Máster Universitario en

Ingeniería **C**omputacional y **S**istemas **I**nteligentes

Universidad del País Vasco Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifiziala Saila –
Departamento de Ciencias de la Computación e Inteligencia Artificial

K I S A I C S I

Tesis de Máster

Detección automática de presencia/ausencia de atún en imágenes obtenidas mediante sonar de largo alcance a bordo de buques pesqueros y aplicación de *Optical Character Recognition* (OCR) para la extracción de parámetros

Jon Uranga Aizpurua

Tutor(a/es)

Clemente Rodríguez (UPV/EHU)
Departamento de Arquitectura y Tecnología de Computadores
Facultad de Informática

Haritz Arrizabalaga (AZTI Tecnalia)
Unidad de Investigación Marina. Gestión recursos Túnidos

Carmen Hernández (UPV/EHU)
Departamento de Ciencia de la Computación e Inteligencia Artificial
Facultad de Informática

informatika fakultatea facultad de informática

KZAA /CCIA

Septiembre 2013

Agradecimientos

Agradezco sinceramente a Clemente Rodríguez y Carmen Hernández, mis tutores en la UPV-EHU y a Haritz Arrizabalaga y Guillermo Boyra, mis tutores en AZTI-Tecnalia, por sus consejos, su preocupación, su interés y su completa disposición para ayudarme con la supervisión del trabajo realizado. También quiero agradecer a Ari Urkullu los consejos que me ofreció al comienzo de este proyecto.

El proyecto se está llevando a cabo gracias a la beca predoctoral concedida conjuntamente por el Centro Tecnológico AZTI y por el departamento de Medio Ambiente, Planificación Territorial, Agricultura y Pesca. Dicha beca fue concedida el 2 de diciembre de 2011, donde se dictó la resolución la beca de formación de jóvenes investigadores en el sector agropesquero vasco, correspondiente al ejercicio del año 2011. El título del proyecto de la beca de 4 años es el siguiente: *Análisis de imagen y clasificación supervisada de señales de atún obtenidas mediante sonar de largo alcance.*

Resumen

Abstract

In this master thesis a methodology for automated analysis of long-range sonar signals and an Optical Character Recognition (OCR) application are presented. The first contribution involves analyzing sonar screenshots by image processing techniques. In this process, for each sonar image we extract and analyze measurable regions, obtaining a descripting set of characteristics for each region. Through the help of experts, each region is identified into a class (tuna or not-tuna) and a database is created by supervised learning. Thus, a classification model is performed. The second contribution, recognizes and extracts from the sonar image screenshots, the alphanumeric characters referred to the situation parameters (speed, bearing, GPS localization) and sonar setup (gains, tilt, beam-width). The final aim of this process is to automatically detect tuna in the sonar screenshots, as a first step towards the developtment of an index of abundance of this species, based on the automatic processing of the sonar images of the commercial fleet recorded during its rutinary fishing activity.

Resumen

En esta tesis de máster se presenta una metodología para el análisis automatizado de las señales del sonar de largo alcance y una aplicación basada en la técnica de reconocimiento óptico de caracteres (*Optical Character Recognition,* OCR). La primera contribución consiste en el análisis de imágenes de sonar mediante técnicas de procesamiento de imágenes. En este proceso, para cada imagen de sonar se extraen y se analizan las regiones medibles, obteniendo para cada región un conjunto de características. Con la ayuda de los expertos, cada región es identificada en una clase (atún o no-atún). De este modo, mediante el aprendizaje supervisado se genera la base de datos y, a su vez, se obtiene un modelo de clasificación. La segunda contribución es una aplicación OCR que reconoce y extrae de las capturas de pantalla de imágenes de sonar, los caracteres alfanuméricos correspondientes a los parámetros de situación (velocidad, rumbo, localización GPS) y la configuración de sonar (ganancias, inclinación, ancho del haz). El objetivo final de este proceso es el de maximizar la eficiencia en la detección de atún en el Golfo de Vizcaya y dar el primer paso hacia el desarrollo de un índice de abundancia de esta especie, el cual esté basado en el procesamiento automático de las imágenes de sonar grabadas a bordo de la flota pesquera durante su actividad pesquera rutinaria.

Laburpena

Master tesi honetan irismen luzeko sonarraren seinalea automatikoki analizatzeko metodologia bat eta *Optical Character Recognition* (OCR) teknikan oinarrituriko aplikazio bat aurkezten dira. Lehenengo ekarpenak sonar irudiak analizatzen ditu, irudi prozesaketa tekniketan oinarriturik. Prozesu honetan, sonar irudi bakoitzetik erregio neurgarriak erauzten eta analizatzen dira, non erregio bakoitzari ezaugarri multzo bat esleitzen zaion. Adituen laguntzari esker, erregio bakoitza klase batean identifikatzen da (atuna edo ez-atuna). Gainbegiraturiko ikasketa baten bitartez datu basea sortuko da eta modu honetan sailkapen modelo bat lortuko dugu. Bigarren ekarpena, sonarretik grabaturiko irudietatik egoera (abiadura, noranzkoa, GSP kokapena) eta konfigurazio parametroak(ganantziak, inklinazioa, uhin-sorta zabalera) bereizten eta erauzten dituen OCR aplikazio bat da. Prozesu honen helburu nagusia Bizkaiko Golkoan atunaren detekzioaren eraginkortasuna maximizatzea eta espezie honen ugaritasun indize bat garatzeko bidean lehen pausu bat ematea da, indize hau euskal arrantza flotako arrantza-ontzietan grabaturiko sonar irudien prozesaketa automatikoan oinarriturik egongo litzatekeelarik.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Objetivos de la tesis de máster

El objetivo de la tesis de máster es, por un lado, validar el preprocesado de imágenes semi-automático (PISA) de sonar de largo alcance propuesto por [29] mediante un estudio comparativo entre diferentes algoritmos de aprendizaje utilizados para realizar una clasificación supervisada. El PISA realiza una serie de preprocesos sobre las imágenes registradas por el sonar, las segmenta en diferentes regiones, extrae sus características, las etiqueta mediante ayuda del juicio de expertos y crea una base de datos para realizar un aprendizaje supervisado. Se debe resaltar que el área dentro de la imagen del sonar que se ha procesado mediante el PISA corresponde al área del ecograma, por lo que todas las características extraídas de las regiones hacen referencia al ecograma (figura 1.1).

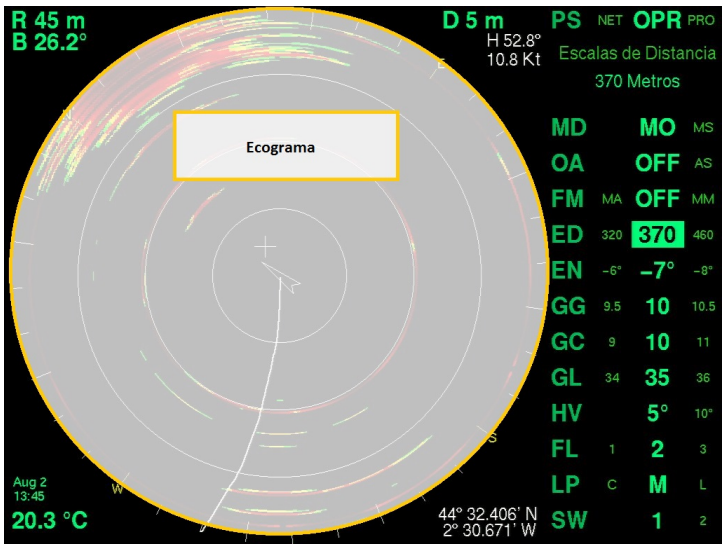


Fig. 1.1: Área del ecograma.

En nuestro caso, después de la fase de preprocesado obtenemos la base de datos de objetos ya etiquetados como "atún" o "sin atún". De este modo disponemos de un modelo de clasificación binaria de los objetos extraídos del área del ecograma de las imágenes de sonar que disponemos.

Por el otro lado, como segundo objetivo se quiere desarrollar una técnica de OCR (*Optical Character Recognition*) para el reconocimiento y extracción de los caracteres alfanuméricos, referidos a parámetros de situación y configuración del sonar, de las imágenes de sonar provenientes de los eventos de pesca a bordo de buques pesqueros en las campañas de pesca del atún en aguas del Golfo de Vizcaya. De cara al futuro, se espera poder mejorar la clasificación supervisada empleando estos nuevos parámetros.

En definitiva, el deseo es el de seguir avanzando en la investigación de procesamiento automático de imágenes de sonar de largo alcance. Esto es posible gracias al proceso formativo ofrecido por la beca doctoral que se disfruta en el centro tecnológico AZTI.

1.2. Objetivos últimos de la investigación

Actualmente no existe una evaluación directa de la abundancia del atún rojo en el Golfo de Vizcaya. Debido a que la superficie de distribución y la movilidad de esta especie son muy grandes, las evaluaciones de la abundancia de hoy en día se basan en la captura por unidad de esfuerzo (CPUE) la cual tiene algunos inconvenientes: ausencia de levantamientos/muestreos científicos, las observaciones pueden estar correlacionadas y existe una gran variabilidad en la capturabilidad del arte de pesca.

En el Golfo de Vizcaya, la pesquería de atún ha sido y sigue siendo importante. La técnica utilizada es la pesca de cebo vivo y esta técnica depende, en gran medida, del hambre del atún y de su disposición a comer, lo cual introduce mucha variabilidad en el análisis de la CPUE como índice de abundancia. Por estas razones, existe la necesidad de desarrollar un índice de abundancia independiente de la pesca y del hambre del atún.

Por otro lado, se dice que la flota vasca de pesca de cebo vivo tiene "ojos". Históricamente se ha utilizado el sonar de largo alcance (LRS) para buscar el atún, ya que rara vez son visibles en la sonda vertical. Pero los sonares utilizados por la flota vasca son sonares analógicos que no tienen una salida digital para los parámetros que se muestran en pantalla. Debido a ello, se ha propuesto estudiar las capturas de pantalla del sonar durante la campaña de pesca del atún y tratar de diseñar un método automático para el análisis de estas imágenes.

Con este enfoque, se aborda el tratamiento de las imágenes de atún procedentes del sonar de los buques pesqueros. La primera contribución ha sido el desarrollo de un procedimiento de análisis de imágenes semi-automatizado. A continuación, se ha construido un método basado en la minería de datos para la validación de este proceso y finalmente se ha desarrollado una técnica de OCR para extraer nuevos parámetros que se espera que sean claves para mejorar la detección del atún en las imágenes estudiadas.

El objetivo final es detectar atún en nuestras series de imágenes y encontrar una nueva metodología para estimar el atún con imágenes tomadas por el LRS y evaluar la precisión de las estimaciones.

1.3. Marco del estudio

El proyecto que se presenta sigue los objetivos marcados por instituciones y centros tecnológicos como el ICCAT (International Commission for the Conservation of Atlantic Tunas), el Departamento de Medio Ambiente, Planificación Territorial, Agricultura y Pesca del Gobierno Vasco y el centro tecnológico AZTI Tecnalia. Dichos objetivos buscan estudiar, gestionar y desarrollar planes de recuperación para diferentes especies. En nuestro proyecto nos centramos en el atún del cantábrico: el atún rojo (*Tunnus thynnus*) y el bonito (*Thunnus alalunga*). Estas especies entran al Golfo de Vizcaya en migración trófica (para alimentarse) durante los meses de junio y julio cuando la temperatura del agua es de 17ºC aproximadamente.

Fig. 1.2: El atún rojo (*Tunnus thynnus*)(a) y el bonito (*Thunnus alalunga*)(b).

Según el ICCAT, existía un grave riesgo de colapso del stock y de las pesquerías de atún rojo. Por tanto, el futuro de las pesquerías tradicionales del atún rojo del Golfo de Vizcaya se ve amenazado por el crítico estado de las poblaciones debido al incremento de la actividad de otras pesquerías industriales en los últimos años. La UE, siguiendo las recomendaciones del ICCAT, ha adoptado un plan de recuperación del atún rojo, que garantiza su recuperación y explotación sostenible a largo plazo. Este plan establece TACs (Total Admisible de Capturas), tallas mínimas, protocolos para la instalación de *logbooks* electrónicos y notificación instantánea de capturas, programa de observadores, etc.

El área de estudio se define gracias a las anotaciones tomadas a bordo los buques pesqueros y la información de los *logbooks* electrónicos de las campañas de pesca a cebo vivo del atún en el Golfo de Vizcaya [25]. Esta pesquería trabaja en el Golfo de Vizcaya en los meses de verano, de junio a octubre y se localiza en la parte sureste del golfo, entre las coordenadas 43–47°N, 2–6°W, cerca de la costa (Figura 1.3). Al atún se explota mediante una flota pesquera especializada. Históricamente la mayor parte de la flota ha trabajado desde el puerto de Hondarribia, pero a partir del año 1996 el puerto de Getaria empezó a tener más protagonismo y, actualmente, el 80 % de la pesca del golfo de Vizcaya se realiza mediante la flota especializada de estos dos puertos [25].

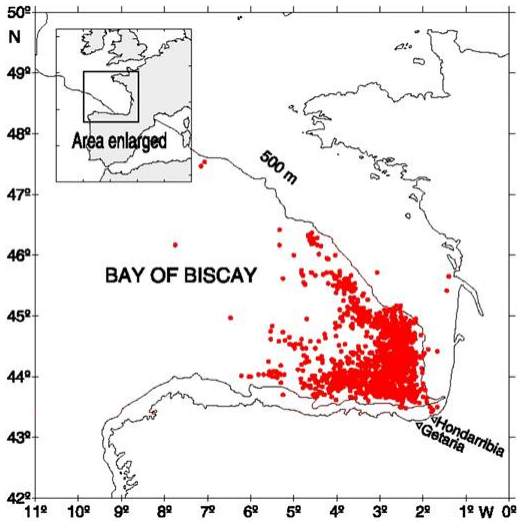


Fig. 1.3: Área de estudio y las localizaciones de los eventos de pesca.

Respecto a la técnica de pesca empleada, en el Golfo de Vizcaya los túnidos son explotados desde tiempo inmemorial por la flota vasca, siendo pioneros en la introducción de la técnica del cebo vivo dirigida al bonito y al atún rojo en Europa a finales de los años 40. El arte del cebo vivo es el que se realiza utilizando cañas sobre cardúmenes que son atraídos y mantenidos próximos a la embarcación, arrojando agua y, periódicamente, peces vivos. El cebo o carnada, que generalmente suele ser chicharro pequeño, y alternativamente, verdel pequeño, sardina o anchoa, se captura con una red de cerco de pequeñas dimensiones y se mantiene vivo a bordo, en los viveros. La captura depende del "hambre" de los atunes y, en el caso del cebo vivo, de la capacidad de pescar el cebo.

1.4. Equipo de sonar y el sistema de adquisición de las imágenes

El sonar activo empleado para el estudio transmite una señal acústica y detecta la respuesta de los objetos en el agua circundante. Se trata de un dispositivo omni-haz que lo utiliza la mayor parte de la flota vasca. El transductor se puede inclinar verticalmente según las condiciones del mar y detecta los cardúmenes y bancos de peces cerca de la superficie en cualquier dirección alrededor de la nave. La pantalla puede ser un ecograma o, como en nuestro caso, una vista en planta presentada en una pantalla de monitor con los objetivos que se muestran tanto en rango y dirección con respecto a la posición del buque en el centro de la pantalla [16]. La mayor parte de la flota vasca utiliza el modelo de 90 kHz. Esta es la frecuencia óptima para garantizar que el haz del transductor sufra la mínima flexión debido a las burbujas de aire de la superficie. El sonar ofrece la opción de elegir entre tres anchuras verticales de haz (5^o,10^o y 20^o). Este parámetro se suele variar según el estado del mar, pero en general, para eventos de búsqueda del atún se suele emplear la anchura mas fina debido a que ofrece una buena respuesta y el rango es mayor. El uso de una mayor anchura ofrece una mejor respuesta cerca del barco.

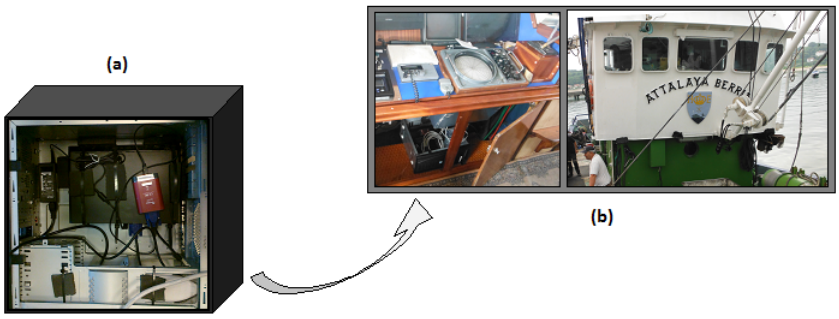


Fig. 1.4: Dispositivo de adquisición de imagen (a) y su instalación en el barco (b).

La salida de pantalla de los sonares de 90 kHz se ha adquirido instalando un dispositivo de adquisición de imagen compuesta por: un video-duplicador de 400MHz, para duplicar la señal de imagen; un dispositivo de captura de VGA externo, para capturar la imagen de pantalla; y un ordenador portátil pre-programado conectado a la red eléctrica, para para poder adquirir imágenes continuamente sin necesidad de ninguna acción externa.

Las imágenes de sonar capturadas están divididas en dos secciones básicas, el panel del menú y la pantalla del sonar. El panel de menú proporciona al usuario la opción de elegir entre distintos *displays* y de configurar parámetros del sonar como la inclinación, la anchura del haz y las ganancias (ganancias generales, cercanas y lejanas). La pantalla del sonar, en cambio, proporciona la respuesta del haz en forma de ecograma. Durante los eventos reales de pesca las imágenes capturadas se clasifican básicamente en dos tipos de imágenes: imágenes con presencia de atún y imágenes con ausencia de atún, por lo que estamos ante un problema de clasificación binaria. Esta clasificación se realiza gracias al juicio de expertos, la información de los *logbooks* y la información registrada por el observador científico. Para los casos de presencia, se pueden tener imágenes con mayor o menor presencia de atún y, para las imágenes de ausencia, se tiene la siguiente casuística: ruido de superficie, ruido causado por el buque pesquero e imágenes vacías. Las imágenes seleccionadas para el estudio que se presenta intentan cubrir todos los casos posibles (ver figura 1.5).

El color del fondo de la pantalla del sonar y la intensidad de los parámetros del panel del menú pueden variar según las necesidades y preferencias del capitán, pero tanto la respuesta del ecograma como la localización de los parámetros dentro de las imágenes es fija lo cual es muy importante para poder ejecutar con garantías el proceso del OCR.

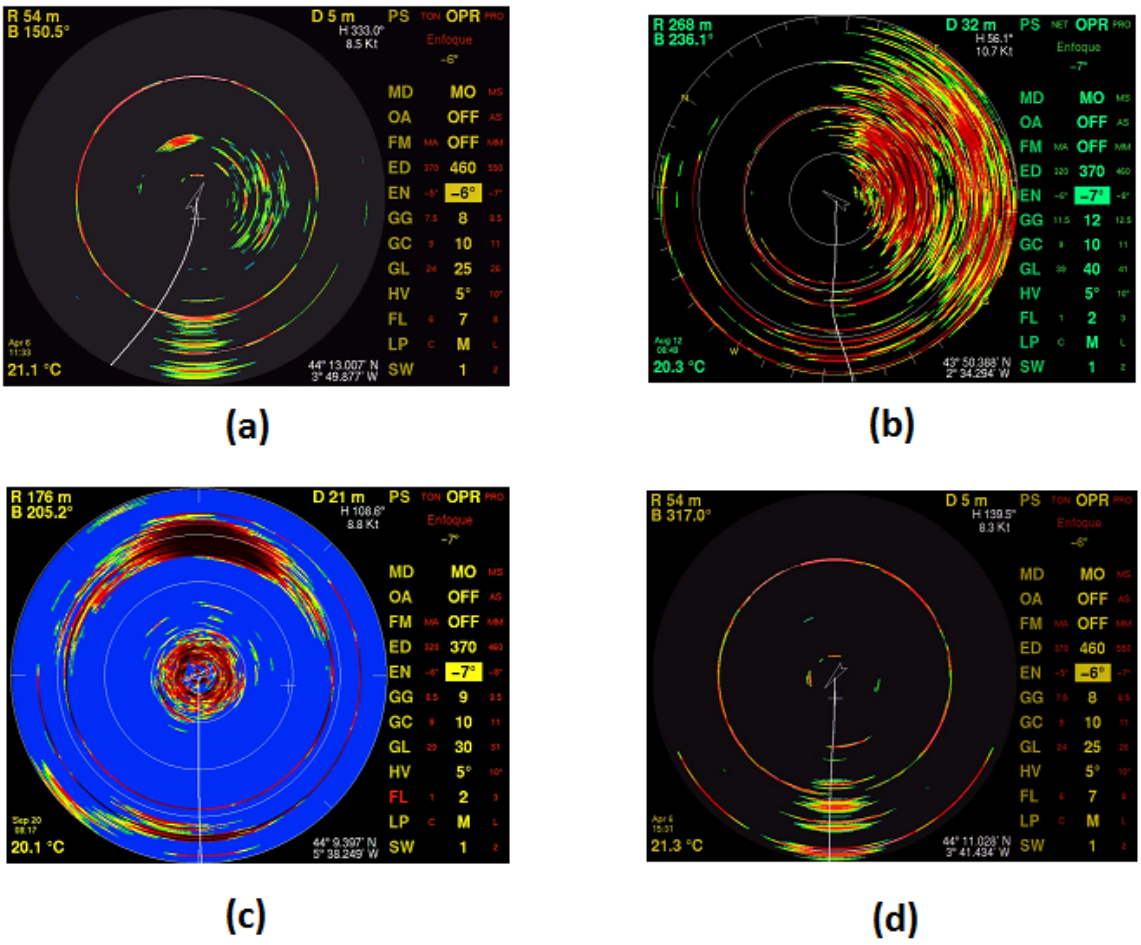


Fig. 1.5: Imagen con fondo negro de poco atún (a), imagen con fondo negro de ruido de superficie (b), imagen con fondo azul de mucho atún (c) y imagen con fondo negro de estela en la parte trasera y de vacío en la parte frontal (d).

Capítulo 2

Clasificación supervisada

2.1. Descripción de la base de datos

Para la construcción de la base de datos se ha aplicado el preprocesado de imágenes semi-automático PISA propuesto por [29]. Mediante dicha tarea se realiza un preprocesado inicial de las imágenes, una segmentación y una extracción de características de las regiones obtenidas en la segmentación. Mediante dicha tarea de extracción se obtienen las características morfológicas (20 características) de las regiones, relativas a las imágenes de sonar obtenidas durante los eventos de pesca del atún. A continuación, se citan las características extraídas:

- Se calcula su área, descrita por el atributo "(1) Área".
- Se calcula su centroide, es decir, promedia las posiciones de los píxeles que componen el objeto, expresando después su centroide según sus dos componentes espaciales, quedando descrito por los atributos "(2) X" e "(3) Y".
- "(4) Perimeter". Se calcula el rectángulo de menor tamaño que contiene al objeto, quedando éste descrito por su anchura, por su altura y por la posición de su esquina superior izquierda, descrita por sus dos componentes espaciales. Esta descripción del rectángulo mínimo se recoge en las variables "(7) Width", "(8) Height", "(5) BX" y "(6) BY".
- Se calcula la elipse que mejor se ajusta al objeto. Esta elipse queda descrita por la longitud del eje de mayor elongación de la elipse, por la longitud del eje de menor elongación de la elipse y por el ángulo que forma el eje de mayor elongación de la elipse con el eje de las abscisas. Estos tres descriptores de la elipse se recogen en los atributos "(9) Major", "(10) Minor" y "(11) Angle" respectivamente.
- Se calcula el cociente entre la elongación del eje mayor (numerador) y el eje menor (denominador) recogido en el atributo "(18) AR" *aspect ratio*.
- Se calcula (12) "circularidad" mediante la siguiente formula: $circularidad = \frac{4\pi.area}{perimetro^2}$ Cuanto más próximo esté el valor de la circularidad a uno, más próxima a un círculo será su forma, mientras que valores próximos a cero sugieren una forma alargada.
- Se calcula la (19) "redondez", propiedad definida por la siguiente fórmula: $redondez = \frac{4.area}{\pi.ejemayor^2}$

- Se calcula la solidez, se recoge en el atributo "(20) Solidity". Se trata del cociente entre el área del objeto (numerador) y el área convexa del objeto (denominador).
- Se calcula la mayor distancia existente entre cualquier par de píxeles del objeto, que se recoge en el atributo "(13) Feret".
- En el atributo "(16) Feret Angle" encontramos el ángulo de la línea que describe la distancia de "Feret".
- En "(14) FeretX" y "(15) FeretY" se recoge la ubicación de uno de los dos extremos de la línea que describe la distancia "Feret".
- Se calcula el valor "(17) Min Feret" que es la longitud de una línea perpendicular y que intersecta por el centro a la línea que describe la distancia de "Feret".

Se puede observar como todas las características extraídas hacen referencia a la morfología de las regiones extraídas. Este hecho se tiene que tener en cuenta a la hora de sacar conclusiones y de avanzar en la búsqueda del método para la detección de presencia/ausencia del atún en las imágenes del sonar.

El número de series e imágenes a utilizar en el PISA ha sido establecido con el objetivo de construir una base de datos relativamente balanceada. Las imágenes seleccionadas han sido tomadas a bordo de dos buques pesqueros de la flota vasca de pesca a cebo vivo del atún, durante las campañas de pesca de verano del año 2009 y 2011. Han sido seleccionados un total de seis series representativas: 3 series de agosto del 2009 y otras tres series de agosto del 2011. Por lo que la estructura inicial de la base de datos se muestra en la tabla 2.1[1].

	PRESENCE IMAGES	ABSENCE IMAGES
Gure Gogoa 2009	428	429
Gure Gogoa 2009	273	273
Gogoa 2009	277	277
Attalaya berria 2011	240	240
Attalaya berria 2011	116	116
Attalaya berria 2011	63	63
Total	1397	1398

Tabla 2.1: Número de imágenes empleadas para la construcción de la base de datos.

De este modo, antes del PISA se tienen 1397 casos de presencia y 1398 casos de ausencia. Después de realizar el preprocesado, se obtienen 22411 regiones que se van a utilizar para construir la base de datos binaria (presencia/ausencia). Del total de los casos, 1407 casos son positivos (presencia) y 21004 casos negativos (ausencia). Estos resultados se pueden observar en la tabla 2.2. El ratio obtenido entre los casos positivos y negativos es de 1/14.03, el cual refleja que estamos trabajando con una base de datos ligeramente desequilibrada.

[1] Nótese que los campos de la tabla, "PRESENCE IMAGES" y "ABSENCE IMAGES", corresponden al número de imágenes con presencia y ausencia de atún respectivamente.

	PRESENCE IMAGES	ABSENCE IMAGES
Gure Gogoa 2009	358	4108
Gure Gogoa 2009	279	6581
Gogoa 2009	180	4935
Attalaya berria 2011	516	3409
Attalaya berria 2011	113	1076
Attalaya berria 2011	61	895
Total	1407	21004

Tabla 2.2: Número de final de regiones empleadas en la base de datos después de ejecutar el PISA.

El uso de bases de datos no-balanceadas suelen causar problemas en estudios de clasificación. Para evitar dichos problemas existen métodos para sub-muestrear y sobre-muestrear los casos de estudio. Con este objetivo se ha aplicado la técnica de SMOTE (*Synthetic Minority Oversampling Technique* [7])para el sobre-muestreo y para el sub-muestreo se ha aplicado el filtro *Spread Sample filter* [30]. La técnica de SMOTE sobre muestrea los casos que son minoría, en este caso los casos de presencia. En cambio el filtro *Spread Sample* sub-muestrea los casos que son mayoría, en este caso los casos de ausencia. Cada uno de los métodos tienen sus debilidades y fortalezas. En la tabla 2.3 se muestran los ratios obtenidos para las nuevas bases de datos.

	Tuna	No Tuna	Ratio
TOTAL	1497	21004	14.03
SMOTE	2994	21004	7.02
SPREAD	1497	10502	7.01

Tabla 2.3: Ratios obtenidos para las nuevas bases de datos.

De este modo, finalmente obtenemos tres bases de datos de 22501 (TOTAL), 23998 (SMOTE) y 11999 (SPREAD) instancias, con 20 características morfológicas para cada una.

2.2. Clasificación supervisada

En este apartado se presentan los algoritmos de clasificación supervisada utilizados para la clasificación de nuestras bases de datos. Los algoritmos de clasificación aplicados empleando el software Weka [13] son los siguientes:

1. ***Random forest* (RF)** [5], es un multi-clasificador que consiste en una combinación o mezcla de árboles de decisión no-podados con una selección aleatoria de las variables en cada división de cada árbol. El resultado de la clasificación es la clase mayoritaria de los resultados de los arboles que pertenecen al multiclasificador. Este algoritmo mejora la exactitud de la clasificación por la construcción estocástica de cada árbol de clasificación individual;

2. ***Multilayer perceptron* (MLP)** es una de red neuronal compuesta por múltiples capas de nodos totalmente conectados el uno al otro y que utiliza funciones discriminantes capaces de clasificar la entrada de acuerdo a la función discriminante máxima [3, 15]. MLP utiliza una técnica de aprendizaje supervisada denominado *backpropagation*;

3. **IBk (IBK),** se trata de un algoritmo basado en el vecino más cercano (Knn), propuesto originalmente por *Fix y Hodges* [11]. Este algoritmo pertenece a la familia de clasificadores "*lazy*" en Weka. Es un algoritmo basado en instancias, de aprendizaje no paramétrico, que utiliza una distancia métrica para encontrar el patrón de aprendizaje más cercano al patrón de clasificación. Cuando se trabaja con datos numéricos se utilizan las distancias euclídea o de *Manhattan*. Para los datos nominales, se usa la distancia de *Hamming*. Cuando se utilizan k vecinos, la clase desconocida del patrón se obtiene a partir de la mayoría de las clases de los k vecinos más cercanos. Este método es muy intuitivo, ya que clasifica los casos basándose en su similitud al conjunto de entrenamiento;

4. **J48,** este algoritmo es una versión optimizada del algoritmo de clasificación mediante árboles de decisión denominado C4.5 [23, 22]. Este algoritmo genera un árbol de decisiones probabilístico que puede ser fácilmente interpretado por expertos y transformado en claras y comprensibles reglas;

5. Los **SVM** son algoritmos de aprendizaje supervisados ampliamente utilizados en problemas de clasificación. El algoritmo del SVM se basa en la teoría del aprendizaje estadístico y fue presentado por *Vladimir N. Vapnik* en 1995 [9, 6]. Estos modelos crean un modelo probabilístico lineal no-binario construyendo un hiperplano o conjunto de hiperplanos óptimos en un espacio de alta dimensión que linealmente separa los patrones de clasificación. Generalmente, estos patrones no se pueden separar linealmente en el espacio original. Por lo tanto, se proyectan en un espacio de dimensión mucho mayor. De esta forma, el producto escalar se calcula fácilmente, en términos de las variables, en el espacio original. Los SVM maximizan el margen alrededor del hiperplano de separación. La función de decisión se especifica mediante un subconjunto de muestras de entrenamiento (llamados vectores de soporte).

Como se ha comentado previamente, en el experimento se han utilizado estos cinco algoritmos de aprendizaje sobre estas tres bases de datos: (i) base de datos completa (BD completa); (ii) base de datos sobre-muestreada SMOTE; (iii) y la base de datos sub-muestreada SPREAD.

Para evaluar la eficiencia y la eficacia de los diferentes métodos de clasificación, se han calculado los valores medios de los siguientes índices: Kappa [8], Sensibilidad (*Sensitivity*) [10], Especificidad (*Specificity*) [14] y AUC (curva ROC) [14].

Estos índices de validación se definen a partir de los valores *True Positive TP* (a), *True Negative TN* (d), *False Positive FP* (b) y *False Negative FN* (c) de la matriz de confusión (ver tabla 2.4). El valor (a) indica el número de instancias del caso positivo (presencia) que se han clasificado correctamente, el valor (b) indica el número de instancias del caso positivo que se han clasificado incorrectamente, el valor (c) indica el número de instancias del caso negativo (ausencia) que se han clasificado correctamente y el valor (d) indica el número de instancias del caso negativo que se han clasificado incorrectamente. Estos valores se obtienen mediante la comparación entre los resultados observados (clasificación supervisada a partir del juicio experto) y los resultados predichos por el modelo de clasificación.

VALIDATION DATA SET		
	Presence	Absence
Presence	a	b
Absence	c	d

Tabla 2.4: Matriz de confusión.

A partir de dichos valores se evalúa la precisión predictiva del modelo de clasificación binaria calculando los índices de validación comentados previamente [1] para la clasificación realizada:

$$Sensitivity = \frac{TP}{(TP + FN)} = \frac{a}{a + c}$$

$$Specificity = \frac{TN}{(FP + TN)} = \frac{d}{b + d}$$

$$Kappa = \frac{\frac{a+d}{n} - \frac{(a+b)(a+c)+(c+d)(d+b)}{n^2}}{1 - \frac{(a+b)(a+c)+(c+d)(d+b)}{n^2}}$$

donde $n = a + b + c + d$.

La curva ROC se define mediante la relación entre los falsos positivos ($\frac{FP}{FP+TN} = \frac{b}{b+d} = 1 - Specifity$) en el eje x y la sensibilidad en el eje y. Este índice muestra el balance entre los verdaderos positivos y falsos positivos. La curva ROC (AUC) es equivalente al estadístico de Mann-Whittney y proporciona la probabilidad de, al tomar aleatoriamente un par de casos: uno positivo y uno negativo, el modelo positivo tome un valor mayor que el negativo. La escala más utilizada para validar este índice es [27]:

AUC > 0.95	Modelo excelente
$0.85 <$ AUC < 0.95	Buen modelo
$0.75 <$ AUC < 0.85	Modelo de calidad aceptable
AUC < 0.75	Modelo de calidad pobre

Tabla 2.5: Escala de validación para el parámetro AUC.

Los resultados de los experimentos se han analizado en base a la desviación mínima, máxima, media y estándar de Kappa, sensibilidad, especificidad y AUC. Estos resultados se han obtenido después de realizar 30 ejecuciones y una validación cruzada de 10 *folds*, a fin de evitar el sobre-entrenamiento y conseguir resultados estables.

2.3. Resultados

Los resultados obtenidos para cada índice y sobre las tres bases de datos, pueden observarse en la figura 2.1.

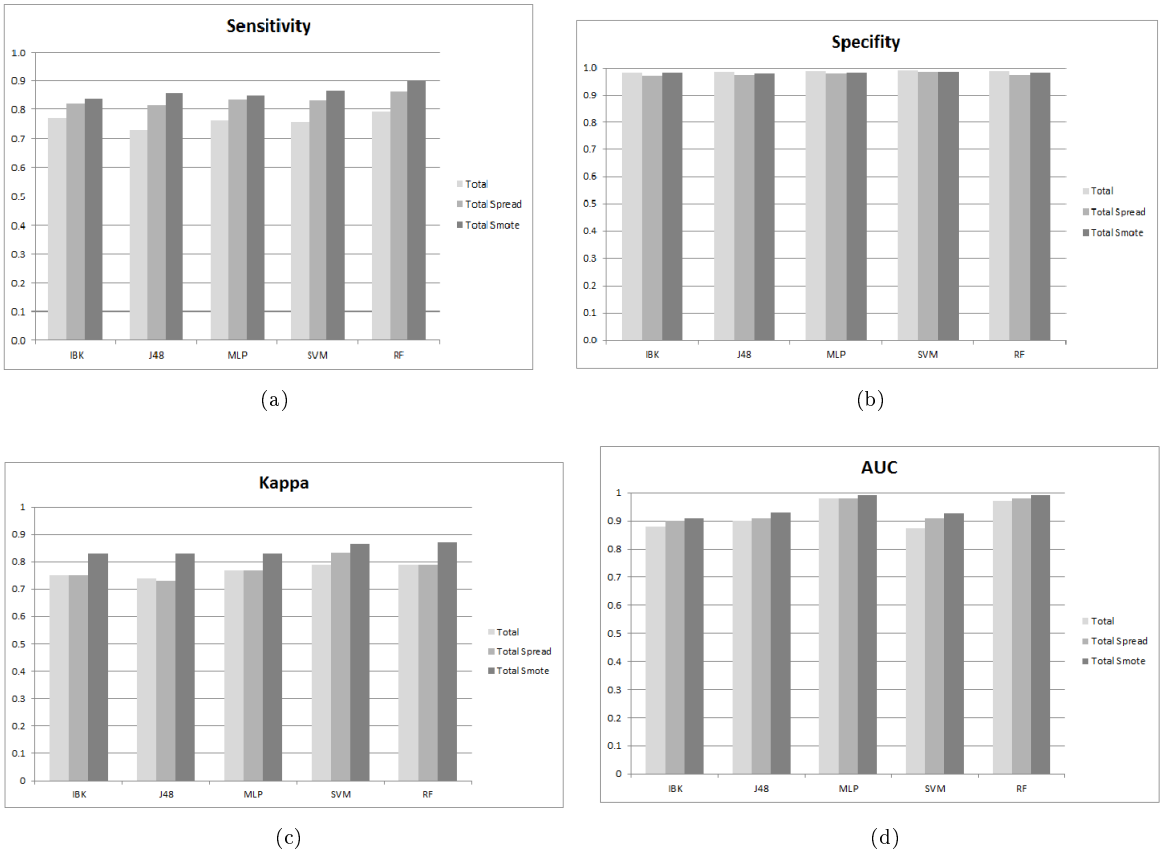


Fig. 2.1: Los resultados del experimento: índices (a) *Sensitivity*, (b) *Specifity*, (c) Kappa y (d) AUC para las tres bases de datos.

Con la base de datos original TOTAL, se obtienen buenos resultados para todos los algoritmos e índices de validación. En cuanto a los resultados obtenidos para el AUC, los índices van de 0,87 a 0,97 y hay que señalar que la mejora en el rendimiento entre los algoritmos es del 10 %, como ocurre para los algoritmos RF y IBK. La sensibilidad obtiene valores entre 0,73 y 0,79 lo cual indica que todos los algoritmos clasifican los casos positivos de la misma manera. Respecto de especificidad, todos los algoritmos obtienen resultados muy altos, por lo que la gran mayoría de los casos negativos se reconocen correctamente. Los valores de Kappa varían de 0,74 hasta 0,79, por lo que se considera un buen resultado [2].

	AUC			*Sensitivity*			*Specifity*			*Kappa*		
	TOTAL	*SPREAD*	*SMOTE*	*TOTAL*	*SPREAD*	*SMOTE*	*TOTAL*	*SPREAD*	*SMOTE*	*TOTAL*	*SPREAD*	*SMOTE*
RF	0.97	0.98	0.99	0.79	0.86	0.90	0.98	0.98	0.98	0.79	0.79	0.87
SVM	0.87	0.91	0.93	0.76	0.83	0.87	0.99	0.98	0.99	0.79	0.83	0.87
IBK	0.88	0.90	0.91	0.77	0.82	0.84	0.99	0.97	0.98	0.75	0.75	0.83
MLP	0.98	0.98	0.99	0.76	0.83	0.85	0.99	0.98	0.98	0.77	0.77	0.83
J48	0.90	0.91	0.93	0.73	0.82	0.86	0.99	0.97	0.98	0.74	0.73	0.83

Tabla 2.6: Los resultados del experimento: parámetros (a) *Sensitivity*, (b) *Specifity*, (c) Kappa y (d) AUC para las tres bases de datos.

Con la base de datos SPREAD, el rendimiento ha mejorado para todos los algoritmos. El índice AUC obtiene resultados entre el 0,90 y 0,98. La sensibilidad mejora considerablemente consiguiendo valores entre 0,82 y 0,86 para todos los algoritmos. Debemos hacer notar que el valor más bajo de esta base de datos, es mejor que el obtenido con el TOTAL. Respecto a la especificidad, todos los algoritmos obtienen resultados muy buenos. Los valores de Kappa varían de 0,73 a 0,83 y el valor obtenido por el algoritmo SVM destaca sobre el resto.

Por último, para la base de datos SMOTE se obtienen los mejores resultados en general. Con respecto a AUC, se obtienen índices entre 0,93 y 0,99; esto es, un rendimiento ligeramente mejor en comparación al resultado obtenido con la base de datos SPREAD. Con esta base de datos obtenemos los mejores valores de sensibilidad, obteniendo índices entre 0,86 y 0,90 para todos los algoritmos, y destacamos que el mejor valor se obtiene a partir del algoritmo RF. Respecto a la especificidad, todos los algoritmos obtienen resultados muy satisfactorios. Los valores de Kappa varían de 0,83 hasta 0,87, donde los mejores valores se obtienen con los algoritmos SVM y RF, ambos con un valor de 0,87.

Por último, se observa como los resultados mejoran cuando utilizamos las bases de datos SMOTE y SPREAD debido a la utilización de bases de datos más balanceadas. Acerca del tiempo requerido para el entrenamiento y la verificación del modelo, salta a la vista que los algoritmos MLP y SVM necesitan un tiempo de cálculo mayor. El algoritmo MLP puede tardar horas y un experimento realizado con SVM, un día. Para hacer frente a este problema, se recomienda programar dichos algoritmos mediante la computación paralela, especialmente el algoritmo SVM. En nuestro caso, hemos utilizado una implementación paralela propia basada en computación *multi-core*.

Capítulo 3

Aplicación OCR (Optical Character Recognition)

La aplicación de OCR se ha desarrollado mediante el programa estadístico R [28] el cual se distribuye gratuitamente desde 1995, y cuenta con una amplia red de colaboradores internacionales. El programa resulta práctico para el manejo de datos, gráficos y el desarrollo de cálculos estadísticos. El que se inspiró en los programas comerciales S y S-PLUS, es gratis y completamente programable, lo cual aporta flexibilidad. El programa R se complementa con los paquetes accesibles desde Internet. Estos paquetes se encuentran en los repositorios del grupo "*Comprehensive R Archive Network*" CRAN, en la página: https://cran.r-project.org y las versiones se suelen actualizar frecuentemente.

3.1. Descripción de la metodología de OCR

La tecnología OCR (*Optical Character Recognition*) engloba a un conjunto de técnicas que complementándose entre sí, se emplean para distinguir de forma automática entre los diferentes caracteres alfanuméricos existentes que aparecen en una imagen o en un archivo de texto, para así poder utilizar esta información en cualquier otro programa o aplicación que lo necesite. En nuestro caso, concretamente, se requiere el reconocimiento de los caracteres alfanuméricos que se encuentran en imágenes de sonar grabadas a bordo de buques pesqueros en campañas de pesca a cebo vivo del atún en el Golfo de Vizcaya. El potencial de la técnica OCR no se ciñe exactamente a los caracteres de un determinado alfabeto, sino que posibilita reconocer cualquier tipo de grafo. Dado que nuestro caso se centra en un modelo de sonar que aporta una serie de imágenes bien definidas (apartado 1.4) y que la variabilidad entre los distintos tipos de fuentes no es muy alta, de antemano se puede decir que nos enfrentamos a un problema no muy complicado y se esperan obtener unos resultados correctos.

Ante este escenario, existe un factor adverso: existe un desfase entre la frecuencia de actualización de la pantalla del sonar y la frecuencia de adquisición del dispositivo que graba imágenes de la pantalla del sonar (5 segundos). Debido a este desfase, en ciertos casos, el tiempo de la captura y de la actualización de la pantalla del sonar coinciden y esto causa ruido en la imagen grabada mediante el dispositivo de adquisición (ver figura 3.1). Este ruido distorsiona los caracteres a reconocer y causa errores en la segmentación y en el reconocimiento.

Fig. 3.1: Ruido generado debido al desfase en la grabación.

Debemos encontrar alguna solución a este problema y, sobre todo, intentar que la aplicación resulte lo más robusta posible.

Para ello, la aplicación OCR desarrollada se ha organizado en las siguientes etapas:

- Selección de las zonas de interés.
- Preprocesado de las imágenes.
- Segmentación de las zonas de interés.
- Extracción de características.
- Reconocimiento de los caracteres.
- Validación de resultados

A continuación, se describen los métodos que se han aplicado para resolver cada una de las etapas.

3.1.1. Selección de las zonas de interés

En este primer paso previo se han delimitado las zonas de estudio. Para ello se ha diseñado un programa en Matlab que recoge las coordenadas de los rectángulos de la imagen de los cuales queremos extraer los caracteres alfanuméricos. Esta aplicación capta las coordenadas en pantalla de la zona de interés mediante el cursor y genera las coordenadas de la imagen del rectángulo que las abarca. Este rectángulo se denomina el *Bounding Box* (BB) de la zona de interés y consta de dos coordenadas, x e y, y de la anchura y altura del rectángulo. A partir del BB extraemos las coordenadas de los píxeles que se encuentran dentro del rectángulo en las fases posteriores de esta tarea de OCR.

Estas zonas pueden observarse en la figura 3.2.

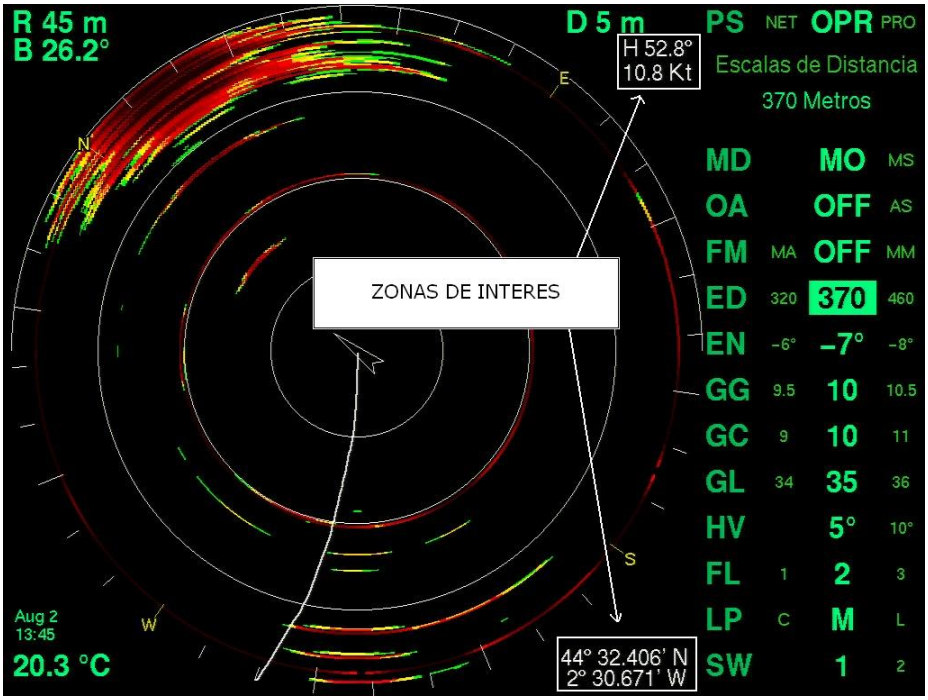


Fig. 3.2: Selección de las zonas de interés.

3.1.2. Preprocesado de las imágenes

Las técnicas de OCR se suelen aplicar en la digitalización de textos de algún libro o formularios rellenados manualmente. El proceso para los caracteres mecanografiados es más sencillo que en el caso de texto manuscrito. El objetivo del preprocesado es el de eliminar de la imagen cualquier tipo de ruido o imperfección que no pertenezca al carácter, así como normalizar el tamaño del mismo. Para ello se suelen emplear imágenes binarizadas, por lo que es conveniente convertir una imagen en escala de grises, o una de color, en una imagen en blanco y negro, de tal forma que se preserven las propiedades esenciales de la imagen.

Para ello emplearemos el histograma de la imagen donde se muestra el número de píxeles para cada nivel de grises que aparece a la imagen. Para binarizarla se debe de elegir el umbral a partir del cual todos los píxeles que no lo superen se convertirán en negro y el resto en blanco. En nuestro caso específico se ha empleado el umbral de Otsu [19]. Mediante este proceso obtenemos una imagen en blanco y negro con los contornos de los caracteres. A partir de las imágenes correctamente binarizadas se puede acometer el paso de la segmentación. En la siguiente figura (figura 3.3) se muestra el resultado de una imagen preprocesada.

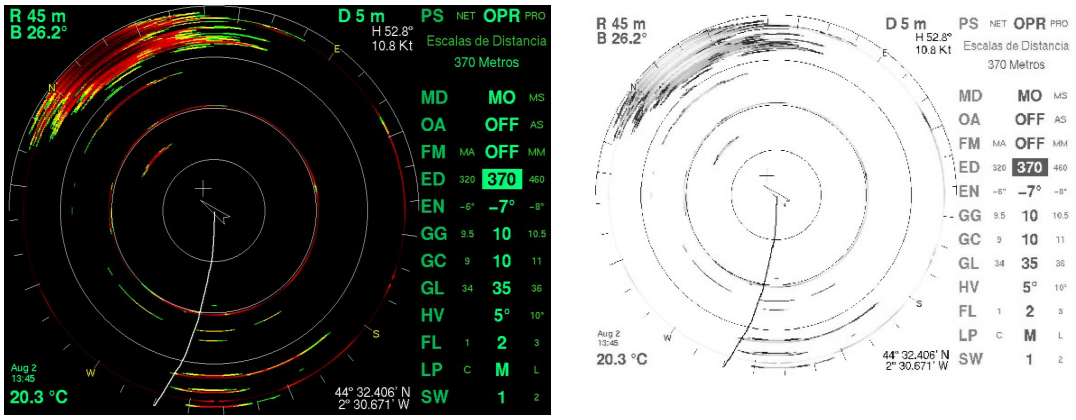


Fig. 3.3: Ejemplo de imagen original de sonar binarizada.

3.1.3. Segmentación de las zonas de interés

En nuestro caso, las zonas de interés están claramente localizadas, por lo que la ubicación de las áreas donde se tiene que aplicar la segmentación se sabe de antemano. Dentro de estas zonas de interés, dependiendo de las características de la imagen, se tendrán más o menos caracteres por lo que la demarcación de los caracteres se tiene que adaptar a cada imagen.

Para la segmentación se pueden explotar diferentes características de las imágenes: la longitud de los caracteres, los valles de separación entre letras o números distintos, las proyecciones del texto sobre líneas imaginarias y el posterior análisis de los perfiles obtenidos, etc. En nuestro caso concreto, al ser caracteres digitales provenientes a su vez de imágenes digitales obtenidas a partir del display del sonar, la problemática es más sencilla por lo que aplicaremos la búsqueda de los valles entre los distintos caracteres alfanuméricos.

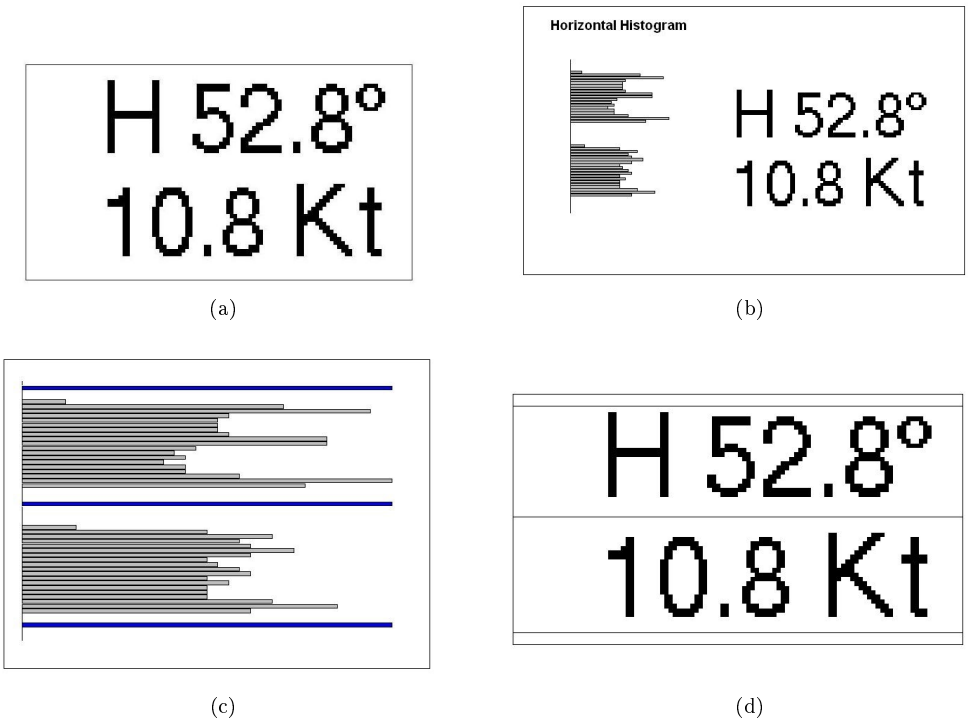
Como se ha comentado previamente, el desfase en la grabación nos genera problemas en la segmentación, obteniendo delimitaciones erróneas para los caracteres a reconocer (ver figura 3.4) y esto conllevará claramente resultados erróneos en los pasos posteriores del reconocimiento. A posteriori se comentará esta particularidad del error en la segmentación, donde se explicará cómo aprovecharemos esta circunstancia a modo de control para detectar las imágenes con presencia de ruido.

Fig. 3.4: Valle falso generado debido al desfase en la grabación, el cual genera problemas en la segmentación.

La segmentación la organizamos en dos pasos: en el primer paso se realiza una segmentación vertical, mediante la cual separaremos los diferentes parámetros que aparezcan dentro del área de interés y, a continuación, se procederá a la segmentación horizontal de las zonas previamente definidas en el primer paso, para así en este segundo paso poder realizar la segmentación horizontal y obtener la demarcación de cada carácter. A continuación, se explica el proceso de segmentación aplicado y se presentan los resultados de cada paso.

Para la segmentación vertical (figura 3.5), una vez que se ha cargado la imagen binarizada de la zona a estudiar, se calcula el histograma resultante de la suma de los píxeles negros por filas y se calculan los valles del histograma horizontal tomando como umbral el valor del cuantil 0,20. A continuación se puede observar el resultado numérico del histograma y los índices de los valles, para el caso en concreto que se presenta:

- Histograma horizontal: **0 0 0 0 0 0 0 4 24 32 19 18 18 18 19 28 28 16 14 15 13 15 15 20 34 26 0 0 0 0 0 0 0 0 5 17 23 20 21 25 21 17 18 20 21 17 19 17 17 17 23 29 21 0 0 0 0 0 0**
- Índice de los valles: **4 30 56**

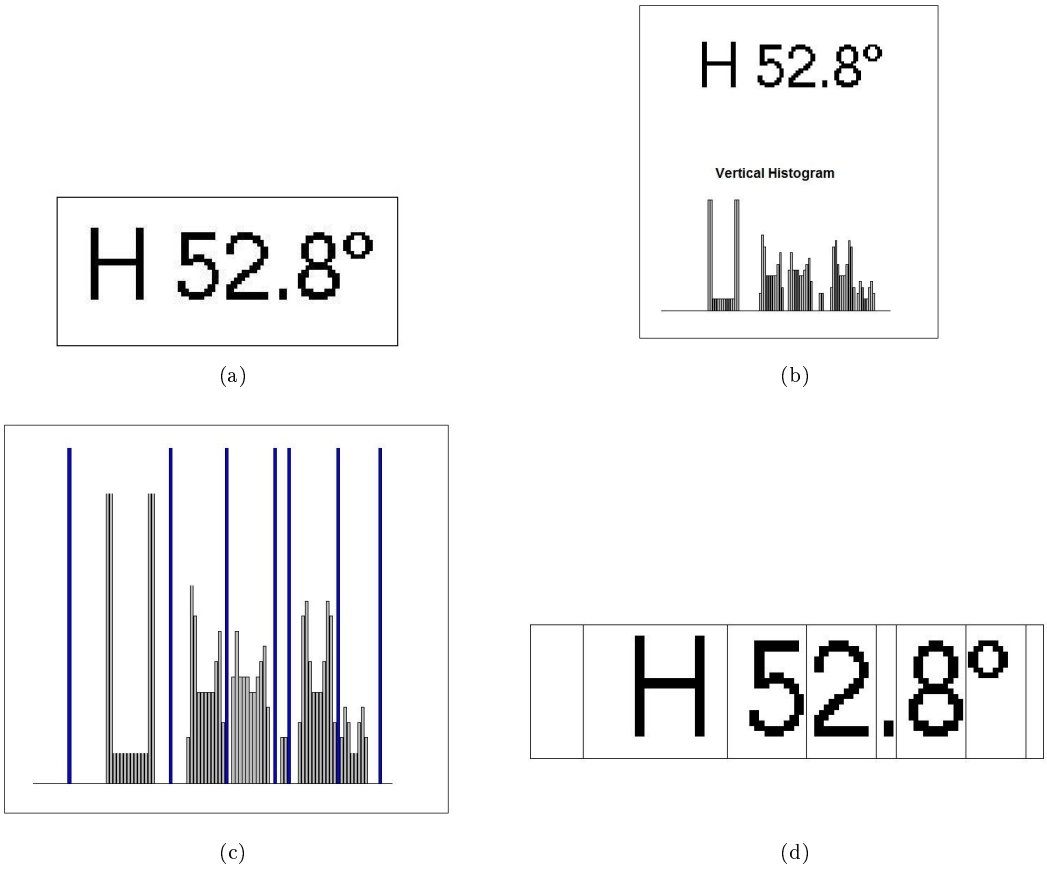


(a) (b)

(c) (d)

Fig. 3.5: Segmentación vertical: (a) Imagen binarizada de la zona a estudiar, (b) histograma resultante de la suma de los píxeles negros por filas junto a la imagen, (c) valles del histograma horizontal, (d) visualización final de la segmentación vertical.

Para cada zona resultante se obtiene la coordenada superior izquierda (X,Y), la anchura y la altura de la zona, quedando estas correctamente delimitadas.



(a) (b) (c) (d)

Fig. 3.6: Segmentación horizontal: (a) Imagen binarizada de la zona a estudiar, (b) histograma resultante de la suma de los píxeles negros por columnas junto a la imagen, (c) valles del histograma vertical, (d) visualización final de la segmentación horizontal.

En la segmentación horizontal (figura 3.6), se carga la imagen binarizada de la zona a estudiar, que en este caso son las zonas generadas previamente con la segmentación vertical. En las nuevas zonas se calcula el histograma resultante de la suma de los píxeles negros por columnas y se calculan los valles del histograma vertical tomando como umbral el valor del cuantil 0,20. A continuación se puede observar el resultado numérico del histograma vertical y los índices de los valles, para el caso en concreto que se presenta:

- Histograma vertical: **0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 19 2 2 2 2 2 2 2 2 2 2 19 19 0 0 0 0 0 0 0 0 0 3 13 11 6 6 6 6 6 8 10 4 0 0 7 10 7 7 7 6 6 7 8 9 5 0 0 0 3 3 0 0 0 4 11 12 8 6 6 6 8 12 11 4 0 3 5 4 2 2 4 5 3 0 0 0 0 0 0 0**
- Índice de los valles: **11 40 56 70 74 88 100**

Para cada carácter resultante se obtiene la coordenada superior izquierda (X,Y), la anchura y la altura de la zona, quedando estas correctamente delimitadas.

Una vez que se han obtenido las delimitaciones para cada carácter, y antes de acometer la etapa donde se extraen las características de cada carácter, realizamos un ajuste de dichas delimitaciones. El objetivo es eliminar los espacios en blanco tanto por arriba como por abajo respecto al eje horizontal y vertical de la imagen. Un vez realizado el ajuste, obtenemos las delimitaciones ajustadas definitivas, a partir de las cuales se van a extraer las características en la siguiente etapa. En la siguiente ilustración (figura 3.7) se observan las zonas ajustadas.

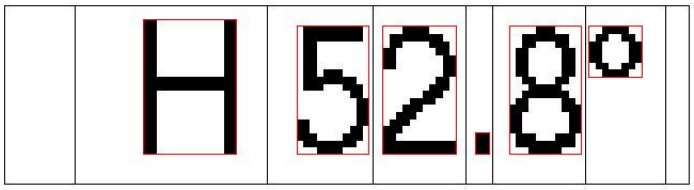


Fig. 3.7: Visualización final de la segmentación horizontal ajustada.

3.1.4. Extracción de características

Una vez realizada la segmentación, se dispone de una imagen binarizada para cada carácter. La información se presenta en una matriz bidimensional binaria (figura 3.8), y su dimensión (18x12), es relativa a los píxeles del carácter número "4", que se empleará a modo de ejemplo en este apartado.

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	0	0	0	0	0	0	0	1	1	0	0	0
[2,]	0	0	0	0	0	0	1	1	1	0	0	0
[3,]	0	0	0	0	0	0	1	1	1	0	0	0
[4,]	0	0	0	0	0	1	1	1	1	0	0	0
[5,]	0	0	0	0	1	1	1	1	1	0	0	0
[6,]	0	0	0	0	1	1	0	1	1	0	0	0
[7,]	0	0	0	1	1	0	0	1	1	0	0	0
[8,]	0	0	1	1	1	0	0	1	1	0	0	0
[9,]	0	0	1	1	0	0	0	1	1	0	0	0
[10,]	0	1	1	0	0	0	0	1	1	0	0	0
[11,]	1	1	1	0	0	0	0	1	1	0	0	0
[12,]	1	1	0	0	0	0	0	1	1	0	0	0
[13,]	1	1	1	1	1	1	1	1	1	1	1	0
[14,]	1	1	1	1	1	1	1	1	1	1	1	0
[15,]	0	0	0	0	0	0	0	1	1	0	0	0
[16,]	0	0	0	0	0	0	0	1	1	0	0	0
[17,]	0	0	0	0	0	0	0	1	1	0	0	0
[18,]	0	0	0	0	0	0	0	1	1	0	0	0

Fig. 3.8: Matriz bidimensional binaria que representa al carácter estudiado.

Para los procesos de reconocimiento, la matriz bidimensional se debe transformar en un vector, el cual se tomará como el vector de características del carácter estudiado.

La dimensión de dicho vector puede ser de tantas dimensiones como componentes tiene la matriz, pero el número de componentes puede llegar a ser importante y suponer un coste computacional elevado a la hora de procesar el mismo. Además, pequeñas diferencias en el vector, pueden causar diferencias importantes en la etapa de reconocimiento, por lo que se aplica una máscara de 8x5 a la imagen a fin de obtener una representación del objeto a reconocer más eficiente. De este modo, la aplicación de la máscara se puede considerar como un filtro de paso bajo.

Teniendo en cuenta que las dimensiones de todos los caracteres no son múltiplos de las dimensiones vertical y horizontal de la máscara, se han tenido que adaptar las máscaras de todos los caracteres a dicha máscara. Para ello se ha diseñado una función que realiza la división entera entre el número de píxeles de la imagen y la dimensión correspondiente de la máscara y guarda la división entera y el resto. A continuación, se reparte el resto por unidades entre los primeros elementos de la máscara. De este modo la máscara siempre resulta ser de 8x5 compuesta por elementos de diferentes dimensiones según las dimensiones de cada carácter. A continuación se puede observar la máscara que se le aplica al carácter "4" de dimensión 18x12.

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	0	0	0	0	0	0	0	1	1	0	0	0
[2,]	0	0	0	0	0	0	1	1	1	0	0	0
[3,]	0	0	0	0	0	0	1	1	1	0	0	0
[4,]	0	0	0	0	0	1	1	1	1	0	0	0
[5,]	0	0	0	0	1	1	1	1	1	0	0	0
[6,]	0	0	0	0	1	1	0	1	1	0	0	0
[7,]	0	0	0	1	1	0	0	1	1	0	0	0
[8,]	0	0	1	1	1	0	0	1	1	0	0	0
[9,]	0	0	1	1	0	0	0	1	1	0	0	0
[10,]	0	1	1	0	0	0	0	1	1	0	0	0
[11,]	1	1	1	0	0	0	0	1	1	0	0	0
[12,]	1	1	0	0	0	0	0	1	1	0	0	0
[13,]	1	1	1	1	1	1	1	1	1	1	1	0
[14,]	1	1	1	1	1	1	1	1	1	1	1	0
[15,]	0	0	0	0	0	0	0	1	1	0	0	0
[16,]	0	0	0	0	0	0	0	1	1	0	0	0
[17,]	0	0	0	0	0	0	0	1	1	0	0	0
[18,]	0	0	0	0	0	0	0	1	1	0	0	0

Fig. 3.9: Máscara 8x5 para el carácter "4".

Una vez que se obtiene la máscara correspondiente al carácter se calcula el porcentaje de píxeles negros en cada elemento de la máscara y éste será el resultado final del vector de características que describirá a la imagen en el apartado del reconocimiento. En la siguiente imagen se puede observar la matriz de porcentajes final.

0.00	0.00	0.83	0.5	0.0
0.00	0.55	0.83	0.5	0.0
0.16	0.66	0.50	0.5	0.0
0.50	0.16	0.50	0.5	0.0
0.83	0.00	0.50	0.5	0.0
1.00	1.00	1.00	1.0	0.5
0.00	0.00	0.50	0.5	0.0
0.00	0.00	0.50	0.5	0.0

Fig. 3.10: Matriz de porcentajes final del descriptor del carácter "4".

La salida de la función transformará la matriz de características de 8x5 en un vector lineal de 40 elementos.

Finalmente, además de la máscara se ha añadido un elemento más al vector de características de cada carácter. Este elemento describe la tipología del carácter, es decir, si se trata de un carácter normal (número, letra), alto ("º", "'") o bajo (","). Se observa que las delimitaciones previas al ajuste nos aportan información respecto al área de estudio de cada carácter y que las delimitaciones obtenidas después del ajuste, nos aportan información sobre las dimensiones propias del carácter y su situación respecto al área de estudio del mismo. Para calcular la tipología se han establecido relaciones geométricas entre las dimensiones del área de estudio y las dimensiones del carácter ajustado:

- Tipología normal: La altura del carácter ajustado tiene que ser mayor que el 32.5 % de la altura del área de estudio del carácter.
- Tipología alto: La altura del carácter ajustado tiene que ser menor que el 32.5 % de la altura del área de estudio del carácter y la coordenada Y del carácter tiene que estar por debajo del umbral 52 % que se refiere aproximadamente a la mitad de la imagen.
- Tipología bajo: La altura del carácter ajustado tiene que ser menor que el 32.5 % de la altura del área de estudio del carácter y la coordenada Y del carácter tiene que estar por encima del umbral 52 % que se refiere aproximadamente a la mitad de la imagen.

3.1.5. Reconocimiento de los caracteres

Finalmente, una vez extraídas las características del carácter, se tienen que clasificar los caracteres de una imagen entre un conjunto de símbolos posibles. El problema que se plantea consiste en desarrollar un método capaz de distinguir la clase a la que pertenece un objeto entre un conjunto limitado de clases posibles. En nuestro caso tenemos, números, letras, grados, minutos y comas, todos ellos del mismo tipo de fuente.

A continuación, se describe el método de reconocimiento diseñado. Una vez aplicado el metodo mediante el cual las imágenes están ya preprocesadas, segmentadas, normalizadas y transformadas

(extracción de características), se genera inicialmente la base de datos supervisada o de entrenamiento. Para ello se deben reunir un conjunto de datos etiquetados, es decir, un conjunto de caracteres con las características, tipología y clases a las que pertenecen[1]. Aprovechando que además de las características extraídas, tenemos la información de la tipología, generaremos tres bases de datos de entrenamiento según la tipología, una para el tipo "normal", otra para el tipo "alto" y una última para el tipo "bajo".

En la fase de clasificación o reconocimiento del carácter prototipo que se quiere clasificar, se tiene, por un lado, una base de datos supervisada organizada según la tipología y, por el otro lado, se tiene un nuevo objeto cuya clase no conocemos. No sabemos la clase del nuevo objeto, pero del mismo modo que hemos obtenido las características para la base de datos supervisada, obtenemos las características para el objeto prototipo. Una vez obtenidas dichas características, calcularemos las distancias entre el objeto prototipo y los objetos de la base de datos supervisada. Se pueden aplicar diferentes tipos de distancias: "*euclidean*", "*maximum*", "*manhattan*", "*canberra*", "*binary*" o "*minkowski*". En nuestro caso, hemos empleado la distancia de "Canberra". Se trata de la distancia entre dos vectores p y q, en un espacio vectorial real de n-dimensiones definida como:

$$d(p,q) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|'}$$

donde $p = (p_1, p_2, ..., p_n)$ y $q = (q_1, q_2, ..., q_n)$ son dos vectores.

El objeto de la base de datos supervisada que esté a menor distancia resultará ser el que más se asemeje a las características del objeto prototipo y, por lo tanto, la clase asociada a dicho objeto será la que se le asigne al objeto prototipo.

Cabe destacar que debido a que la base de datos supervisada está organizada según la tipología, la búsqueda se realiza también según su tipología (los objetos prototipo de tipología "normal" se compararán con los objetos de la base de datos de tipología "normal"). Así se optimiza el método y se evitan confusiones que se puedan dar entre caracteres que por tipología nunca deberían de coincidir en la fase de clasificación.

Para la generación de la base de test se ha utilizado el mismo método. La única diferencia respecto a la base de datos de entrenamiento es que, en la fase de reconocimiento se emplea la base de datos construida por las imágenes de la base de datos de entrenamiento.

3.1.6. Validación de resultados

Para la evaluación del método, en primer lugar, se compara el resultado de clasificación en la base de datos supervisada y en la base de datos de test. En ambas se estudian las estadísticas asociadas a los resultados de la matriz de confusión. La matriz de confusión se construye a partir de dos vectores: el primero es el resultado obtenido por la clasificación (clase predicha) y el segundo es el vector de caracteres supervisado (clase observada). Para la evaluación, se estudian los resultados estadísticos basados en el parámetro de Kappa y los valores de la precisión.

Por otro lado, en el siguiente paso se clasifica una serie de imágenes más amplia. Se trata de la serie de imágenes de sonar provenientes de la campaña de pesca del atún a cebo vivo en el Golfo de Vizcaya que se quiere estudiar. Dicha serie se procesa aplicando la metodología explicada hasta el momento y puesto que, en este caso, debido al gran número de imágenes a estudiar no se tiene una clasificación supervisada, el resultado se controla mediante una serie de autómatas o expresiones

[1] La clase de un dato se introduce de manera supervisada.

regulares. Estos autómatas nos indicarán si los resultados obtenidos son los esperados o, por lo contrario, si se tiene algún tipo de error y, a continuación, se evaluaran los resultados obtenidos.

3.2. Descripción del Programa Principal

En el programa principal desarrollado, lo primero que hacemos es establecer nuestro directorio de trabajo y, a continuación, cargamos los paquetes y las funciones necesarias para el desarrollo del programa.

Los paquetes empleados en este programa son:

- `library(lattice)` [26]: El paquete *lattice* es una implementan de los gráficos Trellis para R desarrollados principalmente por W. S. Cleveland para S-PLUS. Este sistema permite la visualización de datos multivariantes que es especialmente útil para explorar las relaciones o interacciones entre las variables. La principal ventaja de la utilización de las funciones de este paquete es la visualización de gráficos múltiples condicionados, de modo que los gráficos bivariados se dividen en varios gráficos según el valor de una tercera variable.

- `library(biOps)` [4]: Este paquete incluye varios métodos muy útiles en el procesamiento y análisis de imágenes. Proporciona operaciones geométricas, aritméticas, lógicas, morfológicas, tablas de búsqueda, máscaras para la detección de bordes (se incluyen los algoritmos de Roberts, Sobel, Kirsch, Marr-Hildreth y Canny, entre otros) y, ademas, las máscaras de convolución. El paquete también proporciona métodos de clasificación no supervisada tales como Isodata y *k-means*. También están disponibles los filtros y la transformada rápida de Fourier. Hasta el momento el paquete soporta imágenes JPEG y TIFF. Requiere la instalación de las librerías LIBJPEG y LIBTIFF.

- `library(abind)` [21]: Combina matrices multidimensionales en un solo vector. Esta es una generalización de las funciones genéricas de R: `cbind` y `rbind`. Funciona con vectores, matrices y matrices de dimensiones superiores. También proporciona las funciones `adrop`, `asup`, y `afill` para la manipulación, extracción y sustitución de los datos en los vectores.

- `library(plotrix)` [17]: Aporta una gran variedad de gráficas y de diferentes escalados para el color, los ejes y las etiquetas.

- `library(raster)` [24]: Permite leer, escribir, manipular, analizar y modelar los datos espaciales de tipo *raster*. El paquete implementa funciones básicas y de alto nivel y puede procesar archivos de gran tamaño.

- `library(EBImage)` [20]: EBImage es un paquete de R que proporciona funcionalidad de carácter general para la lectura, la escritura, el procesamiento y el análisis de imágenes. En el contexto de ensayos celulares basados en microscopía (para el cual fue desarrollado), EBImage ofrece herramientas para transformar las imágenes, segmentar las células y extraer características cuantitativas de las células.

- `library(e1071)` [18]: Funciones para la Transformada de Fourier de Tiempo Reducido, y algoritmos de clasificación tales como: *fuzzy clustering*, *support vector machines*, clasificador *Naive Bayes*...

- `library(caret)` [12]: Esta libreria incluye funciones para el entrenamiento y visualización de modelos de regresión y clasificación.

Las funciones empleadas se explican a continuación siguiendo el flujo lógico del programa que se puede observar en el siguiente diagrama:

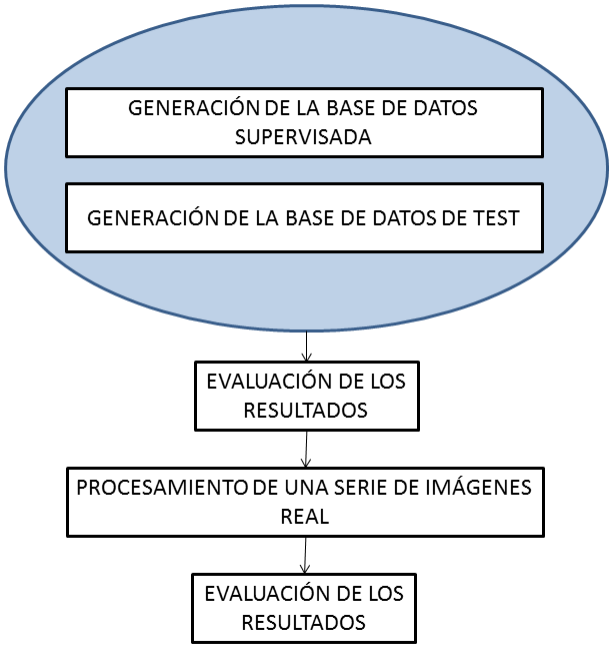


Fig. 3.11: Diagrama de flujo general.

Una vez seleccionadas las zonas dentro de las imágenes a procesar, donde se sitúan los parámetros que queremos reconocer, procesaremos una serie de imágenes, extraeremos sus características y las asociaremos a diferentes caracteres para, de este modo, generar una base de datos supervisada o de aprendizaje. En el segundo paso, procesaremos una serie de datos para generar una base de datos de test. En el tercer paso, compararemos las bases de datos que hemos generado, para evaluar el comportamiento del programa. En el cuarto paso, procesaremos las imágenes correspondientes a un estudio real y controlaremos el resultado mediante una serie de autómatas o expresiones regulares que, a su vez, nos indicaran la presencia de ruido o errores en las imágenes de entrada y extraeremos los parámetros que nos interesan para cada imagen. Finalmente, realizaremos una evaluación final de los resultados.

3.2.1. Generación de la base de datos supervisada (Paso 1)

En la figura 3.12 se presenta el diagrama de flujo para la generación de la base de datos supervisada y las funciones que utiliza.

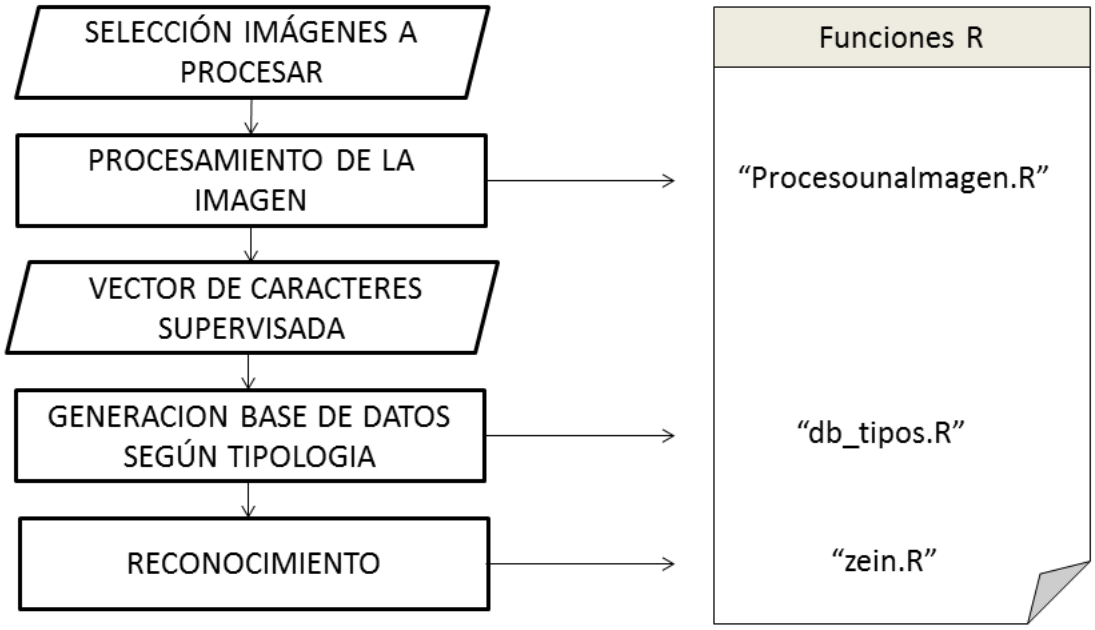


Fig. 3.12: Diagrama de flujo para la generación de la base de datos supervisada.

A continuación se presenta el diagrama de flujo que describe la relación entre las distintas funciones empleadas para la generación de la base de datos supervisada, función base que se emplea para realizar el procesamiento de las imágenes (`procesoUnaImagen.R`).

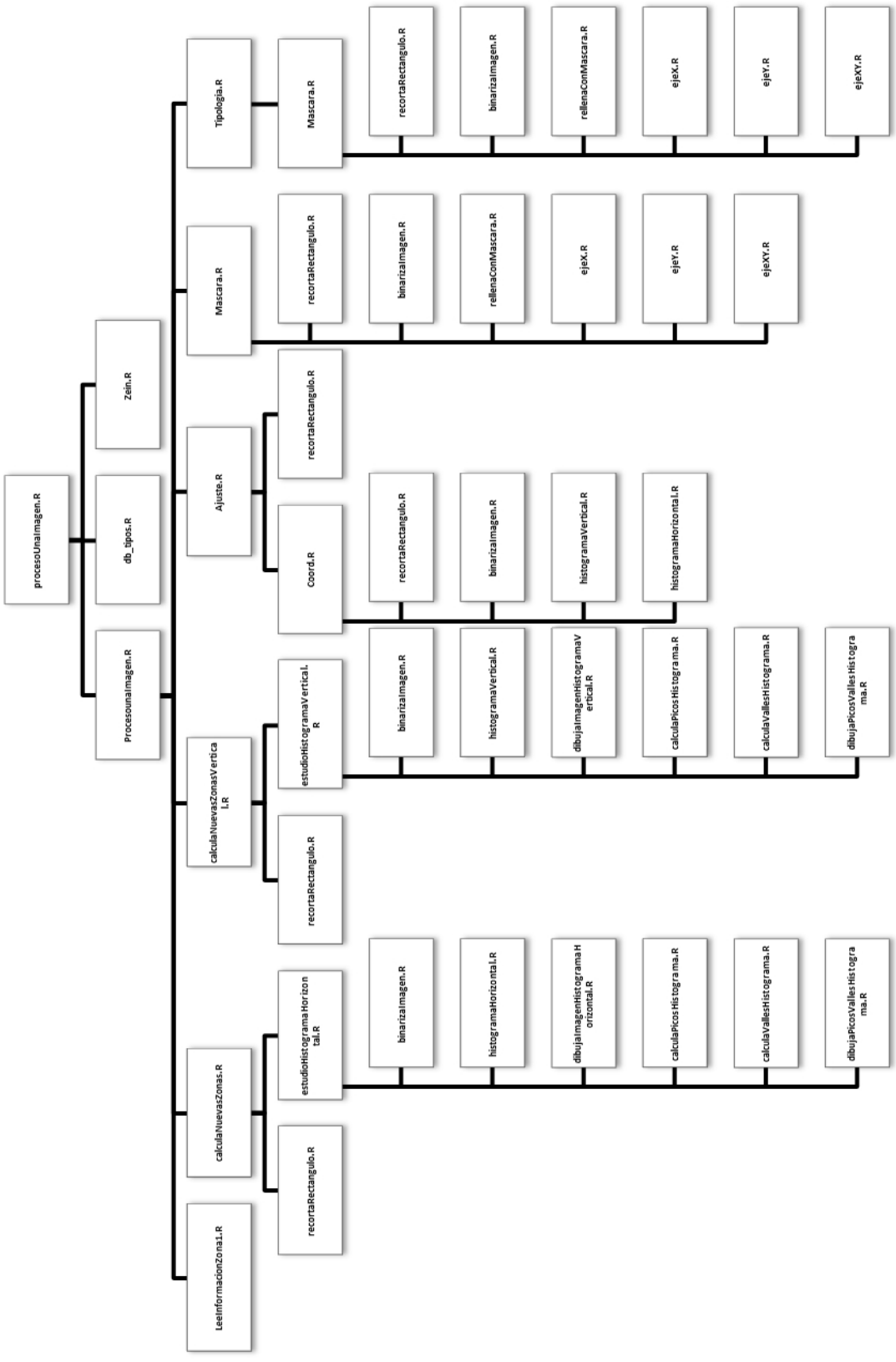


Fig. 3.13: Diagrama de flujo para las funciones que procesan la imagen.

Descripción de funciones

A continuación, se describen las distintas funciones empleadas durante el proceso de la imagen para la generación de la base de datos supervisada.

1. `source('leeInformacionZona1.R')`

   a) Descripción: Devuelve las coordenadas superior-izquierda y la dimensión (ancho y alto) de las zonas que se van ha analizar.

   b) Dependencias: Función `InformacionZona`.

2. `source('calculaNuevasZonas.R')`

   a) Parámetros de entrada:

   - `imagen(matrix)`: imagen a procesar.
   - `zonaBB(list)`: dimensiones de la zona a estudiar.
   - `q(integer)`: cuantil empleado para el umbral.
   - `flag(integer)`: valor empleado para ciertas condiciones.

   b) Salida:

   - `zonasnuevas`: dimensiones de las nuevas zonas.

   c) Descripción: Esta función devuelve las dimensiones de las nuevas zonas. Las nuevas zonas son las resultantes de la división horizontal.

   d) Dependencias: Funciones `recortaRectangulo`, `estudioHistogramaHorizontal`, `estudioHistogramaVertical`.

3. `source('recortaRectangulo.R')`

   a) Parámetros de entrada:

   - `imagen_i(matrix)`: imagen a procesar.
   - `bb(list)`: dimensiones de la zona a estudiar.

   b) Salida:

   - `imagen_o (matrix)`: imagen de la nueva zona recortada.

   c) Descripción: Esta función devuelve la zona de interes de la imagen ya recortada.

   d) Dependencias: Función `imgCrop` (es una función del paquete `Biops`, que recorta una imagen).

4. `source('estudioHistogramaHorizontal.R')`

   a) Parámetros de entrada:

   - `imagen(matrix)`: imagen de la zona ya recortada, la cual se va a dividir en diferentes zonas, según los cortes horizontales de la misma.

- `q(integer)`: cuantil empleado para el umbral.
- `flag(integer)`: valor empleado para ciertas condiciones.

b) Salida:

- `v$sumaH`: estructura que incluye el histograma horizontal(sumaH).
- `v$vallesH`: índices de los picos y valles del histograma horizontal.
- `v$horizontal`: unión de estos índices (horizontal) ordenados.

c) Descripción: Esta función devuelve la zona de interes de la imagen ya recortada.

d) Dependencias: Funciones `binarizaImagen`, `histogramaHorizontal`, `dibujaImagenHistogramaHorizontal`, `calculaPicosHistograma`, `calculaVallesHistograma`, `dibujaPicosVallesHistograma`.

5. `source('binarizaImagen.R')`

a) Parámetros de entrada:

- `x(matrix)`: imagen en escala de grises [0,255].

b) Salida:

- `x(matrix)`: imagen binaria [0,1].

c) Descripción: Esta función devuelve una imagen binaria [0,1].

d) Dependencias: Función `umbralOtsu`.

6. `source('umbralOtsu.R')`

a) Parámetros de entrada:

- `x(matrix)`: imagen en escala de grises [0,255].

b) Salida:

- `threshold(entero)`: umbral para la imagen binarizada.

c) Descripción: Esta función devuelve el umbral para la binarización.

7. `source('histogramaHorizontal.R')`

a) Parámetros de entrada:

- `x(matrix)`: imagen en escala de grises [0,255].
- `flag`: valor empleado para ciertas condiciones. Si `flag == 0`, no dibuja el histograma. Si `flag == 1`, dibuja el histograma.

b) Salida:

- `suma`: histograma.

c) Descripción: Función que calcula y dibuja el histograma horizontal.

d) Dependencias: Funciones `rowsums`, `barplot`.

8. `source('dibujaImagenHistogramaHorizontal.R')`

a) Parámetros de entrada:

- `x(matrix)`: imagen en escala de grises [0,255].
- `sumaV`: histograma horizontal.

b) Salida: Dibuja el histograma horizontal.

c) Descripción: Dibuja la imagen y el histograma horizontal.

d) Dependencias: Función `nuevaVentana` (crea una nueva ventana dependiendo del sistema operativo).

9. `source('calculaPicosHistograma.R')`

   a) Parámetros de entrada:

   - `suma(vector enteros)`: histograma.

   b) Salida:

   - `picos(vector enteros)`: vector con picos del histograma.

   c) Descripción: Dado un histograma, calcula donde comienzan los picos del histograma.

10. `source('calculaVallesHistograma.R')`

    a) Parámetros de entrada:

    - `suma(vector enteros)`: histograma.

    b) Salida:

    - `picos(vector enteros)`: vector con valles del histograma.

    c) Descripción: Dado un histograma, calcula donde comienzan los valles del histograma.

11. `source('dibujaPicosVallesHistograma.R')`

    a) Parámetros de entrada:

    - `suma(histogram)`: histograma.
    - `picos(vector entero)`: vector picos.
    - `valles(vector entero)`: vector valles.
    - `flag(integer)`: valor empleado para ciertas condiciones.

    b) Salida:

    - `x(matrix)`: imagen binaria [0,1].

    c) Descripción: Dibuja una imagen con los picos y valles del histograma.

    d) Dependencias: Función `barplot`.

12. `source('calculaNuevasZonasVertical.R')`

    a) Parámetros de entrada:

    - `imagen(matrix)`: imagen a procesar.

- `zonaBB(list)`: dimensiones de la zona a estudiar. Estas han sido generadas después de haber aplicado la función: `calculaNuevasZonas`. Las nuevas zonas son las resultantes de la división horizontal.
- `q(integer)`: cuantil empleado para el umbral.
- `flag(integer)`: valor empleado para ciertas condiciones.

b) Salida:

- `zonasnuevasV`: dimensiones de las nuevas zonas, zonas verticales referidas a cada carácter.

c) Descripción: Esta función devuelve las dimensiones de las nuevas zonas. Las nuevas zonas son las resultantes de la división vertical y se corresponden a los caracteres.

d) Dependencias: Funciones `recortaRectangulo`, `estudioHistogramaHorizontal`, `estudioHistogramaVertical`.

13. `source('histogramaVertical.R')`

a) Parámetros de entrada:

- `x(matrix)`: imagen en escala de grises [0,255].
- `flag`: valor empleado para ciertas condiciones. Si, flag == 0, no dibuja el histograma. Si, flag == 1, dibuja el histograma.

b) Salida:

- `suma`: histograma.

c) Descripción: Función que calcula y dibuja el histograma vertical.

d) Dependencias: Funciones `colSums`, `barplot`.

14. `source('dibujaImagenHistogramaVertical.R')`

a) Parámetros de entrada:

- `x(matrix)`: imagen en escala de grises [0,255].
- `sumaV`: histograma vertical.

b) Salida: Dibuja el histograma vertical.

c) Descripción: Dibuja la imagen y el histograma vertical.

d) Dependencias: Función `nuevaVentana`, crea una nueva ventana dependiendo del sistema operativo.

15. `source('calculaPicosHistograma.R')`

a) Parámetros de entrada:

- `suma(vector enteros)`: histograma.

b) Salida:

- `picos(vector enteros)`: vector con picos del histograma.

c) Descripción: Dado un histograma, calcula donde comienzan los picos del histograma.

16. `source('calculaVallesHistograma.R')`

    a) Parámetros de entrada:
    - `suma(vector enteros)`: histograma.

    b) Salida:
    - `picos(vector enteros)`: vector con valles del histograma.

    c) Descripción: Dado un histograma, calcula donde comienzan los valles del histograma.

17. `source('dibujaPicosVallesHistograma.R')`

    a) Parámetros de entrada:
    - `suma(histogram)`: histograma.
    - `picos(vector entero)`: vector de picos.
    - `valles(vector entero)`: vector de valles.
    - `flag(integer)`: valor empleado para ciertas condiciones.

    b) Salida:
    - `x(matrix)`: imagen binaria [0,1].

    c) Descripción: Dibuja una imagen con los picos y valles del histograma.

    d) Dependencias: Función `barplot`.

18. `source('ajuste.R')`

    a) Parámetros de entrada:
    - `imagen(matrix)`: imagen de la zona ya recortada la cual se va a dividir en diferentes zonas, según los cortes verticales de la zona.
    - `charBB(lista)`: dimensiones de cada carácter después de haber aplicado las funciones `calculaNuevasZonas` y `calculaZonasVertical`.
    - `q(integer)`: cuantil empleado para el umbral.
    - `flag(integer)`: valor empleado para ciertas condiciones.

    b) Salida:
    - `cc`: dimensiones de cada carácter ajustadas al cuerpo del carácter.

    c) Descripción: Esta función devuelve las dimensiones de cada carácter ajustadas al cuerpo del carácter.

    d) Dependencias: Funciones `coord`, `recortaRectangulo`, `binarizaImagen`.

19. `source('coord.R')`

    a) Parámetros de entrada:
    - `imagen(matriz)`: imagen de la zona ya recortada la cual se va a dividir en diferentes zonas, según los cortes verticales de la zona.

- `charBB(lista)`: dimensiones de cada carácter después de haber aplicado las funciones calculaNuevasZonas y calculaZonasVertical.
- `q(integer)`: cuantil empleado para el umbral.
- `flag(integer)`: valor empleado para ciertas condiciones.

b) Salida:

- `coord$hor`: coordenadas de corte horizontales.
- `coord$ver`: coordenadas de corte verticales.
- `coord$coord`: las 4 coordenadas de corte (N,S,E,O).

c) Descripción: Esta función devuelve las coordenadas de corte para las zonas ajustadas de cada carácter.

d) Dependencias: Funciones `recortaRectangulo`, `binarizaImagen`, `histogramaVertical`, `histogramaHorizontal`.

20. `source('mascara.R')`

a) Parámetros de entrada:

- `imagen(matrix)`: imagen a procesar.
- `imagenBB`: dimensiones de la zona a estudiar. En este caso las zonas de cada carácter.
- `q(integer)`: cuantil empleado para el umbral.
- `flag(integer)`: valor empleado para ciertas condiciones.

b) Salida:

- `lista:` vector de características.

c) Descripción: Se obtienen las características para cada carácter. Según el numero de (i) caracteres se obtendrá una matriz de características de (i x 40).

d) Dependencias: Funciones `recortaRectangulo`, `binarizaImagen`, `rellenaConMascara`, `ejeX`, `ejeY`, `ejeXY`, `creaLista`.

21. `source('rellenaConMascara.R')`

a) Parámetros de entrada:

- `a`: imagen a procesar. zona ajustada del carácter.
- `mr` : altura de la máscara. 8x.
- `mc`: anchura de la máscara. 5x.

b) Salida:

- `mascara`: imagen rellena de píxeles en blancos.

c) Descripción: Devuelve la imagen rellena de píxeles en blanco si se da el caso de que las dimensiones de la zona son menores que la máscara (dim(a)$<$8x5).

22. `source('ejeX.R')`

a) Parámetros de entrada:

- `bx`: anchura de la imagen a procesar (dim(a)[2]).
- `cx`: anchura de la máscara (múltiplo de 5).
- `dx`: el resto.

b) Salida:

- `array_x`: puntos de corte de la imagen para que siempre tenga una dimensión 5x a lo ancho.

c) Descripción: Devuelve los puntos de corte de la imagen para que siempre tenga una dimensión múltiplo de 5 de la anchura.

23. `source('ejeY.R')`

a) Parámetros de entrada:

- `bx`: altura de la imagen a procesar (dim(a)[1]).
- `cx`: altura de la máscara (múltiplo de 8).
- `dx`: el resto.

b) Salida:

- `array_y`: puntos de corte de la imagen para que siempre tenga una dimensión múltiplo de 8 de la altura.

c) Descripción: Devuelve los puntos de corte de la imagen para que siempre tenga una dimensión múltiplo de 8 de la altura.

24. `source('ejeXY.R')`

a) Parámetros de entrada:

- `xx`: anchura de la imagen a procesar (dim(a)[2]).
- `yy`: altura de la imagen a procesar (dim(a)[1]).
- `array_x`: puntos de corte de la máscara en el eje x.
- `array_y`: puntos de corte de la máscara en el eje y.

b) Salida:

- `array_mascara`: puntos de corte de la imagen para que siempre tenga una dimensión 8x5.

c) Descripción: Devuelve los puntos de corte de la imagen para que siempre tenga una dimensión 8x5.

25. `source('creaLista.R')`

a) Parámetros de entrada:

- `ir`: dim(a)[1]. Altura +1.
- `iq`: dim(a[2]. Anchura +1.
- `array_mascara`: puntos de corte de la imagen para que siempre tenga una dimensión 8x5.

- `a`: imagen del carácter.

b) Salida:

- `lista_per`: vector de características del carácter.

c) Descripción: Devuelve el vector de características del carácter.

26. `source('tipologia.R')`

a) Parámetros de entrada:

- `imagen(matrix)`: imagen del carácter a procesar.
- `char`: dimensiones del carácter a procesar.
- `char2`: dimensiones del carácter ajustado a procesar.
- `flag(integer)`: valor empleado para ciertas condiciones.

b) Salida:

- `tipo`: tipología del carácter.

c) Descripción: Se obtiene la tipología del carácter.

d) Dependencias: Función `mascara`.

A continuación, se describe la función empleada para la generación de las bases de datos según la tipología.

1. `source('db_tipos.R')`

a) Parámetros de entrada:

- `rr(char)`: matriz de caracteres. Es la salida obtenida con la función `procesoUnaImagen`, a la cual se le ha añadido una columna con los caracteres de manera supervisada.

b) Salida:

- `db$normal`: base de datos supervisada, que ha sido generada empleando solamente los casos de tipología "normal".
- `db$bajo`: base de datos supervisada, que ha sido generada empleando solamente los casos de tipología "bajo".
- `db$alto`: base de datos supervisada, que ha sido generada empleando solamente los casos de tipología "alto".

c) Descripción: Esta función recibe una matriz con las características y tipología de una serie de caracteres y genera tres bases de datos supervisadas según su tipología.

d) Dependencias: Función `aggregate` (divide los datos en subconjuntos, calcula las medias para cada uno y devuelve el resultado en una forma conveniente).

A continuación, se describe la función empleada para el reconocimiento del carácter.

1. `source('zein.R')`

a) Parámetros de entrada:

- `masc_z`: características de todos los parámetros de las zonas indicadas.
- `bd_z`: base de datos que se emplea para comparar las características del carácter actual con las de la base de datos.
- `id_z`: índice del carácter actual.

b) Salida:

- `letra(char)`: devuelve la letra que menor distancia ha obtenido.

c) Descripción: Esta función recibe una matriz con las características y tipología de una serie de caracteres y genera tres bases de datos supervisadas según su tipología.

d) Dependencias: Función `dist` (esta función calcula y devuelve la matriz de distancia; esto es, calcular las distancias entre las filas de una matriz de datos).

3.2.2. Generación de la base de datos de test (Paso 2)

A continuación se presenta el diagrama de flujo para la generación de la base de datos de test y las funciones que utiliza. El procesamiento de la imagen es exactamente el mismo que el del paso 2 y la relación de las funciones también, con la única diferencia que no se llama a la función para generar la base de datos (`db_tipos.R`). Se utiliza la base de datos supervisada para realizar el reconocimiento y, a continuación, se le añade el vector de los caracteres etiquetados de manera supervisada, para así poder evaluar los resultados en el siguiente paso.

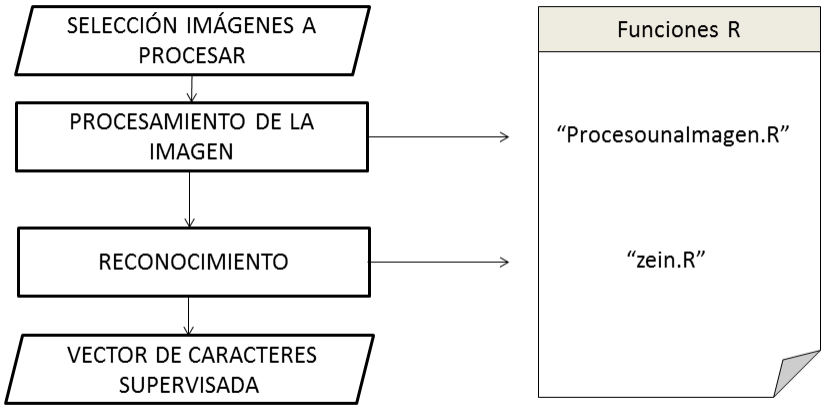


Fig. 3.14: Diagrama de flujo para la generación de la base de datos de test.

3.2.3. Evaluación de los resultados para las bases de datos (Paso 3)

Para evaluar los resultados obtenidos en la generación de las bases de datos supervisadas, se ha generado una matriz de confusión a partir del vector obtenido en el reconocimiento y el vector empleado en el etiquetado supervisado. En la figura 3.15 se puede observar la matriz de confusión resultante.

row.names	'	.	0	1	2	3	4	5	6	7	8	9	H	K	N	º	t	W
'	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
.	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
º	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3

Fig. 3.15: Matriz de confusión resultante para la base de datos supervisada.

Se observa como todos los casos positivos se encuentran en la diagonal principal, lo cual indica que la clasificación ha sido excelente. Puesto que las imágenes que se han empleado para la generación de la base de datos son las mismas que las se han clasificado, estos resultados eran de esperar y confirman el buen funcionamiento de la metodología OCR desarrollada. Estos resultados, se reflejan finalmente en las estadísticas extraídas empleando la herramienta "*overall*" del paquete "`caret`" que calcula la precisión y el parámetro de *Kappa* como se observa en la tabla 3.1:

Precisión	Kappa
1	1

Tabla 3.1: Precisión y kappa para la base de datos supervisada.

Para evaluar los resultados obtenidos en la generación de las bases de datos de test, se ha realizado el mismo estudio que con la base de datos supervisada. Se ha generado una matriz de

confusión a partir del vector obtenido en el reconocimiento y el vector empleado en el etiquetado supervisado. En la figura 3.16 se puede observar la matriz de confusión resultante.

row.names	'	.	0	1	2	3	4	5	6	7	8	9	H	K	N	o	t	W
'	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
.	0	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11

Fig. 3.16: Matriz de confusión resultante para la base de datos de test.

Se observa cómo todos los casos positivos se encuentran en la diagonal principal, lo cual indica que la clasificación ha sido excelente. Puesto que, en este caso, las imágenes que se han empleado para la generación de la base de datos no son las mismas que las se han clasificado, se confirma nuevamente el buen funcionamiento de la metodología OCR desarrollada. Estos resultados, se reflejan finalmente en las estadísticas extraídas empleando la herramienta "*overall*" del paquete "`caret`" que calcula la precisión y el parámetro de *Kappa* como se observa en la siguiente tabla:

Precisión	Kappa
1	1

Tabla 3.2: Precisión y kappa para la base de datos supervisada.

3.2.4. Procesamiento de una serie de imágenes real (Paso 4)

A continuación se presenta el diagrama de flujo general para el procesado de una serie de imágenes aplicado a un caso real y las funciones que utiliza.

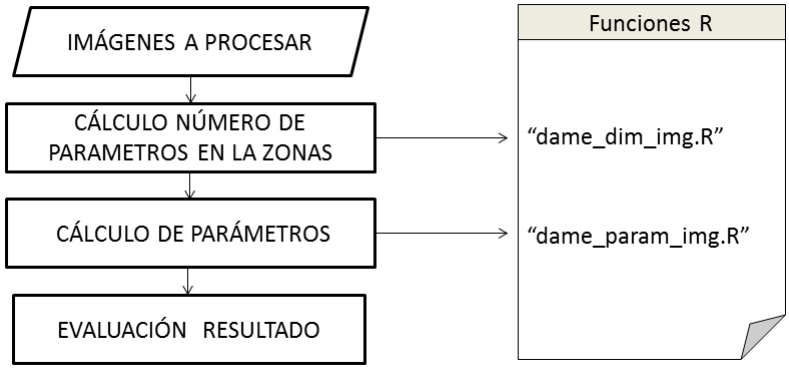


Fig. 3.17: Diagrama de flujo para el procesamiento de una serie de imágenes real.

En el diagrama se observa cómo, en primer lugar, recibe las imágenes a procesar; a continuación, calcula el número de parámetros que se tiene en las zonas de interés; después, se procesan dichas zonas y se calculan los parámetros de cada zona y, finalmente, se evalúa el resultado obtenido.

Estos procesos comparten la mayor parte de las funciones empleadas en los pasos anteriores, pero en las nuevas funciones ("`dame_dim_img.R`" y "`dame_param_img.R`") se aplican de manera distinta para obtener los parámetros de las zonas de interés por separado. Para entender el nuevo proceso se presentan los diagramas que describen los procesos para el cálculo del número de parámetros de cada zona y para el cálculo de dichos parámetros.

Fig. 3.18: Diagrama de flujo para las funciones que procesan la imagen.

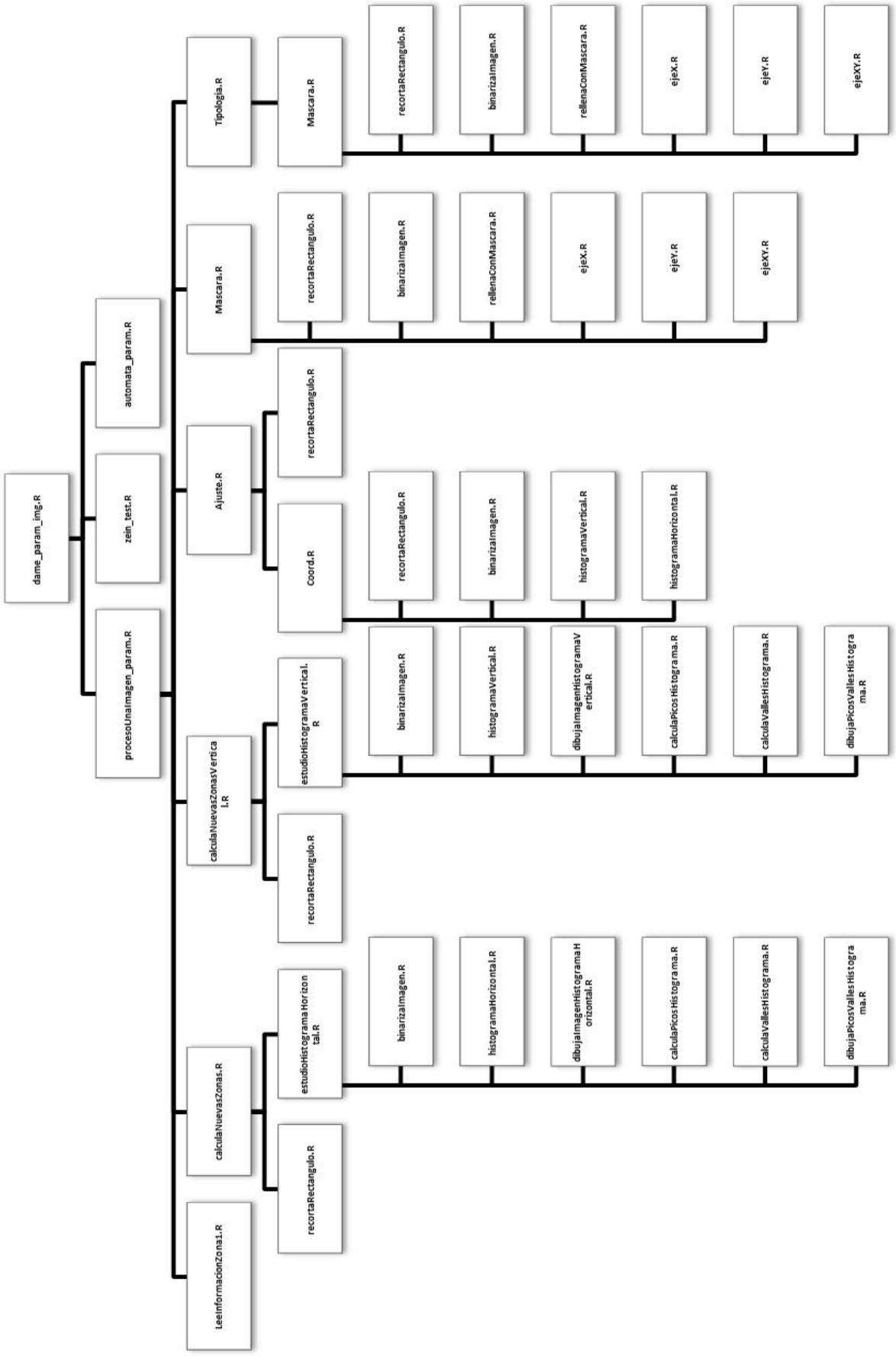


Fig. 3.19: Diagrama de flujo para las funciones que calculan los parámetros.

En los pasos anteriores, donde se han generado las bases de datos de entrenamiento y de test dentro de las zonas de interés, los diferentes caracteres que se han extraído y reconocido no han sido organizados por parámetros, pero para nuestro caso particular necesitamos diferenciar cada parámetro a fin de poder trabajar, a posteriori, con ellos. Para ello, la función "`dame_dim_img.R`" nos indica el número de parámetros que se tienen dentro de la zona de interés.

En la función "`dame_param_img.R`", el procesado de las imágenes la realizamos gracias a la función "`procesoUnaImagen_param.R`" y el reconocimiento de los caracteres de la zona estudiada lo realizamos con la función "`zein_test.R`". Estas dos funciones son las mismas que las empleadas para el proceso de generación de las bases de datos de entrenamiento y de test en los pasos previos, la única diferencia reside en las zonas en las que se aplica este procesado; en este caso, se modifica la llamada a la función "`mascara.R`" y el proceso se organiza según las zonas delimitadas por cada parámetro que nos ha dado la función "`dame_dom_img.R`".

Una vez que se tienen los caracteres de cada zona reconocidos, se realiza un control mediante la función "`automata_param.R`". La localización, el diseño y la organización de los caracteres, dentro de cada zona o parámetro, son fijas. Puede haber más o menos valores numéricos que puedan modificar las delimitaciones y el número de caracteres alfanuméricos. Todas estas variaciones se especifican en unas expresiones regulares o autómatas que empleamos para asegurarnos que el reconocedor ha devuelto una respuesta correcta. Dichas expresiones las podemos observar en la figura 3.20.

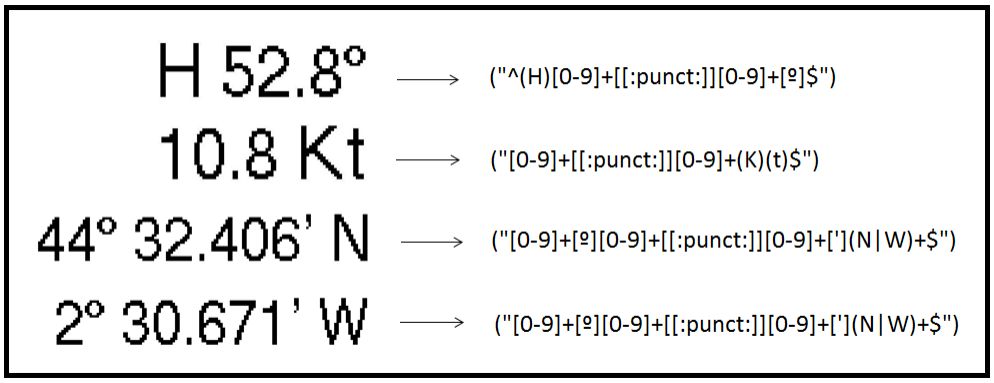


Fig. 3.20: Expresiones regulares o autómatas empleados para controlar la salida del clasificador.

Descripción de funciones

Las funciones que no se describen en este apartado son las mismas que se han empleado en las anteriores etapas. Para el procesado de las imágenes se describen las funciones "`dame_dim_img.R`" y "`leeInformacionZonas.R`". Para el cálculo de los parámetros se describen las funciones:

1. `source('dame_dim_img.R')`

   a) Parámetros deentrada:

- `nombre`: nombre de la imagen.

b) Salida:

- `nZonas_TOTAL`: número de zonas de la imagen.

c) Descripción: Se obtiene el número de zonas de la imagen.

d) Dependencias: Funciones `leeInformacionZona1`, `calculaNuevasZonas`, `leeInformacionZonas`.

2. `source('leeInformacionZonas.R')`

a) Parámetros entrada:

- `zonasTotal(list matrix)`: zonas a estudiar generadas a partir de la función `calculaNuevasZonas`.

b) Salida:

- `todas`: nuevo objeto con las zonas a estudiar.

c) Descripción: Esta función genera un nuevo objeto con las zonas a estudiar.

3. `source("procesoUnaImagen_param.R")`

a) Parámetros de entrada:

- `ind(char)`: indica el nombre de la imagen a procesar.
- `ii(integer)`: indica el número de la zona que se va a procesar (parámetro).

b) Salida:

- `res$mascTotal`: máscara de características de todos los parámetros de las zonas indicadas. En una matriz de (ix40) donde (i) es el número de caracteres que hay en las zonas indicadas.
- `res$tipo_char`: vector con la tipología ("alto","bajo"o"normal") de todos los parámetros de las zonas indicadas.

c) Descripción: Esta función devuelve las características de los caracteres que hay en las zonas indicadas. En este caso las zonas son las referidas a un parámetro. A cada carácter se le asigna un vector de características y una tipología.

d) Dependencias: Funciones `leeInformacionZona1`, `calculaNuevasZonas`, `leeInformacionZonas`, `calculaNuevasZonasVertical`, `ajuste`, `mascara`, `tipología`

4. `source("zein_test.R" )`

a) Parámetros de entrada:

- `masc_z`: características de todos los parámetros de las zonas indicadas.
- `bd_z`: base de datos que se empleará para comparar las características del carácter actual con las de la base de datos.
- `id_z`: índice del carácter actual.

b) Salida:

- `letra(char)`: devuelve la letra situada a menor distancia.

c) Descripción: Esta función recibe una matriz con las características y tipología de una serie de caracteres y genera tres bases de datos supervisadas según su tipología.

d) Dependencias: Función "`dist`" genérica de R, la cual calcula y devuelve la matriz de distancia, calcula las distancias entre las filas de una matriz de datos.

5. `source("automata_param.R ")`

a) Parámetros de entrada:

- `num_grep`: número de la zona a procesar.

b) Salida:

- `grep`: expresión regular de la zona especificada que se empleará para controlar la salida.

c) Descripción: Se obtiene la expresión regular de la zona especificada que se empleará para controlar la salida.

3.2.5. Evaluación de los resultados para la serie de imágenes real (Paso 5)

Para evaluar los resultados obtenidos en la clasificación realizada sobre todas las imágenes correspondientes a una serie de imágenes relativas a un caso real (104 imágenes) se han calculado los porcentajes de acierto y de error.

Se observa cómo en el reconocimiento se han detectado diferentes tipos de errores. El error que se muestra con la etiqueta "ERROR–>RUIDO", es un error de clasificación y el error que se muestra como "ERROR", es un error de segmentación. Como se ha explicado previamente, los dos tipos de errores provienen del ruido generado en las imágenes debido al desfase entre la frecuencia de actualización de la pantalla del sonar y la frecuencia de adquisición del dispositivo que graba imágenes de la pantalla del sonar:

```
"";"v1"
"emaitza1";"HEGALABUR_19288.jpg : H344.8º9.7Kt44º31.436'N2º31.231'W : OK"
"emaitza1";"HEGALABUR_19289.jpg : H345.8º9.6Kt44º31.436'N2º31.231'W : OK"
"emaitza1";"HEGALABUR_19290.jpg : H345.5º9.6Kt44º31.438'N2º31.233'W : OK"
"emaitza1";"HEGALABUR_19291.jpg : H344.6º9.9Kt44º31.444'N2º31.235'W : OK"
"emaitza1";"HEGALABUR_19299.jpg : H341.6º9.6Kt44º31.467'N2º31.245'W : OK"
"emaitza1";"HEGALABUR_19300.jpg : H389.7º9.9Kt44º31.461''N2º31.246'W : ERROR"
"emaitza1";"HEGALABUR_19333.jpg : H337.3º9.1Kt44º31.565'N2º31.307'W : OK"
"emaitza1";"HEGALABUR_19334.jpg : H336.5º9.1Kt44º31.567'N2º31.310'W : OK"
"emaitza1";"HEGALABUR_19335.jpg : ERROR--> RUIDO"
                                  . . .
```

Fig. 3.21: Ejemplo del resultado final obtenido.

- Error de clasificación: el caso de "ERROR–>RUIDO" aparece cuando, debido al ruido presente en las áreas de estudiadas, el preprocesado de la imagen consigue segmentar los caracteres correctamente pero clasifica la tipología del carácter de forma errónea. Al pasar al clasificador la máscara de los caracteres con la tipología errónea, se reconocen menos caracteres de los que se debieran clasificar, por lo que al detectar esta diferencia entre dimensiones, se declara la zona estudiada y a su vez la imagen, como un caso de "ERROR–>RUIDO". En la figura 3.22 se puede observar una imagen que se ha clasificado como "ERROR–>RUIDO".

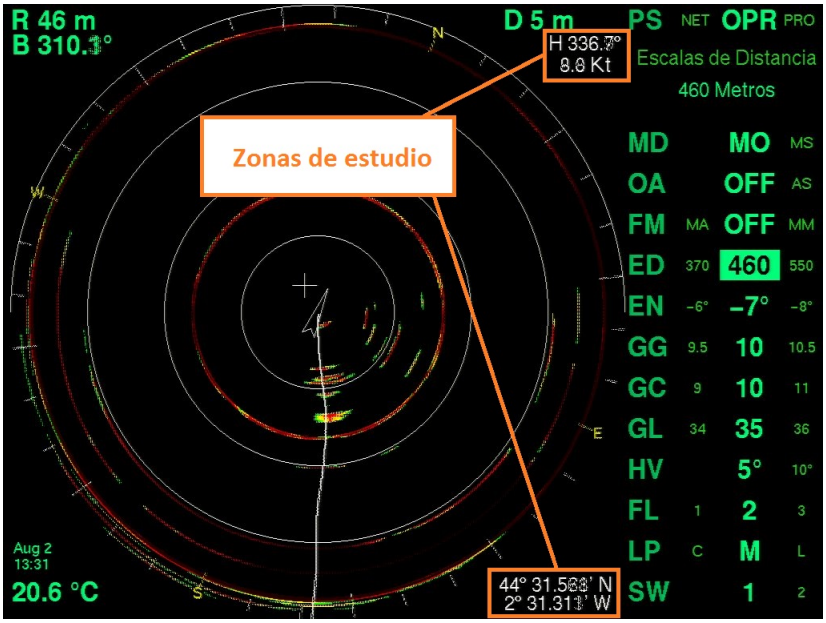


Fig. 3.22: Imagen clasificada como "RUIDO".

- Error de segmentación: el caso de "ERROR" aparece cuando, debido al ruido presente en las áreas estudiadas, el preprocesado de la imagen consigue tanto segmentar correctamente las zonas de estudio como la tipología. Por lo que en este caso, no se tienen problemas de dimensionalidad. El problema, para este caso, lo detectamos gracias al control realizado mediante los autómatas. Debido al ruido, se extraen unas características distorsionadas de los caracteres y esto deriva en la incorrecta clasificación de los mismos. De este modo, las secuencias de imágenes de cada parámetro resultan ser erróneas y no cumplen los patrones exigidos por los autómatas dando lugar a que estos casos se clasifiquen como casos de "ERROR". En la figura 3.23 se puede observar una imagen que se ha clasificado como "ERROR".

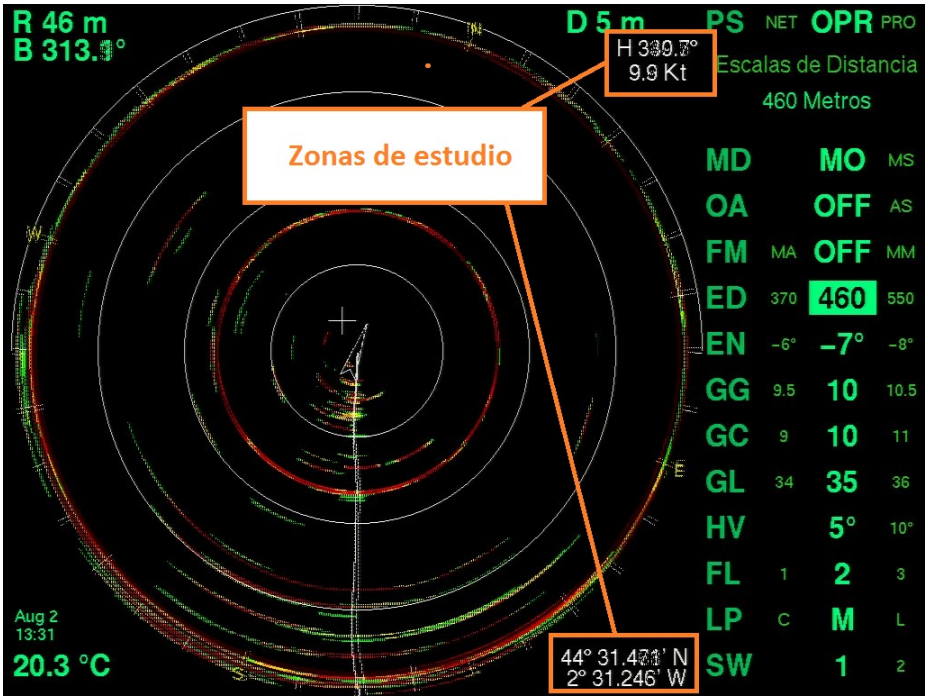


Fig. 3.23: Imagen clasificada como “ERROR”.

Una vez descubierta la naturaleza de los errores, resultaría interesante anticiparse a ellos. Para ello, se podría aplicar un sistema de muestreo doble de las imágenes. Esto es, en vez de grabar una imagen cada 5 segundos como hasta ahora, se podrían adquirir 2 imágenes cada 5 segundos. De este modo, realizando un AND a estas dos imágenes, eliminaríamos el ruido existente.

Para la evaluación de los resultados de este caso no se tiene información de las clases predichas y observadas. En nuestro caso, todas las clases han sido predichas por el clasificador por lo que se ha calculado el porcentaje de aparición de los mismos. En la siguiente tabla (ver tabla 3.3) se observan dichos porcentajes, los cuales indican un resultado excelente.

	OK	ERROR	RUIDO
Porcentaje de aparición	0.978	0.012	0.009

Tabla 3.3: Porcentajes de aparición de las clases “OK”, ”RUIDO” y ”ERROR”

Capítulo 4

Conclusiones

En este apartado, por un lado, se exponen las conclusiones extraídas de la validación realizada del preprocesado de imágenes semi-automático (PISA) de sonar de largo alcance propuesto por [29] mediante un estudio comparativo entre diferentes algoritmos de aprendizaje y, por otro lado, las conclusiones extraídas de la aplicación OCR desarrollada para la extracción de parámetros de situación y configuración del sonar.

Respecto al trabajo de clasificación supervisada realizado, los resultados obtenidos para la clasificación supervisada indican que todos los algoritmos son eficientes y eficaces en general pero el resultado conseguido por el algoritmo *RandomForest,* en concreto, es excelente. Gracias a estos resultados obtenidos mediante técnicas de minería de datos se concluye que, el uso de imágenes de sonar junto al preprocesado de imágenes semi-automatico (PISA) es adecuado para discernir morfológicamente las regiones estudiadas en la clasificación binaria de presencia/ausencia del atún en las imágenes de sonar.

Hay que tener en cuenta que, en el capítulo de la clasificación supervisada, el conjunto de imágenes seleccionado es un conjunto donde todas las imágenes pertenecen o al caso de atún (presencia) o al caso de no-atún (ausencia). Para ello se han empleado capturas de imágenes tomadas a bordo de buques pesqueros en la campaña del atún donde había observadores a bordo apuntando, en todo momento, todos los eventos de la actividad tanto pesquera como científica. De este modo, se garantiza que las imágenes que se han tratado, corresponden a la clase adecuada (atún o no-atún), y se evita el tratamiento de imágenes correspondientes a otro tipo de especie (especialmente, la anchoa) que, en ciertos casos y circunstancias, pueden devolver una respuesta similar a la que pueda dar un banco de atún. Esto quiere decir que si aplicáramos el modelo a una serie de imágenes correspondiente a una campaña de atún completa, probablemente los resultados no serían tan buenos y, dentro de la clase positiva de presencia de atún, podríamos dar por buenos datos de bancos de peces de otras especies. En conclusión, para poder validar este tipo de estudios, es vital obtener información lo más precisa posible de los *logbooks* de abordo en las diferentes campañas de pesca,pesca, así como estudiar el modo de identificar la especie observada por el sonar, ya que cada especie puede presentar sus características particulares..

Respecto a la aplicación OCR, los resultados obtenidos tanto en la generación de las bases de datos (entrenamiento y test) como en el procesado de una serie de imágenes real dejan un balance positivo. La metodología diseñada ha resultado ser la correcta y los resultado obtenidos han confirmado la robustez de la aplicación.

La manera de seleccionar las zonas de estudio y sus características, son un tema a tener en cuenta. A la hora de seleccionar las coordenadas de las zonas de estudio se ha recurrido a un software alternativo (Matlab) al que se ha empleado para desarrollar la aplicación de OCR (R), permitiendo obtener las coordenadas de las zonas de estudio de manera más rápida y empleando una herramienta de interfaz gráfica de usuario. Respecto a las características de las zonas de estudio, comentar que, tanto la metodología como la aplicación diseñada, funciona para una serie de zonas concretas y delimitadas dentro de la imagen. Esta característica podría resultar no ser la más idónea para aplicaciones OCR donde las áreas de búsqueda de los caracteres sean móviles, pero para nuestro caso concreto esto no es un problema, puesto que las áreas de estudio son fijas. Y si por algún motivo las zonas de estudio cambian, sólo se tendrían que redefinir las delimitaciones de las nuevas zonas y procesar las imágenes con la misma metodología. En relación a este tema, dentro de cada área de estudio, el número de caracteres y su localización sí que varía, por lo que la localización de los caracteres no es fija, pero esto se solventa correctamente con la segmentación.

Respecto a los siguientes pasos de la aplicación, la extracción de características y el reconocimiento del carácter, comentar que los resultados obtenidos avalan el buen diseño de ambos dos.

Sobre los objetivos generales del proyecto, cabe mencionar que tanto la validación del PISA como el desarrollo de la aplicación OCR para la extracción de parámetros de situación y configuración del sonar, representan un paso hacia adelante en el objetivo general del estudio. Con los nuevos parámetros extraídos por la aplicación OCR, se espera dar otra dimensión al estudio morfológico realizado a través de la clasificación supervisada. Con los nuevos parámetros obtenemos automáticamente información sobre el tiempo, localización, velocidad, dirección y configuración del sonar, en los eventos de pesca y se cree que esta información es realmente vital para ayudar en la correcta detección del atún en las imágenes de sonar. Por lo tanto, se espera que el estudio llevado a cabo sirva de ayuda para avanzar en el desarrollo de una evaluación directa para el atún rojo en el Golfo de Vizcaya.

Capítulo 5

Líneas futuras

Un primer posible estudio a desarrollar en el futuro es la realización de una clasificación supervisada similar a la realizada en esta propuesta, pero empleando los parámetros extraídos con la aplicación OCR a fin de comparar los resultados obtenidos y el efecto de las nuevas características en la clasificación.

Otro posible estudio a tener en cuenta es la utilización de los nuevos parámetros obtenidos mediante la aplicación OCR, en la clasificación de una campaña completa de atún. Mediante dichos parámetros y con la ayuda de los *logbooks* se podrían introducir diferentes restricciones en el modelo y estudiar los resultados.

Respecto a la aplicación OCR se deberían realizar alguna serie de mejoras:

- En primer lugar, se debe solucionar el problema del primer paso donde el cálculo de las coordenadas imagen de las zonas de estudio se realiza mediante un programa de Matlab. Se debe desarrollar la correspondiente herramienta GUI (interfaz gráfica de usuario) en R para las delimitaciones de las zonas a estudiar. En relación a los GUI, la clasificación supervisada de los caracteres en la aplicación OCR se realiza una vez introducido un vector de caracteres. Este paso también se debería modificar para poder hacerlo de una manera más visual y cómoda mediante una herramienta de GUI en R.

- La aplicación OCR presentada debería ser verificada con otro tipo de imágenes: imágenes procedentes de otro dispositivo de sonar, de otra configuración del sonar, de otra calidad de imagen, de otra flota pesquera, etc. De este modo, la aplicación quedaría totalmente validada para su uso.

- Una vez que se realice dicho experimento se podría plantear la posibilidad de generar un paquete específico en el software estadístico de R para técnicas de OCR, y para imágenes provenientes de diferentes tipos de dispositivos.

Otro posible avance de los aspectos técnicos a desarrollar en el futuro próximo son la detección automática o *tracking* temporal del atún. En la actualidad se han desarrollado varias técnicas para el tracking en imágenes, una de ellas es el modelo *Multi-hypothesis Tracking,* el cual se podría emplear para obtener las secuencias de imágenes, que cumplen las condiciones de correspondencia de trayectorias entre las regiones segmentadas de imágenes consecutivas.

Si utilizamos dichas secuencias, las restricciones introducidas gracias a los nuevos parámetros obtenidos mediante la técnica OCR y con la ayuda indispensable de los *logbooks*, se dispondría de un conjunto de datos mediante el cual se podría desarrollar un método automático de análisis de imágenes de atún a partir del cual analizar una serie completa de imágenes correspondientes a una campaña completa.

Se cree que la disponibilidad de este conjunto de datos podría ayudar en la evaluación del comportamiento de las regiones estudiadas dentro de las imágenes de sonar y, de este modo, poder distinguir bancos de peces de diferentes especies.

Anexos

Detecting presence-absence of bluefin tuna by automated analysis of long range sonar signals for fishing vessels

Jon Uranga[1], Haritz Arrizabalaga[1], Guillermo Boyra[1], Maria Carmen Hernandez[2], Nicolas Goñi[1], Igor Arregui[1].

1 AZTI-Tecnalia. Herrera Kaia, Portu-aldea z/g. 20110, Pasaia, Gipuzkoa (Spain).

2 UPV-EHU. University of the Basque Country.

INTRODUCTION

PROBLEM DESCRIPTION

BINARY PROBLEM ->TUNA/NO TUNA

- A new image pre-processing methodology for automated analysis in tuna long-range sonar signals is presented.

DATABASE DESCRIPTION

Positive examples (tuna):1497
Negative examples (no-tuna): 21004
ratio: 1/14.03

Slightly unbalanced DB.

20 attributes per example

STUDY AREA & TECHNIQUES

- The artisanal fleet targets juvenile tuna every summer using live bait fishing techniques.

Albacore (*Thunnus alalunga*)

Bluefin (*Thunnus thynnus*)

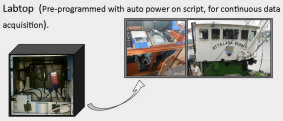
MATERIALS. SONAR EQUIPMENT

MAQ Sonar 90 kHz. Analogic sonar for the commercial fishing industry

+

Azti Tecnalia data acquisition system for the Basque fishing fleet

- Port 400Mhz Video Splitter, External VGA Capture Device, Labtop (Pre-programmed with auto power on script, for continuous data acquisition).



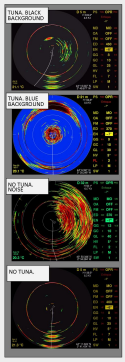
OBJECTIVE

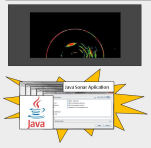
- To assess the possibility of getting acoustic-based abundance indices for bluefin tunas.
- Several uncertainties regarding the abundance of Atlantic bluefin tuna raised the need to develop fishery-independent abundance estimates.
- Comparative study of the data mining techniques over the semi-automated sonar signal data.

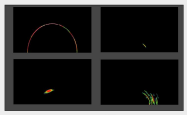
METHODS

1. Semi-automated image pre-processing (SAIPP)

1.IMAGE CAPTURE → 2. IMAGE PRE-PROCESSING → 3. SEGMENTATION







4. CHARACTERISTICS EXTRACTION

DB with 20 Morphological Attributes
Area, X, Y, Perimeter, BX, BY, Width, Height, Major, Minor, Angle, Circularity, Feret, FeretX, FeretY, FeretAngle, MinFeret, AR, Round, Solidity.

5. EXPERTS ON SEGMENT LABELING

Labeling based on the logbooks & scientific observers

COMPLETE DB. BD with 20 Attributes + label (tuna/no-tuna)

2. Data Base Filtering

- COMPLETE DB is slightly unbalanced. To manage this, we apply oversampling (SMOTE) and subsampling (SPREAD) methods.

COMPLETE DB

SMOTE Filter → SMOTE DB

SPREAD Filter → SPREAD DB

New Ratios

	Tuna instances	No Tuna instances	Ratio
COMPLETE DB	1497	21004	1/14.03
SMOTE DB	2994	21004	1/7.02
SPREAD DB	1497	10502	1/7.01

3. Classification

Image selection

SAIPP

Complete DB | SMOTE DB | SPREAD DB

CLASSIFICATION METHODS (CM)
Random Forest, Support Vector Machine, MultiLayer Perceptron, IBK and J48

RESULTS EVALUATION based on:
Kappa, Sensitivity, Specificity, AUC

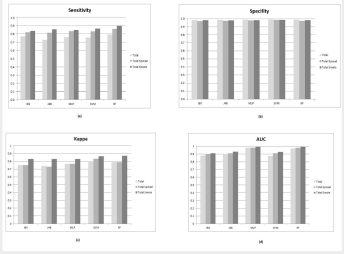
COMPARISON

RESULTS & DISCUSSION

Supervised Classification

- Mean values of Kappa index, AUC (Area under ROC), sensibility and specificity have been computed for 30 runs of cross-validation with 10 folds to avoid over fitting and to find stable results.

	AUC (d)			Sensitivity (a)			Specifity (b)			Kappa (c)		
	TOTAL	SPREAD	SMOTE	TOTAL	SPREAD	SMOTE	TOTAL	SPREAD	SMOTE	TOTAL	SPREAD	SMOTE
RF	0.97	0.98	0.99	0.79	0.86	0.9	0.98	0.98	0.98	0.79	0.79	0.87
SVM	0.87	0.91	0.93	0.76	0.83	0.87	0.99	0.98	0.99	0.79	0.83	0.87
IBK	0.88	0.9	0.91	0.77	0.82	0.84	0.99	0.97	0.98	0.75	0.75	0.83
MLP	0.98	0.98	0.99	0.76	0.83	0.85	0.99	0.98	0.98	0.77	0.77	0.83
J48	0.9	0.91	0.93	0.73	0.82	0.86	0.99	0.97	0.98	0.74	0.73	0.83



Classification

- Mean values for all experiments are computed, to compare the quality of results. As shown in the previous table, the method yields a good performance.

Data Base Filtering

- Filtered databases obtained better performance, specially with the SMOTE DB.

SAIPP

- Overall performance of the comparison study validates the SAIPP for morphological analysis.

CONCLUSIONS & FUTURE LINES

Main ideas to take home

A new supervised classification methodology for automated analysis in tuna long-range sonar signals is presented.

High specificity and lower sensibility than expected is observed in the performance of the algorithms. This is due to the morphological similarity of the tuna and no-tuna images.

The method provides good performance regarding the quality of classification.

For future efforts, to solve the problem of morphological similarities among tuna and no-tuna images, should be considered a study of temporality.

Future Lines

An OCR (Optical Character Recognition) based methodology is under construction. The aim is to extract de parameters related to the vessel situation (speed, localization & bearing) and the sonar setup (range, beam width, tilt & gain).

Regarding to the temporality study, some automated tracking method should be developed. With the tracks, restrictions established by the parameters obtained with the OCR application and logbook/observers data, improvement of the automated analysis of images for fishing vessels is expected.

References

- Ian H. Witten; Eibe Frank, Mark A. Hall (2011). *Data Mining: Practical machine learning tools and techniques, 3rd Edition*. Morgan Kaufmann, San Francisco. - R.C. Gonzalez and R.E. Woods (2008). *Digital Image Processing, 3rd Edition*. Prentice Hall, New Jersey.

ACKNOWLEDGMENTS

Project details: Assessment of the applicability of direct methods for estimating abundance of bluefin tuna (Zuzenatun), Exp.: GV 351NPVA00062.

Funding sources: Eusko Jaurlaritza - Gobierno Vasco - Basque Goverment, Dept. of Agriculture, Fisheries and Food, Ministry of Agriculture and Fisheries Development, Fisheries and Aquaculture Management. Jon Uranga was funded by a PhD grant of this Dept.

UPV EHU

azti tecnalia

EUSKO JAURLARITZA GOBIERNO VASCO

ICES CIEM

Fig. 5.1: Póster a presentar en la *ICES Annual Science Conference*.

Bibliografía

[1] Omri Allouche, Asaf Tsoar, and Ronen Kadmon. Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (tss). *Journal of Applied Ecology*, 43(6):1223–1232, 2006.

[2] D.G. Altman. *Practical Statistics For Medical Research*. Chapman and Hall/CRC Texts in Statistical Science Series. Chapman and Hall, 1991.

[3] C.M. Bishop and G. Hinton. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[4] Matias Bordese and Walter Alini. *biOps: Image processing and analysis*, 2013. R package version 0.2.2.

[5] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[6] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, Volume 2, Issue 2:pp 121–167, 1998.

[7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[8] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

[9] Vladimir Vapnik Corinna Cortes. Support-vector networks. *Machine Learning*, Volume 20, Issue 3:pp 273–297, 1995.

[10] Alan H. Fielding and John F. Bell. A review of methods for the assessment of prediction errors in conservation presence/absence models. *Environmental Conservation*, 24:38–49, 2 1997.

[11] E. Fix and J.L. Hodges. *Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties*. USAF School of Aviation Medicine, 1951.

[12] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, and Tony Cooper. *caret: Classification and Regression Training*, 2013. R package version 5.16-24.

[13] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[14] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic(roc) curve. *Radiology*, 142:29–36, 1982.

[15] S.S. Haykin. *Neural Networks: Comprehensive Foundation*. Prentice Hall, 1999.

[16] David MacLennan. John Simmonds. *Fisheries Acoustics*. Blackwell P, 1992.

[17] J. Lemon. Plotrix: a package in the red light district of r. *R-News*, 6(4):8–12, 2006.

[18] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2012. R package version 1.6-1.

[19] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):62–66, 1979.

[20] Gregoire Pau, Andrzej Oles, Mike Smith, Oleg Sklyar, and Wolfgang Huber. *EBImage: Image processing toolbox for R.* R package version 4.2.1.

[21] Tony Plate and Richard Heiberger. *abind: Combine multi-dimensional arrays*, 2011. R package version 1.4-0.

[22] J. R. Quinlan. Improved use of continuous attributes in c4.5. *J. Artif. Int. Res.*, 4(1):77–90, March 1996.

[23] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Mateo, CA, San Francisco, CA, USA, 1993.

[24] Jacob van Etten Robert J. Hijmans. *raster: Geographic data analysis and modeling*, 2013. R package version 2.1-37.

[25] E. Rodríguez-Marín, H. Arrizabalaga, M. Ortiz, C. Rodríguez-Cabello, G. Moreno, and L.T. Kell. Standardization of bluefin tuna, *Thunnus thynnus*, catch per unit effort in the baitboat fishery of the bay of biscay (Eastern Atlantic). *ICES Journal of Marine Science: Journal du Conseil*, 60(6):1216–1231, January 2003.

[26] Deepayan Sarkar. *Lattice: Multivariate Data Visualization with R.* Springer, New York, 2008. ISBN 978-0-387-75968-5.

[27] J. A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240:1285–1293, 1988.

[28] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[29] Ari Urkullu. Análisis automático de señales de atún obtenidas mediante sonar de largo alcance a bordo de buques pesqueros. Master's thesis, Universidad del País Vasco / Euskal Herriko Unibertsitatea, 2011.

[30] I.H. Witten, E. Frank, and M.A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.