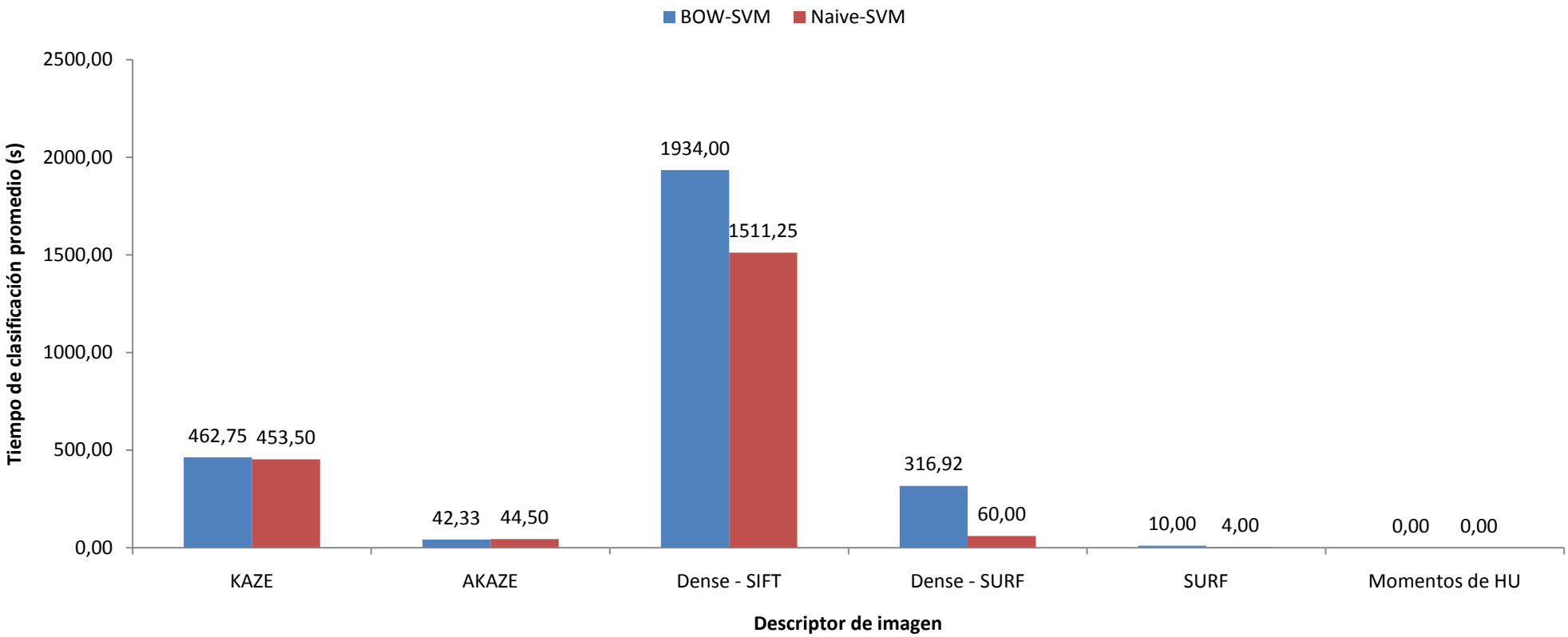


Figura 29 Tiempo de clasificación promedio de los modos de clasificación en función del descriptor de imagen utilizado

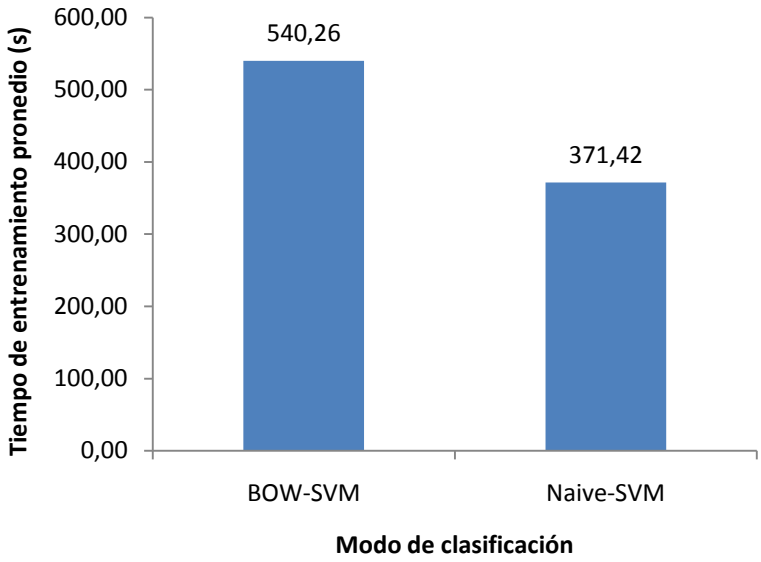


Figura 30 Tiempo de entrenamiento promedio en función del modo de clasificación

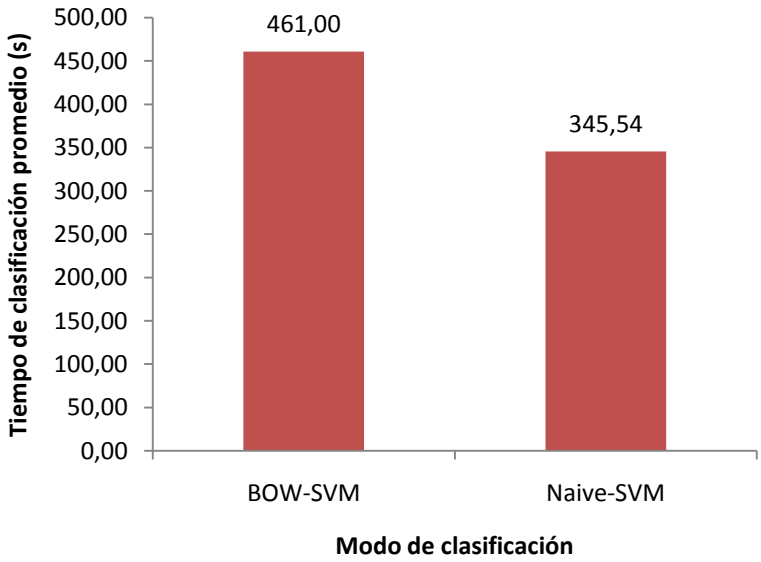


Figura 31 Tiempo de clasificación promedio en función del modo de clasificación

En las figuras 32 al 34podemos observar datos estadísticos de los resultados referentes a la tasa de acierto promedio.

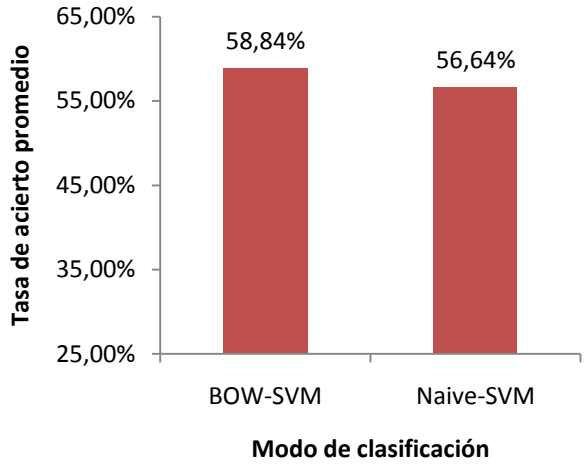


Figura 32 Tasa de acierto promedio según el modo de clasificación

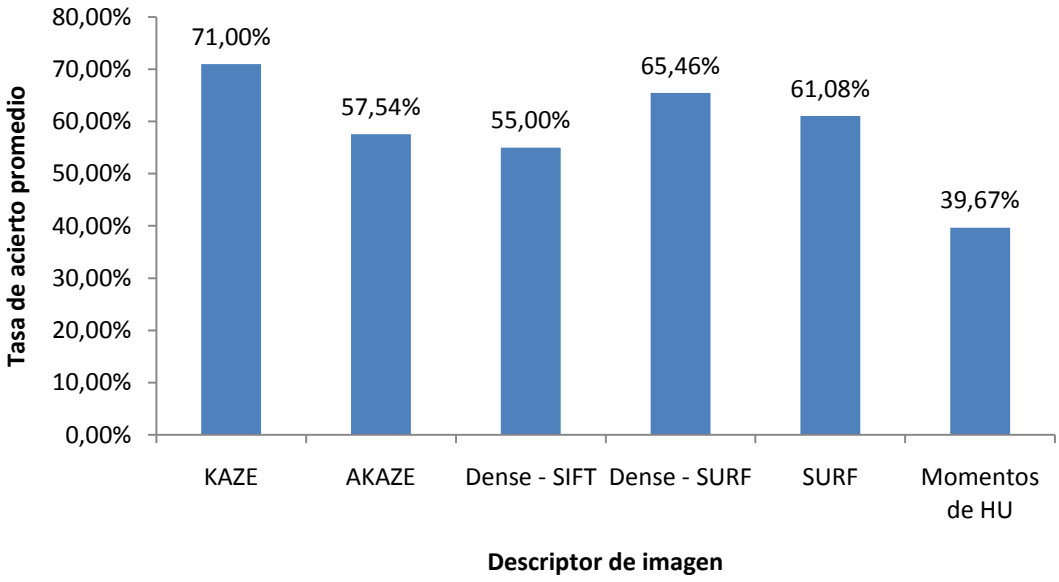


Figura 33 Tasa de acierto promedio global en función del descriptor de imagen utilizado

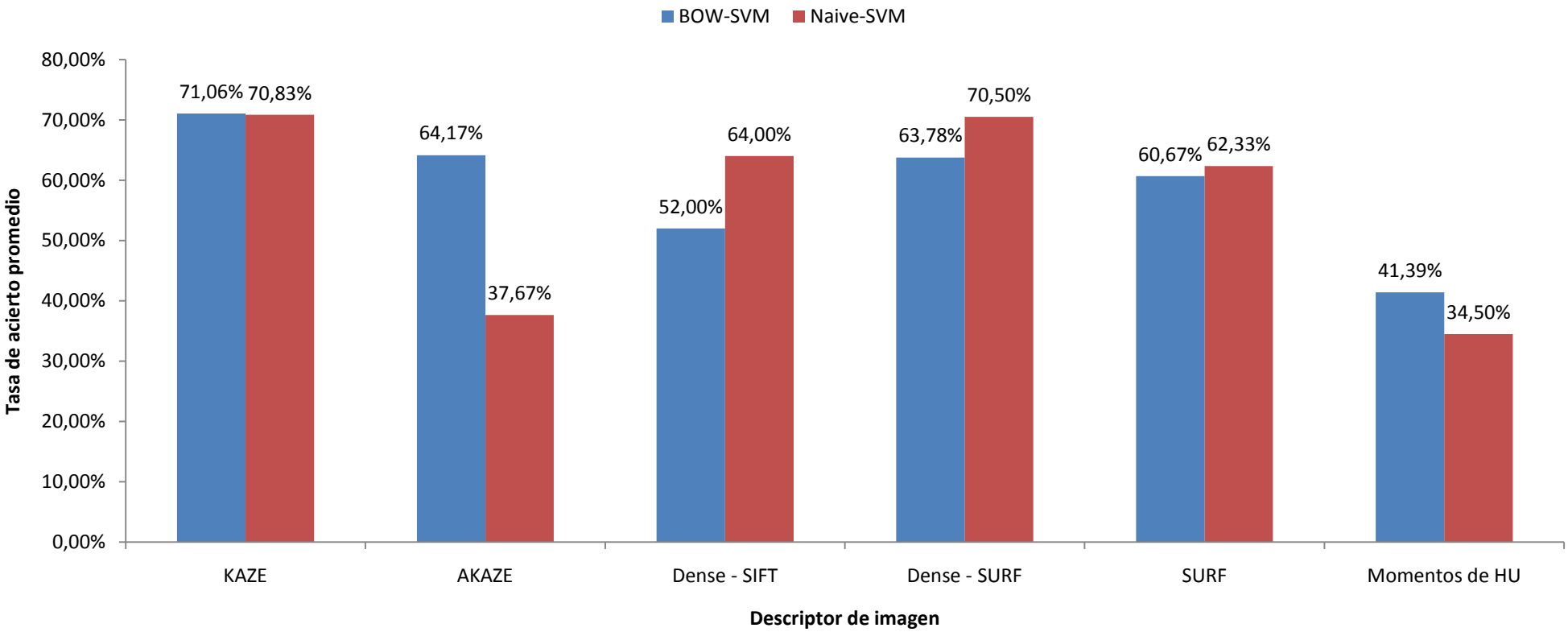


Figura 34Tasa de acierto promedio de los modos de clasificación en función del descriptor de imagen utilizado

En las figuras 35 al 40 podemos observarcómo influye el tamaño del clúster del diccionario en la tasa de aciertodel modo de clasificación BOW-SVM. No se ha incluido los núcleos en los que su tasa de acierto para todas sus configuraciones es nula.

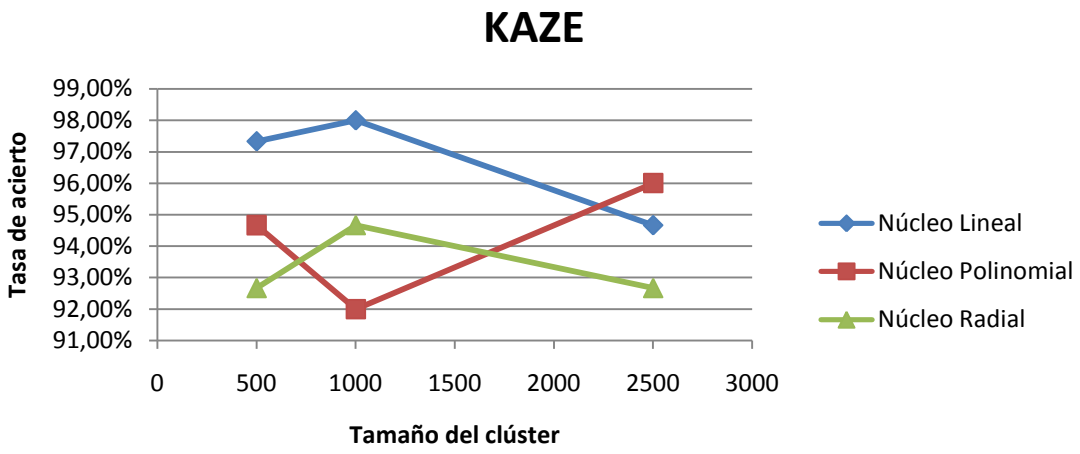


Figura 35Influencia en el modo de clasificación BOW-SVM del tamaño del clúster del diccionario en la tasa de acierto utilizando el descriptor de imagen KAZE

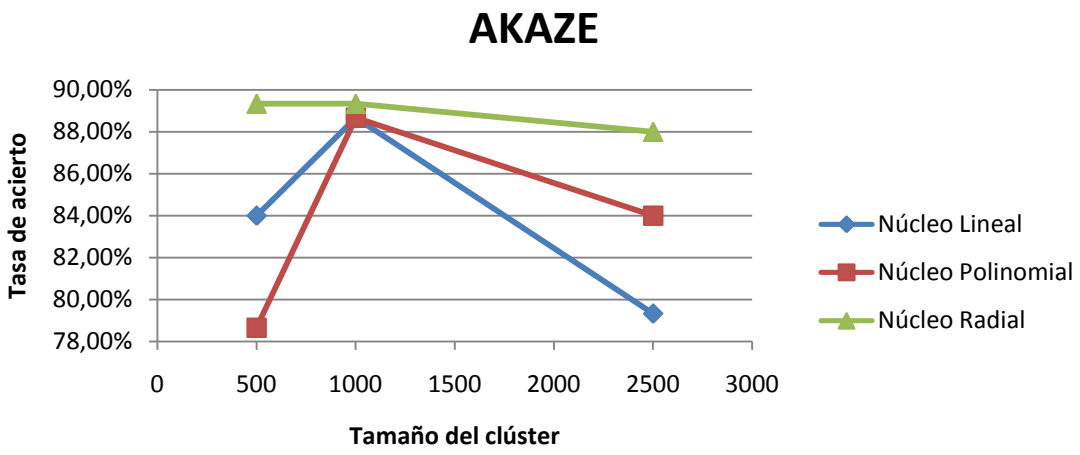


Figura 36 Influencia en el modo de clasificación BOW-SVM del tamaño del clúster del diccionario en la tasa de acierto utilizando el descriptor de imagen AKAZE

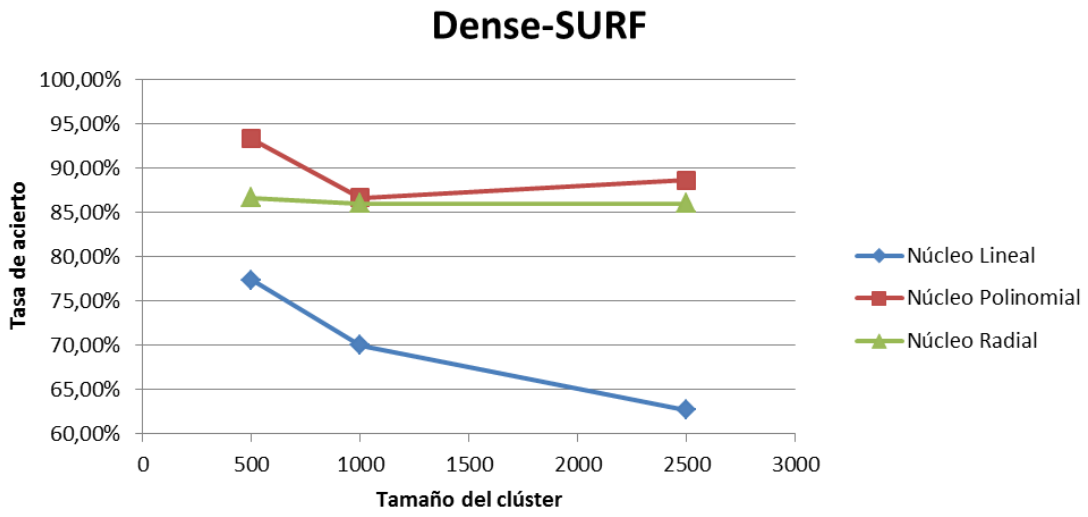


Figura 37Influencia en el modo de clasificación BOW-SVM del tamaño del clúster del diccionario en la tasa de acierto utilizando el descriptor de imagen Dense-SURF

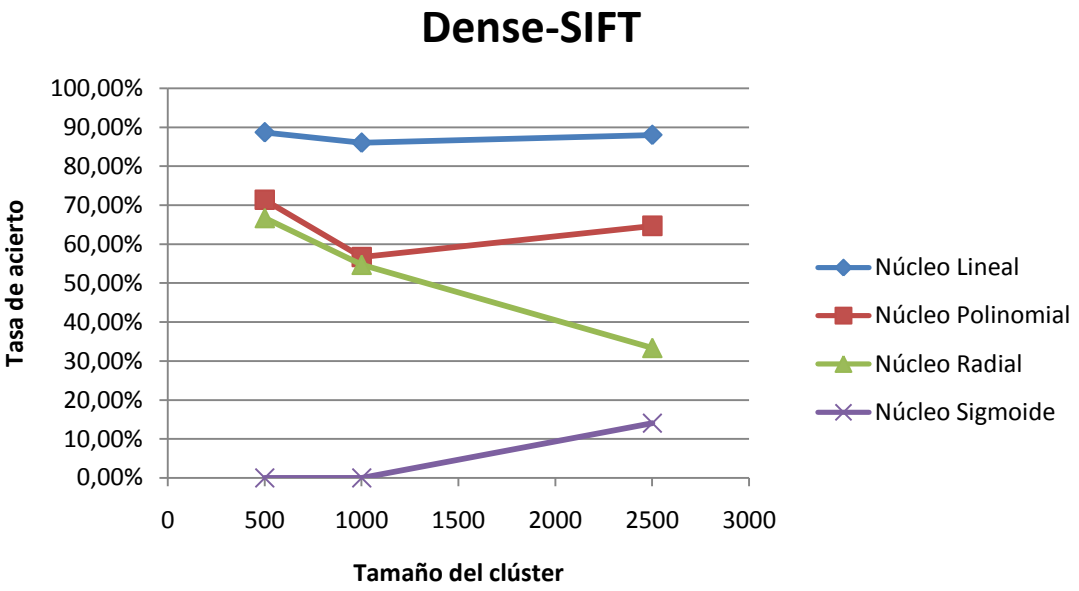


Figura 38Influencia en el modo de clasificación BOW-SVM del tamaño del clúster del diccionario en la tasa de acierto utilizando el descriptor de imagen Dense-SIFT

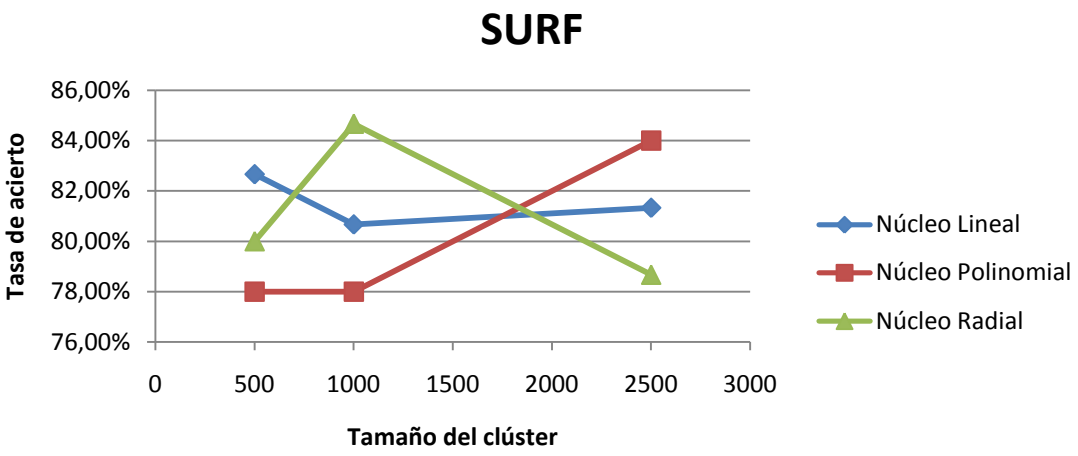


Figura 39Influencia en el modo de clasificación BOW-SVM del tamaño del clúster del diccionario en la tasa de acierto utilizando el descriptor de imagen SURF

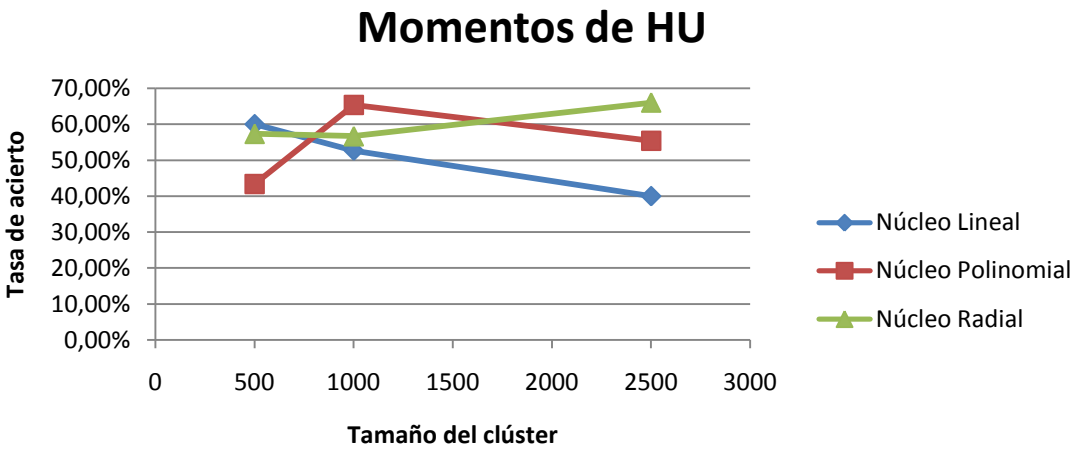


Figura 40Influencia en el modo de clasificación BOW-SVM del tamaño del clúster del diccionario en la tasa de acierto utilizando el descriptor de imagen Momentos de HU

Finalmente en la Figura 41 podemos observar cómo influye en función del modo de clasificación cada uno de los núcleos en la tasa de acierto. Para ello se ha calculado la tasa de acierto promedio de cada uno de los núcleos en base al modo de clasificación.

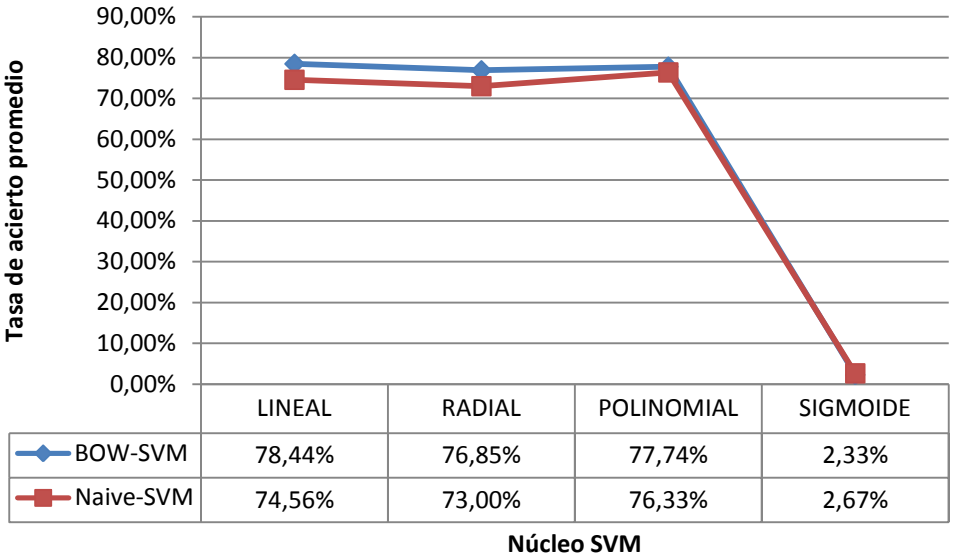


Figura 41 Influencia en función del modo de clasificación del núcleo SVM en la tasa de acierto

15 Conclusiones

De los resultados de la prueba realizada al módulo Detección y Segmentación podemos inferir que su comportamiento ha sido el esperado y el tiempo del proceso de normalización en promedio no supone un coste excesivo.

De los resultados de la prueba del módulo Clasificación y conteo podemos inferir que:

- La configuración con modo de clasificación BOW, tamaño de clúster 500, núcleo Lineal y descriptor de imagen KAZE presenta la máxima tasa de acierto obtenida (98 %).
- En el caso del modo de clasificación BOW-SVM el tamaño del clúster del diccionario influye en la tasa de acierto. Dicha variación depende estrechamente del descriptor de imagen y tipo de núcleo SVM utilizado.
- El tipo de núcleo SVM influye en la tasa de acierto. En nuestro caso particular, se observa claramente que el núcleo que peor modela nuestro problema es el Sigmoide. De hecho su tasa de acierto es prácticamente nula. Por el contrario, los otros núcleos no presentan una gran diferencia.
- La tasa de acierto promedio del modo de clasificación BOW-SVM es 1,10 % superior con respecto al modo de clasificación Naive-SVM.
- El tiempo promedio de entrenamiento en el modo de clasificación Naive-SVM es un 31,25 % menor con respecto al modo de clasificación BOW-SVM.
- El tiempo promedio de clasificación en el modo de clasificación Naive-SVM es un 25,04 % menor con respecto al modo de clasificación BOW-SVM.
- Las configuraciones que emplean el descriptor de imagen KAZE son las que mejor tasa de acierto promedio obtienen (71%).

Por tanto, se propone como configuración para nuestra futura herramienta auditora, la configuración de prueba que emplea el modo de clasificación BOW, tamaño de clúster 500, núcleo Lineal y descriptor de imagen KAZE al obtener la máxima tasa de acierto registrada (98 %).

Así mismo, a nivel profesional este TFG ha supuesto:

- Enfrentarme a una disciplina compleja y nunca vista anteriormente. Esto ha supuesto un gran desafío, tanto desde el punto de vista humano (acceso a la información, presentación de la información,…) como técnico (véase a modo de ejemplo el subapartado 11.1 Módulo Adquisición de imágenes).
- Emplear un nuevo lenguaje de programación y tenerlo incluso que adaptar a mis necesidades al tener que proveer un mecanismo que me permitiera hacer uso de una característica no soportada por dicho lenguaje (véase subapartado 12.1.2.1 Fase de detección de características).
- Conocer, adaptar y extender la funcionalidad de una librería como OpenCV que es marco de referencia en el campo de visión artificial. A este respecto, he adaptado el modelo BOW que implementa dicha librería para permitir tanto el uso de descriptores de imágenes binarios como de descriptores de imágenes no soportados siempre que se respete su interfaz. Así mismo la he dotado de un mecanismo de clusterización binaria por medio de la técnica *k-medoids*.
- Construir un sistema de visión artificial basado en la librería OpenCV para detectar, segmentar y clasificar piezas Lego. Si bien, me hubiera gustado poder contar con una óptica adecuada que me hubiese permitido testear mejor las configuraciones e incluir otras categorías, he obtenido un sistema con características interesantes de cara a la investigación y experimentación propias de este campo:
    - Normalización de las imágenes segmentadas.
    - Capacidad de guardar y recuperar los datos de sesión del clasificador (datos de entrenamiento y de configuración).
    - Admite el uso de descriptores de imágenes no soportados por dicha librería siempre que se respete su interfaz.
    - Permite el uso de descriptores de imágenes binarios.
    - Técnica de clusterización binaria *k-medoids*.
- Ponerme en contacto y colaborar con expertos en este campo como Pablo Alcantarilla (véase apartado 6. Aportaciones), autor de los descriptores de imágenes KAZE y AKAZE.

Respecto al sistema de visión artificial construido, me gustaría señalar que:

- En la obtención del vector de características, la aproximación “Naive” es fruto de mi curiosidad investigativa, en aras de simplificar y optimizar en términos de tiempo la aproximación “BOW”. En este sentido , el modo de clasificación que emplea la aproximación “Naive” obtiene:
    - ✓ En promedio un tiempo total del proceso (entrenamiento más clasificación) un 28,39 % menor con respecto al modo de clasificación con aproximación BOW.

- ✓ La tercera configuración con mejor tasa de acierto (96 %).

- El descriptor de imagen Momentos de HU fue propuesto por uno de nuestros tutores pero adaptado para poder ser usado por ambas aproximaciones en OpenCV.

Finalmente, ha sido un esfuerzo arduo y aunque queda mucho camino por correr (véase Trabajo Futuro) me siento especialmente orgulloso de que parte de este TFG sea útil a la comunidad. A este respecto y como se mencionó en el apartado 6. Aportaciones, participaremos en el OpenCV Vision Challenge, que tiene como objetivo la actualización y extensión de OpenCV.

16 Trabajo Futuro

Como ya se mencionó en el apartado 15. Conclusiones, este TFG supone el comienzo de la futura herramienta de visión artificial para auditar los kits Lego de la serie MindStorm.

Por tanto, indicaremos a continuación algunas propuestas de mejora de cada uno de los módulos que integran nuestra solución:

- Módulo de adquisición de imágenes: contar con un dispositivo óptico apropiado.
- Módulo de detección y segmentación: ajustar el proceso para permitir el contacto entre piezas, nuevas condiciones de iluminación y distinto fondo.
- Módulo de clasificación y conteo:
  - Realizar un ajuste y optimización de las configuraciones de prueba con mejor tasa de acierto.
  - Admitir la clasificación en color, ya que hay piezas Lego que solo se distinguen por su color.
  - Explorar nuevos paradigmas de clasificación no contemplados.

Con carácter general:

- Diseñar una interfaz de usuario multiplataforma que facilite su uso y acceso.
- Estudiar la posibilidad de ampliar la clasificación a otros kits de robótica.

17 Fuentes de información

[1] Apliquality Engineering, «Blog INFAIMON,» 13 05 2014. [En línea]. Available: http://blog.infaimon.com/2014/05/contador-de-objectos-en-contacto/.

[2] MathWorks, «MATLAB,» [En línea]. Available: http://es.mathworks.com/solutions/image-video-processing/.

[3] MVTec Software GmBH , «HALCON,» [En línea].

[4] EURESYS, «EasyObject,» [En línea]. Available: http://www.euresys.com/Products/MachineVisionSoftware/EasyObject.asp.

[5] «Generación de contenidos digitales,» [En línea]. Available: http://es.creativecommons.org/blog/licencias/.

[6] Otsu's method, [En línea]. Available: http://en.wikipedia.org/wiki/Otsu%27s_method.

[7] J. Canny, «A Computational Approach to Edge Detection,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 679-698, 1986.

[8] F. Meyer, « Color Image Segmentation,» 1992.

[9] M. C. a. R. S. Stefan Leutenegger, «BRISK: Binary Robust Invariant Scalable Keypoints,» *ICCV,* pp. 2548-2555, 2011.

[10] V. R. K. K. G. R. B. Ethan Rublee, «ORB: An efficient alternative to SIFT or SURF,» *ICCV,* pp. 2564-2571, 2011.

[11] D. G. Lowe, «“Distinctive Image Features from Scale-Invariant Keypoints”,» *International Journal of Computer Vision,* pp. 91-110, 2004.

[12] C. Mikolajczyk y C. Schmid, « "A performance evaluation of local descriptors",» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 1615-1630, 2005.

[13] H. Bay, A. Ess, T. Tuytelaars y L. Van Gool, «Speeded-Up Robust Features (SURF),» 2008.

[14] P. F. Alcantarilla, A. Bartoli y A. J. Davison, «KAZE Features,» [En línea]. Available: http://www.robesafe.com/personal/pablo.alcantarilla/papers/Alcantarilla12eccv.pdf.

[15] P. F. Alcantarilla, J. Nuevo y A. Bartoli, «AKAZE Features,» [En línea]. Available: http://www.robesafe.com/personal/pablo.alcantarilla/papers/Alcantarilla13bmvc.pdf.

[16] Wikipedia, «Bag of Words model in computer vision,» [En línea]. Available: http://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision.

[17] Wikipedia, «Support Vector Machine,» [En línea]. Available: http://en.wikipedia.org/wiki/Support_vector_machine.

[18] Wikipedia, «KMeans,» [En línea].

[19] «Microsoft Office 2010,» [En línea]. Available: http://office.microsoft.com/es-es.

[20] «Lovely Charts Online Edition,» [En línea]. Available: http://www.lovelycharts.com/web.

[21] «Microsoft Visual Studio 2013,» [En línea]. Available: http://msdn.microsoft.com/es-es/vstudio/aa718325.aspx.

[22] «FTP Filezilla,» [En línea]. Available: https://filezilla-project.org/.

[23] «OpenCV,» [En línea]. Available: http://opencv.org/.

[24] «Boost C++ Libraries,» [En línea]. Available: http://www.boost.org/.

[25] R. C. Martin, «The principles of OOD,» [En línea]. Available: http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod.

[26] «OpenCV - findContours(),» [En línea]. Available: http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#findcontours.

[27] «OpenCV - drawContours(),» [En línea]. Available: http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#drawcontours.

[28] H. Park y C. Jun, «A simple and fast algorithm for K-medoids clustering,» *Expert Systems with Applications,* p. 3336–3341, 2009.

[29] «OpenCV - CvSVMParams,» [En línea]. Available: http://docs.opencv.org/modules/ml/doc/support_vector_machines.html#cvsvmparams.

[30] «OpenCV - Clase FileStorage,» [En línea]. Available: http://docs.opencv.org/modules/core/doc/xml_yaml_persistence.html#filestorage.

[31] P. Viola y M. Jones, «Rapid Object Detection using a Boosted Cascade Simple Features,» de *Conference on Computer Vision and Pattern Recognition*, 2001.

[32] K. E. A. Van de Sande, T. Gevers y C. G. M. Snoek, «Evaluating Color Descriptors for Object and Scene Recognition,» 2009. [En línea]. Available: http://www.colordescriptors.com.

[33] S. J. Prince, Computer Vision: models, learning and inference, Cambridge University Press, 2012.

[34] C. Pantofaru y M. Hebert, «A comparison of Image Segmentation Algorithms. The Robotics Institute. Carnegie Mellon University,» September 2005. [En línea]. Available: http://www.ri.cmu.edu/pub_files/pub4/pantofaru_caroline_2005_1/pantofaru_caroline_2005_1.pdf.

[35] OpenCV, «Image Segmentation with Watershed Algorithm,» [En línea]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_watershed/py_watershed.html.

[36] S. Liao, X. Zhu, Z. Lei, L. Zhang y S. Z. Li, «Learning Multi-scale Block Local Binary Patterns for Face Recognition,» de *International Conference on Biometrics (ICB)*, 2007.

[37] J. Howse, «OpenCV Computer Vision with Python,» 2013, p. 65.

[38] Wikipedia, «Feature detection (computer vision),» [En línea]. Available: http://en.wikipedia.org/wiki/Feature_detection_%28computer_vision%29.

[39] C. Evans, «Notes on the Open Surf Library,» 2009. [En línea]. Available: http://www.cs.bris.ac.uk/Publications/Papers/2000970.pdf.

[40] «CVonline: Vision Related Books including Online Books and Book Support Sites,» [En línea]. Available: http://homepages.inf.ed.ac.uk/rbf/CVonline/books.htm.

[41] N. Cristianini y J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, 1 ed., Cambridge University Press, 2000.

[42] «Course Image Based Recognition and Classification. KTH Royal Institute of Technology,» [En línea]. Available: http://www.csc.kth.se/utbildning/kth/kurser/DD2427/bik12/index.php.

[43] A. J. Chavez, *Image classification with Dense Sift sampling: an exploration of optimal parameters,* K. S. University, Ed., 2012.

[44] Ó. Boullosa García, *Estudio comparativo de descriptores visuales para la detección de escenas cuasi-duplicadas,* Universidad Autónoma de Madrid, 2011.

[45] A. L. Barczac, *Toward and Efficient Implementation of a Rotation Invariant Detector Using Haar-like Features,* Dunedin: Institute of Information and Mathematical Sicences. Massey University, 2005, pp. 31-36.

[46] A. Cerda, «Alternativas de licenciamiento de software libre y open source. Análisis legal,» 2005. [En línea]. Available: https://www.derechosdigitales.org/29/alternativas-de-licenciamiento-de-software-libre/.

[47] A. Fernández Moreno, «LA PROPIEDAD INTELECTUAL: PASADO, PRESENTE Y... ¿FUTURO?,» 2012. [En línea]. Available: http://www.eumed.net/rev/cccss/22/propiedad_intelectual.html .

[48] J. J. Zuta Ortiz, «Tipos de licencia para software,» 2011. [En línea]. Available: http://www.monografias.com/trabajos88/tipos-licencias-software/tipos-licencias-software.shtml.

18 Anexo I Manual de usuario

En este anexo indicaremos cómo hacer uso de nuestra librería Inventariador.

Explicaremos tanto los pasos a seguir si queremos poder usarla en un proyecto (Instalación),como el procedimiento para llevar a cabo los procesos de detección y segmentación y de clasificación y conteo soportados por esta.

18.1 Instalación

Se debe incluir el código fuente de nuestra librería en el proyecto que la vaya usar.

Así como tener en dicho proyecto correctamente referenciado las librerías externas de las que depende:

- OpenCV 2.4.9.
- Módulo FileSystem de la librería Boost1.55.
- Librería de los descriptores de imágenes KAZE y AKAZE (se incluyen en el CD al ser generadas por nosotros).

18.2 Detección y segmentación

Suponiendo que desea detectar y segmentar esta escena conteniendo Piezas Lego:

Figura 42 Manual de usuario - Escena a segmentar

Debe incluir en el fichero cabecera de la clase del proyecto en la que desea realizar dicho proceso, o en la clase donde se encuentra el programa principal la clase*`LegoImageSegmentation`*.

A continuación, por medio del método de openCV `imread()` cargue la escena o imagen anterior. Para ello, debe suministrar la ruta donde se encuentra la imagen o escena y el objeto de la clase Mat en donde desee almacenarla:

```cpp
#include <iostream>
#include <opencv2/opencv.hpp>
#include "LegoImageSegmentation.h"

using namespace cv;
using namespace std;
using namespace inventariador;

int main() {
    // Cargamos la escena o imagen que vamos a segmentar
    Mat scene = imread("p:/example/scene.jpg");

    // Instanciamos un objeto de la clase LegoImageSegmentation
    LegoImageSegmentation lego_image_segmentation = LegoImageSegmentation();
```

Figura 43 Extracto de código de carga de una escena e instanciación de un objeto de la clase LegoImageSegmentation

Una vez realizado esto, lo siguiente será declarar e instanciar un objeto de la clase *`LegoImageSegmentation`* e invocar su método *segment()*. Este recibe como parámetro de entrada el objeto de la clase `Mat` (pasado por referencia) conteniendo la escena o imagen a segmentar.

Si desea obtener el conjunto de imágenes detectadas y segmentadas para poder utilizarlas en otra sección del programa, invoque al método `get()de dicho objeto`. Le devolverá un vector de objetos de la clase Mat (clase utilizada por OpenCV para representar imágenes).

```cpp
    // Segmentamos la escena o imagen cargada anteriormente y mostramos por pantalla una de las imágenes
    lego_image_segmentation.segment(scene);
    vector<Mat> images_extracted = lego_image_segmentation.get();
    namedWindow("Objeto extraído", WINDOW_NORMAL | CV_WINDOW_KEEPRATIO);
    imshow("Objeto extraído", images_extracted[5]);
    waitKey(0);
```

Figura 44 Captura de código que realiza el proceso de segmentación, obtiene el conjunto de imágenes segmentadas y muestra una por pantalla.

Por el contrario, si desea normalizar el conjunto de imágenes detectadas y segmentadas (requerido para llevar a cabo el proceso de clasificación y conteo), invoque al método *`normalize_size_of_images_extracted()`*. Recibe como parámetro de entrada un entero que expresa el ancho en píxeles de dicho conjunto. Como se mencionó en el subapartado 12.1.1 Módulo Detección y segmentación, el conjunto de imágenes normalizada mantiene su relación de aspecto y los objetos son centrados en el marco que los contiene. Además, este es un proceso irreversible, una vez invocado el conjunto original de imágenes detectadas y segmentadas se pierde.

Para usar el nuevo conjunto de imágenes detectadas, segmentadas y normalizadas en otra sección del programa, invoque como ya hemos indicado al método *`get()`*.

```
    // Normalizamos el conjunto de imágenes extraídas y mostramos por pantalla una de las imágenes normalizadas
    lego_image_segmentation.normalize_size_of_images_extracted(100);
    vector<Mat> images_extracted_normalized = lego_image_segmentation.get();
    namedWindow("Objeto extraído normalizado", WINDOW_NORMAL | CV_WINDOW_KEEPRATIO);
    imshow("Objeto extraído normalizado", images_extracted_normalized[5]);
    waitKey(0);
```

Figura 45 Captura de código que realiza el proceso de normalización, obtiene el conjunto de imágenes normalizadas y muestra por pantalla una de las imágenes normalizadas

Puede consultar:

- El número de imágenes extraídas en el proceso de detección y segmentación por medio del método de tipo entero`number_of_images_extracted()`.
- Si el conjunto de imágenes detectadas y segmentadas ha sido normalizado. Para ello invoque al método de tipo booleano *`is_normalized()`*.

```
cout << "¿El conjunto de imágenes extraídas está normalizado? " << lego_image_segmentation.is_normalized();

cout << "¿Cúantas piezas han sido extraídas?" << lego_image_segmentation.number_of_images_extracted();
```

Figura 46Captura de código que indica si el conjunto de imágenes extraídas está normalizado, así como el número de objetos extraídos

Finalmente, puede guardar en disco el conjunto de imágenes extraídas (normalizadas o no) a través del método *`save()`*. Recibe como parámetros de entrada dos cadenas de texto pasadas por referencia: la primera indica la ruta de la carpeta en la que desea guardar dicho conjunto de imágenes y la segunda el nombre de la categoría a la que pertenecen, y que servirá para generar los nombres de estas.

```
    // Guardamos el conjunto de imágenes extraídas y normalizadas correspondiente a la categoría conector corto
    lego_image_segmentation.save("p:/example/normalized/", "conector corto");
}
```

Figura 47Captura de código en el que guardamos el conjunto de imágenes extraídas y normalizadas de la categoría conector corto en el directorio p;/example/normalized.

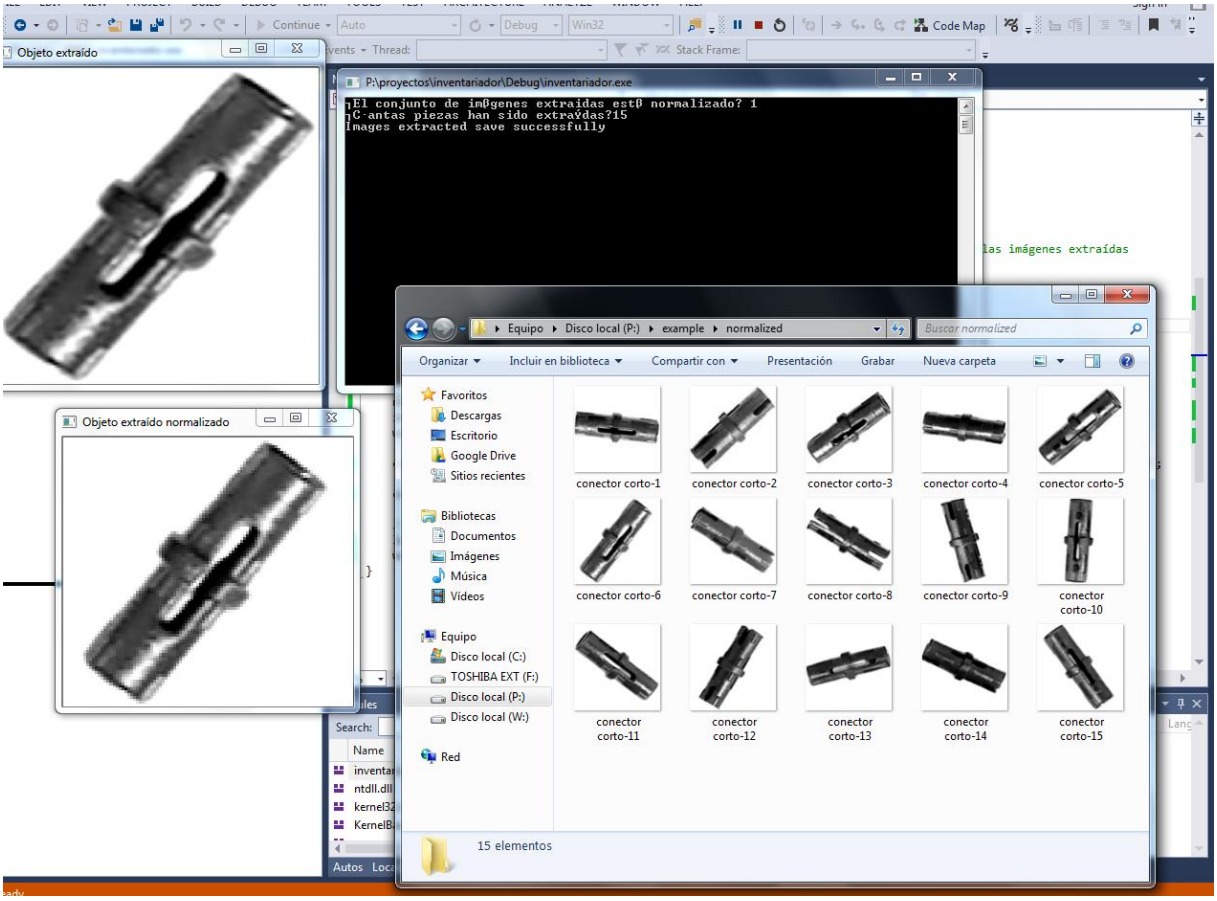


Figura 48 Captura del resultado generado al ejecutar el código indicado

18.3 Clasificación y conteo

Al igual que hacía en el subapartado anterior, debe incluir en el fichero cabecera de la clase del proyecto en la que desea realizar el proceso de clasificación y conteo, o en la clase donde se encuentra el programa principal:

- La clase *Classifier*.
- Las clases de los descriptores de imágenes que vaya a emplear:
  - *AKAZEFeatureExtractor*(AKAZE),
  - *DenseSIFTFeatureExtractor* (Dense – SIFT),
  - *DenseSURFFeatureExtractor*(Dense – SURF),
  - *HUMomentsFeatureExtractor* (Momentos de HU),
  - *KAZEFeatureExtractor* (KAZE), y
  - *SURFFeatureExtractor* (SURF).

Así mismo, antes de llevar a cabo dicho proceso, es necesario garantizar que losconjuntos de imágenes de entrenamiento:

- Están normalizados. Tamaño de imagenrecomendado 150*150 píxeles.
- Se encuentran en un directorio con la siguiente estructura: tantos subdirectorios como categorías vayamos a procesar, cada uno con el nombre de su categoría asociada. El nombre del subdirectorio es utilizado como etiqueta de la categoría. Las etiquetas "*Wrong*" y "*Unknown*" están reservadas por el sistema.

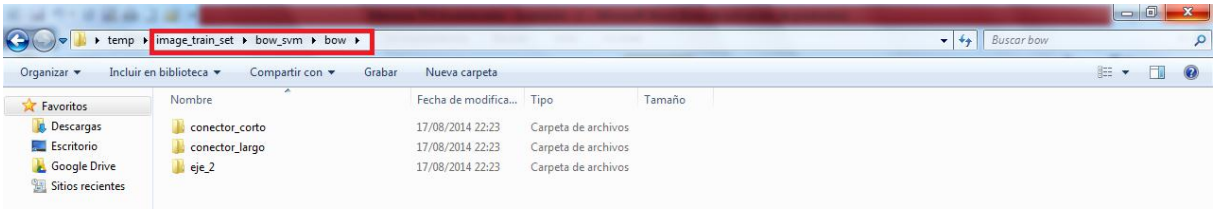


Figura 49 Ejemplo de estructura de directorio

A continuación indicaremos cómo configurar y usar la clase *Classifier* para llevar a cabo el proceso anteriormente mencionado.

18.3.1 Configuración de la clase Classifier

Una vez dispone del conjunto de datos de entrenamiento estructurado conforme a lo visto al comienzo de este subapartado, así como de un objeto de la clase *Classifier* correctamente instanciado, procederemos a establecer su configuración a través de los métodos siguientes:

- *set_classification_mode()*. Como parámetro de entrada recibe un enumerado que indica el modo de clasificación (BOW-SVM, Naive-SVM).Dicho enumerado es accesible por el espacio de nombre *inventariador::classification_mode*.
- *set_FeatureExtractor()*. Recibe como parámetro de entrada un puntero a un objeto de la clase FeatureExtractor. Nos permite establecer el descriptor de imagen a utilizar, ya que como se vio en el subapartado 12.1.2 Módulo clasificación y conteo, todas las clases de los descriptores de imágenes heredan de la clase *FeatureExtractor*. Para obtener una instancia de dichos descriptores se puede invocar al método estático *get_new_instance()* de la clase *FeatureExtractorFactory*. Dicho método recibe como parámetro de entrada una cadena de texto conteniendo el nombre del descriptor del que queremos obtener su instancia, y retorna un puntero de tipo FeatureExtractor que apunta a la nueva instancia creada.
- *set_image_train_folder_path_of_bow_vocabulary()*. Recibe como parámetro de entrada una cadena de texto con la ruta al directorio del conjunto de imágenes de entrenamiento para construir el vocabulario del modo de clasificación BOW-SVM.
- *set_cluster_size_of_bow_vocabulary()*. Recibe como parámetro de entrada un entero que indica el tamaño del clúster del vocabulario para el modo de clasificación BOW-SVM.

- `set_image_train_folder_path_of_svm_set()`. Recibe como parámetro de entrada una cadena de texto con la ruta al directorio en donde se encuentra el conjunto de imágenes de entrenamiento para el conjunto de SVM.
- *`set_svm_parameters()`*. Recibe como parámetro un puntero a un objeto de la estructura*`CvSVMParams`*[29] de OpenCV. Esta nos permite en nuestro caso configurar los parámetros del SVM relativos al tipo de núcleo y criterios de terminación. En caso de no proporcionar dicho parámetro, se utiliza la configuración por defecto de dicha estructura, a excepción del tipo de núcleo que lo establecemos en Lineal.

Se muestra en las figura 50 y 51 un ejemplo de configuración para los modos de clasificación BOW-SVM y Naive-SVM.

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include "FeatureExtractorFactory.h"
#include "AKAZEFeatureExtractor.h"
#include "Classifier.h"

    Ptr<Classifier> classifier = new Classifier();
    classifier->set_classification_mode(training_mode::BOW_SVM);
    classifier->set_image_train_folder_path_of_svm_set("p:/image_train_set/bow_svm/svm");
    Ptr<CvSVMParams> svm_params = new CvSVMParams();
    svm_params->kernel_type = CvSVM::RBF;
    classifier->set_svm_parameters(svm_params);
    classifier->set_FeatureExtractor(FeatureExtractorFactory::get_new_instance("KAZEFeatureExtractor"));
    classifier->set_cluster_size_of_bow_vocabulary(500);
    classifier->set_image_train_folder_path_of_bow_vocabulary("p:/image_train_set/bow_svm/bow");
```

Figura 50 Configuración ejemplo del clasificador para el modo de clasificación BOW-SVM

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include "FeatureExtractorFactory.h"
#include "AKAZEFeatureExtractor.h"
#include "Classifier.h"

    Ptr<Classifier> classifier = new Classifier();
    classifier->set_classification_mode(training_mode::NAIVE_SVM);
    classifier->set_image_train_folder_path_of_svm_set("p:/image_train_set/naive_svm");
    Ptr<CvSVMParams> svm_params = new CvSVMParams();
    svm_params->kernel_type = CvSVM::RBF;
    classifier->set_svm_parameters(svm_params);
    classifier->set_FeatureExtractor(FeatureExtractorFactory::get_new_instance("AKAZEFeatureExtractor"));
```

Figura 51 Configuración ejemplo del clasificador para el modo de clasificación Naive-SVM

18.3.2 Uso de la clase Classifier

Antes de llevar a cabo el proceso de clasificación y conteo, es necesario cargar y almacenar en un vector de objetos de la clase Mat las imágenes de las piezas Lego que serán procesadas.

De esta forma y una vez configurado correctamente nuestro objeto de la clase *Classifier*, para realizar el proceso de clasificación y conteo solo tiene que invocar al método *classify()*. Este recibe como parámetro de entrada pasado por referencia el vector de objetos de la clase Mat ya mencionado.

Pudiendo una vez finalizado dicho proceso realizar las siguientes acciones:

- Obtener el resultado de la clasificación:`get_classification_list()`. Retorna un puntero a un objeto de tipo *map* conteniendo el resultado del proceso de clasificación y conteo. La clave corresponde con la categoría detectada y el valor con el número de objetos identificados en dicha categoría.
- Guardar en disco el resultado de la clasificación: *save_classification_list()*. Recibe como parámetro de entrada una referencia a un objeto de la clase de OpenCV *FileStorage*[30] al hacer uso del modelo de persistencia de dicha librería.
- Obtener los datos estadísticos de sesión (tiempo de entrenamiento y tiempo de clasificación):*get_statistics_time()*. Retorna un objeto de la estructura *ClassifierStatisticsTime*. Esta contiene dos campos de tipo int64: *training_time* (tiempo de entrenamiento) y *classification_time* (tiempo de clasificación).
- Guardar en disco los datos estadísticos de la sesión (tiempo de entrenamiento y tiempo de clasificación): *save_statistics_time()*. Recibe como parámetro de entrada una referencia a un objeto de la clase de OpenCV FileStorage al hacer uso del modelo de persistencia de dicha librería.
- Guardar en disco los datos de la sesión (datos de configuración y de entrenamiento): `save_session_data()`. Recibe como parámetro de entrada una cadena de texto con la ruta al archivo XML que contendrá dichos datos.
- Cargar los datos de sesión: `load_session_data()`. Recibe como parámetro de entrada la cadena de texto conteniendo la ruta al archivo XML de sesión de datos.

```
string save_path = "p:/test/results/naive-akaze.xml";
FileStorage fs = FileStorage(save_path, FileStorage::WRITE);
if (fs.isOpened()) {
    classifier->save_statistics_time(fs);
    classifier->save_classification_list(fs);
    fs.release();
}
```

Figura 52 Extracto de código para guardar tanto los datos de sesión como los datos estadísticos

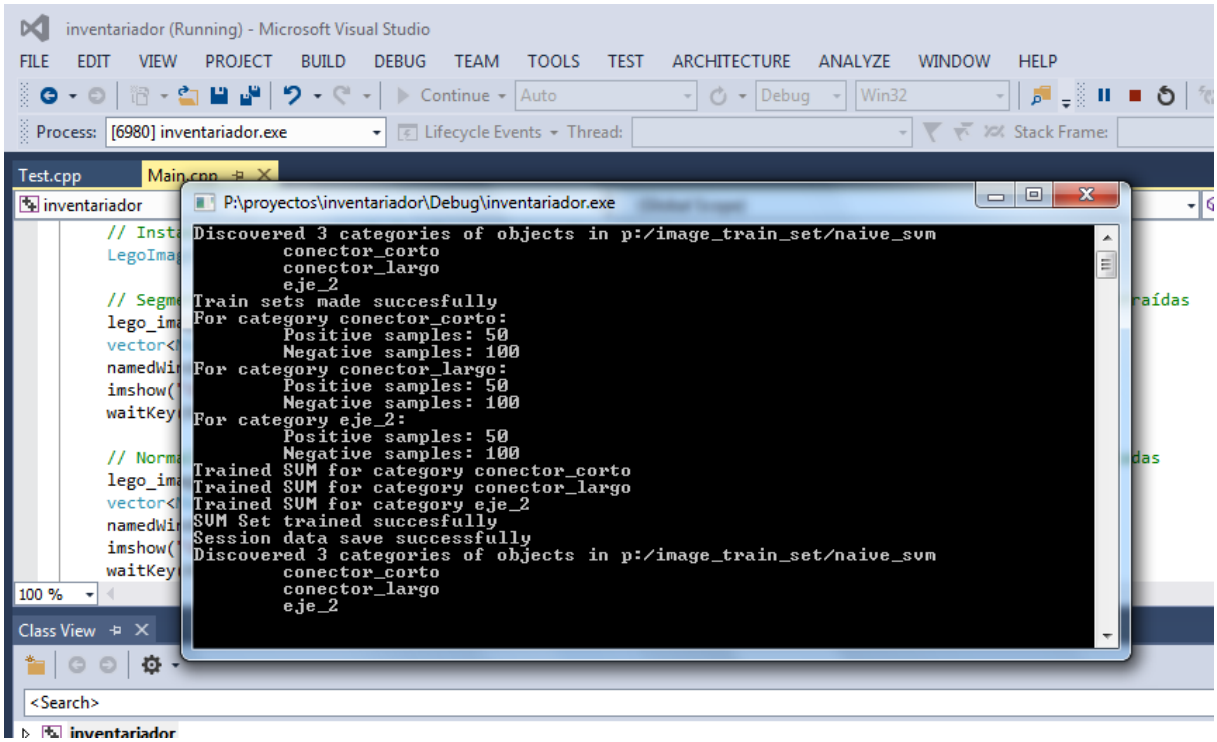


Figura 53 Captura de la información mostrada por pantalla durante un proceso típico de clasificación y conteo, en el que se guarda los datos de sesión del clasificador.