**UNIVERSIDAD CARLOS III DE MADRID**

ESCUELA POLITÉCNICA SUPERIOR


PROYECTO DE FIN DE CARRERA

*Ingeniería de Telecomunicación*

# Análisis de las normas internacionales de firmas manuscritas ISO/IEC 19794-7 y 19794-11

*Autor: Roberto Pizarro Santos*

*Tutor: Óscar Miguel Hurtado*

# AGRADECIMIENTOS

Este proyecto significa el final de una increíble etapa de mi vida, y el inicio de otra completamente diferente y misteriosa. En estos momentos el sentimiento que me rodea es de entusiasmo y satisfacción. Quedaron atrás largos días de estudio, eternos períodos de exámenes y la ilusión del día a día de las clases. Sólo quedan los buenos momentos en el recuerdo, muchos amigos e historias irrepetibles.

En primer lugar quiero hacer mención a mis padres, Rosa María y Félix, y mi hermano pequeño, Javier, por su apoyo incondicional, total confianza y por ayudarme a convertirme en quien soy hoy por hoy, nunca podré agradecerles lo suficiente todo lo que han hecho por mí ni tampoco tengo palabras para expresar lo mucho que significan y participan en mi vida.

Recordar también al resto de mi familia, siempre a mi lado, dándome todo su cariño y apoyo, interesándose por mi evolución como estudiante y también por creer en mí.

El siguiente punto lo dedico a mis amigos y compañeros, tanto los que ya lo eran antes de comenzar mis estudios, como aquellos que he conocido en el transcurso. Con ellos he vivido momentos inolvidables tanto dentro como fuera de las aulas, que quedarán marcados para mí por siempre. Muchas gracias a todos por vuestra ayuda, comprensión y apoyo, que muchas veces era complicado debido a las tensiones de los exámenes, agobios por entregas de prácticas… pero que siempre terminaban con una sonrisa.

También hacer mención a los profesores, una parte importante e imprescindible de este período, mi agradecimiento por su tiempo y por sus conocimientos, además de ayudarme a crecer como persona, así como al resto de personal de la universidad porque ellos también forman parte de estos años

No quiero olvidarme de todos aquellos no mencionados con anterioridad pero que de una forma u otra han formado parte de mi vida en estos años, sin ellos nada hubiera sido igual.

Por último, mi reconocimiento a la otra parte del equipo que ha desarrollado este proyecto, mi tutor Óscar, por ofrecerme esta oportunidad y ayudarme a hacerlo una realidad.

A todos, siempre os guardaré en mi corazón.

Roberto

# ÍNDICE

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# 1 Introducción

## *1.1 Contexto*

Mediante un dispositivo de adquisición de datos, sobre el que el usuario realiza una firma manuscrita, se permite el reconocimiento del usuario de manera segura analizando los datos digitalizados de la escritura y comprobando de acuerdo a una base de datos en la que se comparan las firmas contenidas en ella con la introducida por el usuario.

Para ello, se hace necesaria la creación de normas internacionales que fomenten la interoperabilidad entre sistemas, de forma que se logre el entendimiento entre los mismos con el consiguiente acceso a los datos en cualquier lugar sin problemas. Las normas de datos que se analizarán en este documento, permiten conseguir de una manera sencilla la interoperabilidad en sistemas de verificación de identidad basados en la firma manuscrita con el proyecto 19794, que define un formato de intercambio de datos biométricos.

Su implantación en la vida cotidiana todavía no se ha convertido en un hecho, pero se están siguiendo las directrices necesarias y el desarrollo de aplicaciones que en un futuro cercano permitan su uso generalizado en entornos comerciales. Actualmente se están creando nuevos estándares y normas, que se desarrollarán y comprobará su utilidad y prestaciones, para poder llegar a una solución que cumpla los requisitos que se precisan para ofrecer los servicios demandados

Los dispositivos utilizados, tales como sistemas embebidos o smart cards, tienen características de almacenamiento limitadas, además la velocidad de transmisión de datos en éstos puede ser determinante. De este modo surge la necesidad de comprimir los datos y para ello se utilizan algoritmos de compresión sin pérdidas de información. En función del tipo de datos que se precise guardar, serán de aplicación unos algoritmos u otros, para lo cual se deberá realizar un estudio de dichos algoritmos y los datos a almacenar.

En el presente proyecto se tratarán técnicas para hacer frente a las necesidades expuestas, interoperabilidad y compresión de datos, y se realizarán análisis comparativos para ofrecer una visión de las mejores herramientas en cada caso. El estudio de estándares de almacenamiento de firmas manuscritas y de algoritmos de compresión sin pérdidas de información para aplicar a los datos generados serán los objetos de estudio de este documento

## *1.2 Motivación*

El inicio de este proyecto se lleva a cabo con las siguientes motivaciones:

- Estudio de la norma de firma publicada 19794-7, en sus versiones Completa y Compacta
- Estudio de la nueva propuesta de norma 19794-11
- Implementaciones de las normas para su evaluación, estudio de los tamaños, ratios de compresión y rendimiento que ofrecen en los algoritmos de firma
- Comprensión en su totalidad de la nueva norma 19794-11
- Trabajo con estándares y normas que se están desarrollando en la actualidad, de modo que al encontrarse en período de pruebas, se pueden realizar estudios que permitan conocer si las decisiones tomadas a la hora de su creación han sido correctas, y en caso de que no sean apropiadas, aportar otras posibles soluciones

## *1.3 Objetivos*

Al comienzo de este proyecto se plantean una serie de objetivos primordiales para conseguir la obtención de los resultados esperados a priori:

- El objetivo principal de este documento es el estudio cualitativo de las normas 19794-7, tanto en sus formatos completo como compacto y 19794-11.
- Implementación en código C de las normas mencionadas
- Conversión de la base de datos MCyT y SVC2004 a los formatos marcados por los distintos estándares.
- A partir del análisis de las normas y su implementación, se realizará un estudio comparativo entre ellas en base al tamaño de los ficheros obtenidos.
- Aplicación de algoritmos de compresión sin pérdidas de información a los ficheros con los datos generados.
- A partir de la compresión realizada, se establece un análisis comparativo entre los diferentes formatos de firmas descritos y los distintos algoritmos de compresión estudiados, en base a la reducción del tamaño de las muestras en cada caso
- Estudio de las normas para probar su funcionalidad para en un futuro llevarlas a cabo en aplicaciones comerciales.

## 1.4 Organización del documento

El trabajo realizado y que se expone en el presente documento se divide en dos grandes bloques, por un lado el estudio e implementación de las normas 19794-7 y 19794-11 y por otro la aplicación de algoritmos de compresión sin pérdidas a dichos algoritmos.

Se comenzará con una introducción a firmas manuscritas y biometría, ya que el trabajo expuesto trata sobre biometría basada en firmas manuscritas. Se comentan unas breves pinceladas al origen de cada uno de ellos, su historia, características, aplicaciones y su utilidad al combinarlos.

Seguidamente se comienza con la exposición de las normas a tratar, basadas en los estándares ISO/IEC, encuadrados en el proyecto 19794, explicando el funcionamiento de ellos, su estructura y composición, así como la base de datos MCyT, que contiene la información de los usuarios firmantes, clave en el proceso, ya que será ésta la que se convierta a los distintos formatos que se comentarán posteriormente.

Una vez puestos en situación, con los conceptos teóricos estudiados y las ideas claras de lo que se pretende realizar, se procede a la implementación de las normas, más concretamente, la conversión de la base de datos MCyT a lo dictado en las distintas normas, guardado en las estructuras definidas y almacenado en nuevos ficheros. Las normas que se analizan son 19794-7, tanto en su versión completa como compacta y 19794-11, las dos últimas se crearán a partir de la primera, mediante compresión de la misma.

En este punto se finaliza el primer gran bloque. A continuación se comienza con la fase de compresión. Lo primero será dar unas nociones sobre compresión de datos, distinguiendo entre compresión con pérdidas y sin pérdidas de información, analizando sus diferencias y exponiendo los motivos por los que se elige la compresión sin pérdidas para aplicar a las normas implementadas. Los algoritmos elegidos debido a su buen funcionamiento para los datos a tratar son RLE (Run-Length Encoding), LZSS (Lempel-Ziv-Storer-Szymanski) y LZW (Lempel-Ziv-Welch), de los cuales se detallará su funcionamiento.

Para comprobar el funcionamiento de los anteriores, se construyen varias versiones de las norma 19794-7 almacenada previamente, ya que solamente se aplica compresión para las dos modalidades de ésta, diferenciadas en el orden en el que se almacenan los datos. La versión denominada V0, guarda los datos conforme se define en la norma, V1 agrupa los canales de información y V2 igual que la anterior pero guarda diferencias de muestras en lugar de los datos en sí. De esta forma se podrá apreciar la eficiencia en cada uno de los casos, obteniendo conclusiones para determinar la mejor opción.

La parte final del proyecto consiste en la exposición de los resultados obtenidos para cada uno de los casos que se presentan, mediante el uso de tablas y gráficos ilustrativos, para poder obtener conclusiones al respecto y realizar análisis comparativos entre las diferentes posibilidades.

Tan sólo quedaría hacer una reflexión sobre las conclusiones que se obtienen del trabajo descrito, así como de los futuros puntos a tratar, mejoras a realizar y camino a seguir para su desarrollo.

Para finalizar el documento, se enumeran las fuentes bibliográficas en las que se apoya este trabajo, de las cuales se obtiene información de apoyo y que se han consultado a lo largo de las tareas realizadas.

# 2 Introducción Biometría / Firma Manuscrita

## *2.1 Biometría*

La biometría es el estudio de métodos automáticos para el reconocimiento o verificación de identidades basados en uno o más rasgos conductuales o físicos característicos. El término se deriva de las palabras griegas "bios" (vida) y "metron" (medida) [1].

Dichos métodos automáticos se basan en la captura de rasgos biométricos, que pueden ser tanto dinámicos como estáticos y su análisis a través de herramientas estadísticas.

Con estos rasgos biométricos se evitan problemas que existen con los sistemas de autenticación por tarjetas de identificación y contraseñas, en los que es necesario llevar encima un objeto o recordar una clave, con el riesgo de pérdida u olvido y, por tanto, ofrecen una gran versatilidad.

### 2.1.1 Historia

La biometría era utilizada en China desde al menos el siglo XIV. Joao de Barros, explorador y escritor, cuenta que los comerciantes chinos estampaban las impresiones y las huellas de la palma de las manos de los niños en papel con tinta. Los comerciantes hacían esto como método para distinguir entre los niños jóvenes.

En Occidente no comenzó a usarse hasta finales del siglo XIX. Alphonse Bertillon, jefe del departamento fotográfico de la Policía de París, desarrolló el sistema antropométrico (también conocido más tarde como Bertillonage) en 1883. Éste fue el primer sistema preciso ampliamente utilizado científicamente para identificar a criminales y convirtió a la biométrica en un campo de estudio. Funcionaba midiendo de forma precisa ciertas longitudes y anchuras de la cabeza y del cuerpo, así como registrando marcas individuales como tatuajes y cicatrices. Con posterioridad se comenzó a utilizar la huella dactilar (esencialmente el mismo sistema visto en China cientos de años antes) [2].

En estos últimos años la biométrica ha crecido desde usar simplemente la huella dactilar, a emplear muchos métodos distintos, como pueden ser rasgos físicos (iris, cara, geometría de la mano, etc.) y/o de comportamiento (forma de andar, firma manuscrita, voz, etc.). Las aplicaciones de la biometría también han aumentado, algunas de ellas pueden ser el control de acceso, control de presencia, lectores, votación, etc...

### 2.1.2 Sistemas biométricos

Se entiende por sistema de identificación/verificación biométrico a un sistema que toma sus decisiones de reconocimiento de manera automática teniendo en cuenta una característica/rasgo personal. A esta característica se la denomina indicador biométrico y debe cumplir los siguientes principios [3]:

1. Universalidad: cualquier persona tiene que poseer dicha característica
2. Unicidad: la existencia de dos personas con una característica idéntica tiene una probabilidad muy pequeña
3. Permanencia: no cambia en el tiempo
4. Cuantificación: puede ser medida en forma cuantitativa.

El rendimiento de una medida biométrica se define generalmente en términos de:

- Tasa de falso positivo (False Acceptance Rate o FAR): indica la probabilidad de la aceptación de una muestra errónea, es decir que se dé como válida una muestra que realmente no lo es.

- Tasa de falso negativo (False NonMatch Rate o FNMR, también False Rejection Rate o FRR): indicar la probabilidad del rechazo de una muestra válida, es decirque se dé como errónea una muestra que realmente no lo es [1].

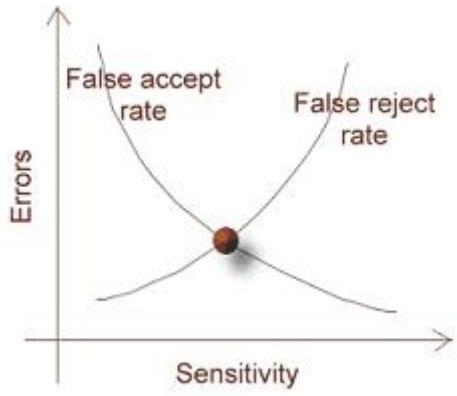



**Figura 1 - Rendimiento de una medida biométrica**

Una de las medidas más comunes de los sistemas biométricos reales es la tasa de error igual (Equal Error Rate o EER), también conocida como la tasa de error de cruce (Cross-over Error Rate o CER). Cuanto más bajo es el EER o el CER, se considera que el sistema es más exacto.

Los sistemas biométricos se pueden clasificar en base a la característica que se estudia para tomar la decisión de identificación de una persona. En base a esto se realiza la división en estáticos y dinámicos:

- Biometría estática: medición de las características fisiológicas de un individuo. Dichas características pueden ser:

    - Huellas dactilares: análisis de la huella dactilar de una persona, es la técnica más extendida debido en gran medida a su estabilidad con el tiempo, la edad y su sencillez de adquisición

    - Análisis de iris y retina: basado en las características de las partes del ojo denominadas iris ocular y retina. Su estudio se basa en la textura del iris ocular y los vasos sanguíneos de la retina.

    - Reconocimiento facial: Se basa en la comparación de diferentes características faciales con respecto a una base de datos o una imagen. su gran inconveniente es el cambio rostro a lo largo del tiempo.

    - Geometría de la mano: se estudian diversos parámetros morfológicos de la mano (o el dedo) del individuo, tales como anchuras, alturas…

- Biometría dinámica: medición de los rasgos de comportamiento de un individuo. Estos rasgos pueden ser:

    - Firma manuscrita: comparación de medidas realizadas sobre la firma escrita por el individuo con respecto los datos almacenados en una base de datos previamente introducidos de la firma verdadera. Tales medidas pueden ser la presión que se ejerce con el bolígrafo, diferentes ángulos de inclinación…

    - Reconocimiento de voz: consiste en procesar la señal de voz y extraer parámetros característicos de ésta. De esta forma para identificar a un individuo se hace mediante la comparación de dichos parámetros con respecto a una base de datos con los parámetros del individuo previamente almacenados.

    - Dinámica de teclado: identificación de una persona por la forma de escribir a máquina u ordenador. Se maneja la hipótesis de que el ritmo de tecleo es característico de una persona.

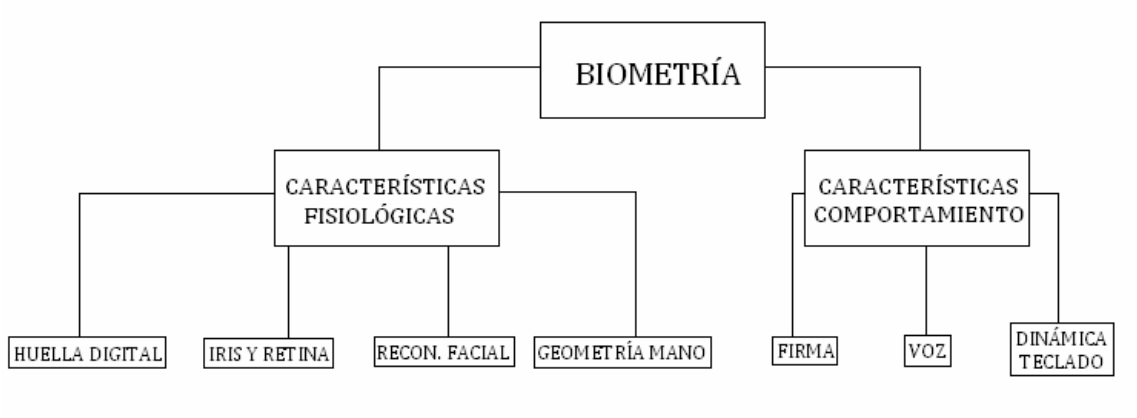



**Figura 2 - Características biométricas**

Una comparativa entre los diferentes sistemas biométricos y sus características sería la siguiente: [1]

| | Ojo (Iris) | Ojo (Retina) | Huellas dactilares | Geometría de la mano | Escritura y firma | Voz | Cara |
|---|---|---|---|---|---|---|---|
| **Fiabilidad** | Muy alta | Muy alta | Alta | Alta | Media | Alta | Alta |
| **Facilidad uso** | Media | Baja | Alta | Alta | Alta | Alta | Alta |
| **Prevención ataques** | Muy alta | Muy alta | Alta | Alta | Media | Media | Media |
| **Aceptación** | Media | Baja | Alta | Alta | Muy alta | Alta | Muy alta |
| **Estabilidad** | Alta | Alta | Alta | Media | Baja | Media | Media |

**Tabla 1 - Comparativa de sistemas biométricos**

El uso de un sistema u otro depende de la aplicación para la que se pretenda implantar, el entorno, los objetivos propuestos, etc… Por lo tanto no hay ningún sistema mejor que otro, irá en función a las necesidades de la aplicación.

### 2.1.3 Estándares asociados a tecnologías biométricas

En los últimos años se ha notado una preocupación creciente por las organizaciones regulatorias respecto a elaborar estándares relativos al uso de técnicas biométricas. Esta preocupación es reflejo del creciente interés industrial por este ámbito tecnológico, y a los múltiples beneficios que su uso aporta. No obstante, aún la estandarización continúa siendo deficiente y como resultado de ello, los proveedores de soluciones biométricas continúan suministrando interfaces de software propietarios para sus productos, lo que dificulta a las empresas el cambio de producto o vendedor.

A nivel mundial el principal organismo que coordina las actividades de estandarización biométrica es el Sub-Comité 37 (SC37) del Joint Technical Committee on Information Technology (ISO/IEC JTC1), del International Organization for Standarization (ISO) y el International Electrotechnical Commission (IEC).

En Estados Unidos desempeñan un papel similar el Comité Técnico M1 del INCITS (InterNational Committee for Information Technology Standards), el National Institute of Standards and Technology (NIST) y el American National Standards Institute (ANSI).

Existen además otros organismos no gubernamentales impulsando iniciativas en materias biométricas tales como: Biometrics Consortium, International Biometrics Groups y BioAPI. Este último se estableció en Estados Unidos en 1998 compuesto por las empresas Bioscrypt, Compaq, Iridiam, Infineon, NIST, Saflink y Unisis. El Consorcio BioAPI desarrolló conjuntamente con otros consorcios y asociaciones, un estándar que promoviera la conexión entre los dispositivos biométricos y los diferentes tipos de programas de aplicación, además de promover el crecimiento de los mercados biométricos [3].

Algunos de los estándares más importantes son [1]:

- Estándar ANSI X.9.84: creado en 2001, por la ANSI (American National Standards Institute) y actualizado en 2003, define las condiciones de los sistemas biométricos para la industria de servicios financieros haciendo referencia a la transmisión y almacenamiento seguro de información biométrica, y a la seguridad del hardware asociado.

- Estándar ANSI / INCITS 358: creado en 2002 por ANSI y BioApi Consortium, presenta una interfaz de programación de aplicación que garantiza que los productos y sistemas que cumplen este estándar son interoperables entre sí.

- Estándar NISTIR 6529: también conocido como CBEFF (Common Biometric Exchange File Format) es un estándar creado en 1999 por NIST y Biometrics Consortium que propone un formato estandarizado (estructura lógica de archivos de datos) para el intercambio de información biométrica.

- Estándar ANSI 378: creado en 2004 por la ANSI, establece criterios para representar e intercambiar la información de las huellas dactilares a través del

uso de minucias. El propósito de esta norma es que un sistema biométrico dactilar pueda realizar procesos de verificación de identidad e identificación, empleando información biométrica proveniente de otros sistemas.

- Proyecto ISO/IEC 19794: define un formato de intercambio de datos biométricos. Por ejemplo mencionar como parte de este proyecto el estándar ISO 19794-2, creado en 2005 por la ISO/IEC con propósitos similares a la norma ANSI 378, respecto a la que guarda mucha similitud o los estándares 19794-7 y 19794-11 sobre biometría en firmas manuscritas.

- Estándar PIV-071006: creado en 2006 por el NIST y el FBI en el contexto de la norma FIPS 201 del gobierno de EE.UU, establece los criterios de calidad de imagen que deben cumplir los lectores de huellas dactilares para poder ser usados en procesos de verificación de identidad en agencias federales.

### 2.1.4 Aplicaciones de la biometría

Algunas de las aplicaciones de la biometría son [3]:

- *Autenticación biométrica de rostro*: identificación de personas por medio de la biometría facial utilizando foto o vídeo

- *Autenticación de documentos*: validación de documentos oficiales por medio de equipos especiales para la creación de expedientes electrónicos de clientes o empleados

- *Soluciones biométricas de huella digital*: tecnologías de hardware y software para reconocimiento de huella digital en ambientes transaccionales, poblacionales y civiles

- *Firma autógrafa digital*: capturar la firma autógrafa digital para procesos de negocios, transacciones electrónicas y sistemas de reconocimiento biométrico de firma digital

- *Autenticación biométrica avanzada de voz*: implementación segura de aplicaciones web y comunicaciones

- *Terminal móvil biométrico*: dispositivos móviles para transacciones y procesos autenticados con comunicación inalámbrica y celular

- *Contador de personas*: generación de estadísticas y métricas de sucursales, tiendas, oficinas… por medio del conteo exacto de personas

- *Control de acceso físico y lógico*: sistemas basados en biometría para administrar el ingreso a instalaciones o redes de cómputo

## *2.2 Firma manuscrita*

Según el Diccionario de la Real Academia, la firma es el nombre y apellido, o título, que una persona escribe de su propia mano en un documento, para darle autenticidad o para expresar que aprueba su contenido.

La denominación de “manuscrita” es debida a que la firma debe plasmarse en persona, con puño y letra del propio firmante. De esta manera se establece un nexo de unión entre firma y firmante único

### 2.2.1 Historia

En Roma, existía la Manufirmatio, que consistía en una ceremonia en la que el autor leía su documento y posteriormente lo colocaba extendido sobre la mesa del escribano para pasar la mano abierta sobre el pergamino en actitud de jurar, pero sin hacerlo, se estampaba el nombre, signo, o una o tres cruces, por el autor o el funcionario en su nombre, haciéndolo seguidamente los testigos. Más que un requisito, la Manufirmatio era en sí misma parte del espectáculo solemne en que se realizaba el acto

En la Edad Media, se inscribía una cruz a la que se le añadían diversas letras y rasgos. Estos signos se utilizaban como firma. Debido a que no sabían leer ni escribir, los nobles remplazaron esta práctica con el uso de sellos [4].

En 1917 Ludwig Klages escribió un libro: “Escritura y carácter” donde hablaba de la presión gráfica de la escritura. Firme o temblorosa, la firma tiene su ciencia llamada grafología que, apelando al estudio material de la letra, se vuelve más específica en la psicografía que pretende descifrar o aventurar no sólo el rasgo de carácter sino el destino de una persona bajo ese género "científico" y predictivo llamado test. La firma con su soporte en la letra funciona como rasgo de personalidad [5].

La diferenciación entre firmas y signos hizo que se empezase a entender que aquéllas eran, más que simples signos, la inscripción manuscrita del nombre o de los apellidos. La firma fue adquiriendo importancia y uso, que con el transcurso del tiempo se fue consagrando como un símbolo de identificación y de enlace entre el autor de lo escrito o estampado y su persona.

### 2.2.2 Características

La firma manuscrita tiene las siguientes características:

- *Identificativa*: Sirve para identificar quién es el autor del documento.

- *Declarativa*: Significa la asunción del contenido del documento por el autor de la firma. Representa la voluntad de consentimiento.

- *Probatoria*: Permite identificar si el autor de la firma es efectivamente aquél que ha sido identificado como tal en el acto de la propia firma.

### 2.2.3 Elementos

- *Elementos formales*: Son aquellos elementos materiales de la firma que están en relación con los procedimientos utilizados para firmar y el grafismo mismo de la misma. La firma se presenta como un signo distintivo y personal, ya que debe ser puesta de puño y letra del firmante.

- *El animas signand:* Es el elemento intencional o intelectual de la firma. Consiste en la voluntad de asumir el contenido, como señala Larrieu [4].

- *Elementos funcionales:* Tomando la noción de firma como el signo o conjunto de signos, podemos distinguir una doble función:
    - Identificadora: La firma asegura la relación jurídica entre el acto firmado y la persona que lo ha firmado. La firma manuscrita expresa la identidad, aceptación y autoría del firmante.
    - Autenticación: El autor del acto expresa su consentimiento y hace propio el mensaje

### 2.2.4 Importancia

La importancia de la firma manuscrita reside en que es personal, intransferible, identificativa y que garantiza que se ha realizado por la persona correspondiente. Cabe tener en cuenta que la firma debe ser siempre la misma y coincidir con la plasmada en el documento nacional de identidad o cualquier documento de identificación con validez legal (como el pasaporte), ya que es donde aparece registrada y la que permite crear la asociación firma – firmante para comprobar su autenticidad.

En la época actual, forma parte del día a día de nuestras vidas, y es muy común su uso para muy diversas aplicaciones, de lo cual se extrae que es imprescindible sentirse identificado con una firma y que ésta sea siempre la misma y a ser posible muy difícil de imitar.

### 2.2.5 Aplicaciones

Las aplicaciones de la firma manuscrita son muchas y variopintas, extendiéndose a todos los sectores y siendo un elemento clave y diario de nuestras vidas. Algunas de ellas se citan a continuación:

- Operaciones bancarias
- Certificaciones
- Contratos (de trabajo, de arrendamiento, de alquiler, de venta…)
- Compras en supermercados y restaurantes mediante tarjeta de crédito
- Autorizaciones y consentimientos
- Registro (de entrada/salida del puesto de trabajo, de acceso a instalaciones…)

### 2.2.6 Interés de la industria

El desarrollo de los sistemas informáticos y la expansión de Internet y las nuevas tecnologías en el mundo empresarial hace necesario un nivel de seguridad acorde a los nuevos tiempos, de modo que se deben buscar soluciones a los problemas generados por el mal uso de los sistemas actuales. Una vez dicho esto, un problema que puede presentarse es la suplantación de la persona y por tanto el desarrollo de acciones no deseadas y dañinas para el usuario. Por todo ello se crea la necesidad de que la identificación personal sea segura y se tenga la certeza de que siempre es así. Por medio de la firma manuscrita se garantiza la identificación única y segura de la persona, teniendo la certeza de que al observar la firma en un documento se tiene la tranquilidad de su veracidad.

Todo lo anterior ha dado lugar a la creación de estándares internacionales para el almacenamiento y envío de muestras de firmas manuscritas de manera interoperable, recogidos en el Organismo Internacional de Estandarización (ISO), los cuales son objeto de estudio y evaluación en este documento.

### 2.2.7 Biometría en firma manuscrita

La autenticación de usuarios a través de firma manuscrita es uno de los sistemas biométricos que más se acerca al paradigma de alta aceptación y bajo coste, que se une a los beneficios que ya por sí misma presenta la tecnología de firma electrónica es el reconocimiento biométrico de rasgos caligráficos de la firma.

Con los modernos dispositivos que permiten establecer parámetros de presión, velocidad e inclinación del trazo se pueden obtener patrones que resultan de una gran eficacia.

Una de las grandes ventajas con la que cuenta es la no necesidad de equipos individuales, sino que un mismo dispositivo es válido y utilizable por todos los usuarios, lo cual implica un gran ahorro económico y facilita su despliegue. Además, el coste de los dispositivos de recogida de firmas no es alto y evita el gasto de papel.

La biometría basada en firma manuscrita se puede dividir en 2 tipos en función del tipo de información que se maneja [6]:

- Firma manuscrita dinámica (on-line): hace uso de la información instantánea proporcionada por la tableta de la firma.

  Cuenta con las siguientes ventajas:

  - Alta aceptación personal, social y legal como medio de autenticación
  - Buena adaptación a entornos móviles y de bajo control.
  - Eclosión de dispositivos móviles/portátiles con interfaz tipo puntero (TabletPC, PDA, ultra-portable PC, Móviles 3G, PocketPC…).
  - Disponible en diversas aplicaciones comerciales (mensajería express, tiendas Opencor).
  - A partir de los datos online, se pueden obtener imágenes estáticas de alta calidad.

  Inconvenientes

  - Requieren el uso de material especial (tableta, lápiz…) y la posibilidad de que varíen ciertas características ya que no es lo mismo escribir con un bolígrafo sobre papel que con uno de plástico sobre una pantalla táctil.
  - Es muy importante la velocidad de respuesta puesto que el usuario se encuentra esperando a que finalice el reconocimiento.

  Los retos que se proponen:

  - Alta variabilidad intra-clase.
  - Variabilidad temporal entre sesiones.
  - Los impostores pueden producir falsificaciones entrenadas.

  Los dispositivos utilizados pueden ser:

  - Tabletas digitalizadoras: también llamadas ePads, en las que se firma en la tableta mediante un bolígrafo especial unido a éstas.
  - Otra opción son los lápices electrónicos (electronic pencils), un dispositivo de escritura que incorpora una diminuta cámara que permite transferir lo escrito a un ordenador o teléfono móvil, generalmente vía bluetooth.

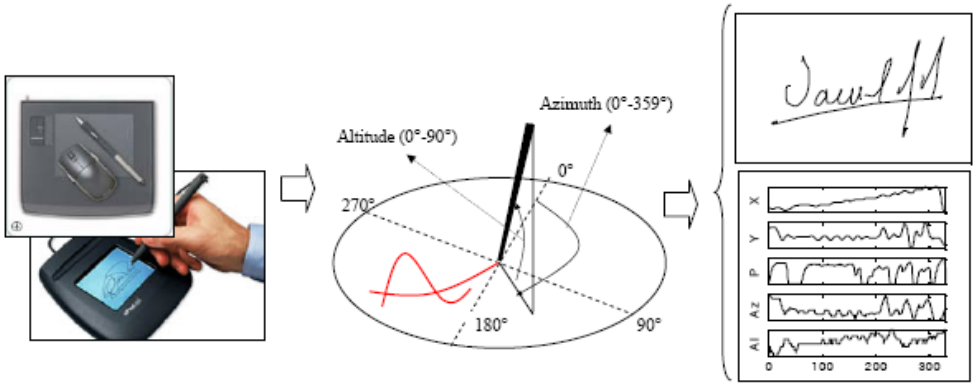



**Figura 3 - Esquema de biometría basada en firma manuscrita**

- Firma manuscrita estática (off-line): no se tiene acceso al acto de firma, sólo se dispone de su forma como imagen.

  Cuenta con las siguientes ventajas:

  - Posibilidad de aplicarse sin la presencia del individuo firmante.
  - Resulta menos costoso a nivel de equipamiento.

  Como inconvenientes:

  - Se tienen tasas de error elevadas debido a la no disposición de la información dinámica de la realización de la firma

  Los dispositivos que almacenan las firmas estáticas pueden ser por ejemplo teléfonos móviles, cámaras digitales, escáneres, en general cualquier dispositivo que permita la obtención de imágenes (fotos de las firmas) a partir de firmas escritas en papel.

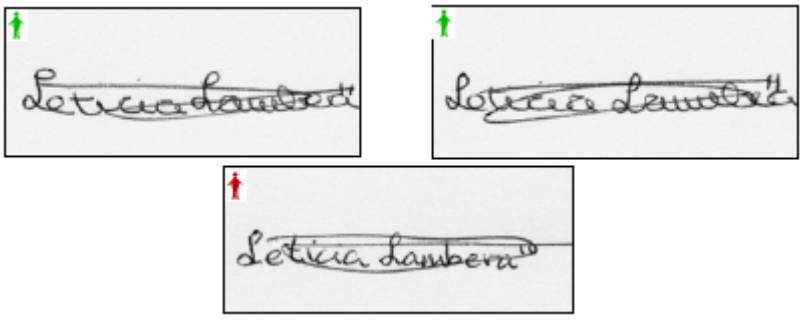


**Figura 4 - Ejemplos de firmas manuscritas**

# 3 Normas ISO/IEC

## *3.1 Introducción a ISO/IEC*

ISO(Organización Internacional para la Estandarización) e IEC (Comisión Electrotécnica Internacional) forman el sistema especializado para la estandarización internacional. Los cuerpos nacionales que son miembros de ISO o IEC participan en el desarrollo de estándares internacionales a través de comités técnicos establecidos por la respectiva organización para tratar los campos particulares de la actividad técnica. Los comités técnicos de ISO e IEC colaboran en campos de mutuo interés. También toman parte en el trabajo otras organizaciones internacionales, gubernamentales y no gubernamentales, en alianza con ISO e IEC. En el campo de la tecnología de información, ISO e IEC han establecido un Comité Técnico Conjunto, ISO/IEC JTC 1

Los Estándares Internacionales se redactan de acuerdo con las reglas dadas en las directivas ISO/IEC, Parte 2

La principal tarea del Comité Técnico Conjunto es preparar Estándares Internacionales. La redacción de Estándares Internacionales aprobados por el comité técnico conjunto pasa a los cuerpos nacionales para votación. La publicación como Estándar Internacional requiere la aprobación de al menos el 75% de los cuerpos nacionales mediante el voto.

## 3.2 Introducción a ISO/IEC SC37

Es un subcomité que forma parte del Comité Técnico Conjunto ISO/IEC JTC 1, tecnología de la Información, dedicado a biometría.

La implantación de los avances tecnológicos actuales no siempre resultan sencillos, a menudo existen problemas debidos a la falta de compatibilidad e interacción entre aplicaciones y dispositivos, que por otro lado es fundamental y necesaria para el desarrollo de estas nuevas tecnologías.

En un momento dado se pensó en la realización de diferentes estándares, procedimientos y formatos de datos, por otro lado necesarios, para que se trabaje en este campo a nivel mundial y todos bajo una misma línea y directrices. Fue a finales de los 90 cuando se comienzan a realizar los primeros estudios y trabajar en la realización de estos formatos de datos. Pero no es hasta el 2002, debido a diferentes incompatibilidades y acelerado por los hechos acontecidos el 11 de septiembre de 2001, cuando se crea un subcomité internacional dedicado a la Identificación biométrica ISO/IEC JTC1/SC37, dentro del Comité Conjunto ISO/IEC sobre tecnologías de la Información (JTC1). Desde su creación el número de países implicados va creciendo significativamente y se va persiguiendo cubrir el máximo de ámbitos posibles y el desarrollo de nuevas técnicas que permitan una mejora de lo actual

Desde su creación, el ISO/IEC JTC1/SC37 "Biometrics", determina que su campo de actividad sea [7]:

> *"Tecnologías biométricas genéricas, correspondientes a seres humanos, para aportar interoperabilidad e intercambio de datos entre aplicaciones y sistemas. Las normas sobre identificación biométrica humana genérica, incluyen:*
>
> • *Entornos de ficheros comunes; interfaces de programación de aplicaciones biométricas; formato de intercambio de datos biométricos; perfiles relacionados con la biometría; aplicación de criterios de evaluación para las tecnologías biométricas; metodologías para la verificación y emisión de informes sobre el rendimiento de los sistemas, y los aspectos sociales y jurisdiccionales*
>
> • *De todo este trabajo se excluye:*
>
> - *El trabajo realizado dentro del ISO/IEC JTC1/SC17, sobre la aplicación de las tecnologías biométricas a las tarjetas y la identificación personal*
> - *El trabajo realizado dentro del ISO/IEC JTC1/SC27, sobre las técnicas de protección de los datos biométricos, la verificación de la seguridad biométrica, sus evaluaciones y sus metodologías de evaluación."*

Para llevar a cabo todo este trabajo, se estableció una estructura inicial dividida en seis Grupos de Trabajo (*WG – Working Groups*), cada uno de ellos destinado a trabajar en un aspecto determinado del campo de la Identificación Biométrica:

- WG1 *Harmonised Biometric Vocabulary*: encargado de crear un catálogo de términos estandarizados que cubran todos aquellos conceptos relacionados con los sistemas biométricos.

- WG2 *Biometric Technical Interfaces*: cuyo ámbito de trabajo incluye los interfaces de comunicación entre aplicaciones y sistemas.

- WG3 *Biometric Data Interchange Formats*: destinado a definir los formatos de datos para cada una de las distintas modalidades biométricas.

- WG4 *Biometric Functional Architecture & Related Profiles*: cuya misión es la de definir las distintas arquitecturas biométricas y los distintos perfiles de aplicación.

- WG5 *Biometric Testing and Reporting*: encargado de definir los mecanismos de evaluación de los sistemas biométricos, desde el punto de vista de la funcionalidad, así como de indicar la forma en la que deben hacerse los informes de dichas evaluaciones.

- WG6 *Cross-jurisdictional and Societal Aspects*: que se encarga de estudiar los efectos jurídicos de la instalación de sistemas biométricos, así como la influencia social de su aplicación.

Cada grupo de trabajo tiene un coordinador responsable, y la presidencia y secretaría de todo el Subcomité la ostenta actualmente Estados Unidos, en las personas *Fernando Podio*, como Presidente, y *Lisa Rajchel*, como Secretaria.

A raíz de la creación del subcomité internacional, y teniendo en cuenta el gran interés de la Biometría en España (tanto en su aplicación, como en su investigación y desarrollo), se detectó la necesidad de que se creara un subcomité espejo del internacional. Este subcomité, integrado en AENOR, representaría los intereses españoles en los trabajos internacional, a la vez que coordinaría algún potencial trabajo en esta línea, en el ámbito puramente nacional.

El pasado 16 de junio de 2006, se procedió a la Constitución formal del Subcomité AEN/CTN71/SC37 con el título de *"Identificación Biométrica"*. Se decidió que el Campo de Actividad fuera el mismo que el del Subcomité internacional, y que en el futuro se determinase una estructura totalmente a imagen de la de ISO.

En la actualidad, el Subcomité está constituido por 11 empresas, 5 organismos e instituciones públicos, y 9 universidades, todos ellos funcionando como vocales, y coordinados por un Presidente (Ángel Luís Puebla) y un Secretario (Raúl Sánchez Reillo).

## *3.3 Introducción a ISO/IEC 19794 Project*

ISO/IEC 19794 define un formato de intercambio de datos biométricos bajo la denominación de tecnología de la Información – formatos de intercambio de datos biométricos. Con este proyecto se consigue la interoperabilidad de sistemas, de forma que si se modifica algún dispositivo, sensor, etc… se puede sustituir por otro que cumpla el formato.

Se compone de las siguientes partes:

- Parte 1: Marco de trabajo
- Parte 2: Minucias de huella dactilar
- Parte 3: Patrón espectral de huella dactilar
- Parte 4: Imagen de huella dactilar
- Parte 5: Imagen facial
- Parte 6: Imagen del iris
- Parte 7: Series de datos temporales para firmas
- Parte 8: Patrón esquemático de huella dactilar
- Parte 9: Imagen vascular
- Parte 10: Silueta geométrica de la mano
- Parte 11: Datos dinámicos de la firma
- Parte 13: Voz
- Parte 14: ADN

Dentro de este proyecto, como se ha visto anteriormente, la parte 7 y 11 definen los formatos de almacenamiento de datos de firmas manuscritas mediante la definición de las estructuras de datos correspondientes y los requisitos que deben cumplir cada uno de ellos, como sus rangos de valores o diferentes particularidades en función a las necesidades precisadas.

## *3.4 Firmas en ISO/IEC 19794 Project*

### 3.4.1 19794-7

#### 3.4.1.1 Introducción

Esta parte de ISO/IEC 19794 tiene como objetivo especificar los formatos de almacenamiento de datos para firmas en forma de series temporales utilizando dispositivos tales como tabletas digitalizadoras o sistemas de bolígrafo avanzados para conseguir interoperabilidad entre los diferentes sistemas biométricos y las aplicaciones. En ella se define la estructura de datos para almacenar las firmas y una descripción de su contenido.

Los datos capturados se almacenan por canales, que almacenan la información recogida, es decir, los diferentes tipos de información, tales como la posición del bolígrafo, velocidad de escritura, etc… La denominación de los canales es en función de la información que contienen y son los siguientes [8]:

- Canales de posición: a la hora de digitalizar los datos, se define un sistema de coordenadas tridimensional, tres canales graban la posición tridimensional del bolígrafo. La unidad de medida es el metro.
    - X: coordenada X o posición horizontal del bolígrafo: incrementa su valor de coordenada hacia la derecha
    - Y: coordenada Y o posición vertical del bolígrafo: el incremento de su valor es hacia arriba
    - Z: coordenada Z o altura del bolígrafo sobre el plano de escritura: movimiento del bolígrafo levantando o presionando la tableta, obtiene su valor mayor de coordenada conforme aumenta la presión.

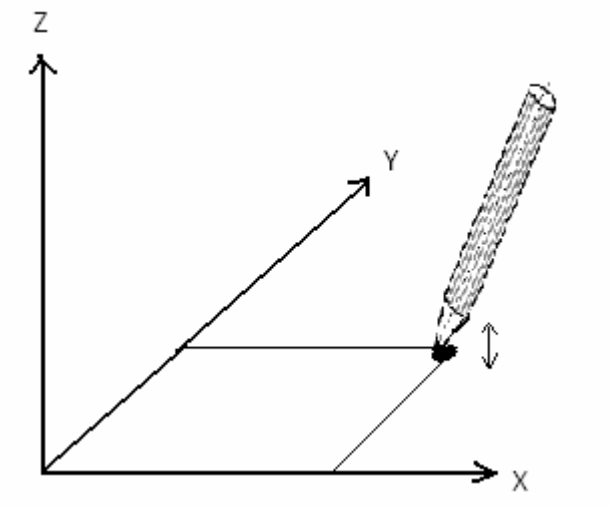



**Figura 5 - Orientación del bolígrafo**

- Canales de velocidad: indica la rapidez con la cual se escribe la firma. La unidad de medida es el m/s
    - VX: velocidad en el plano horizontal
    - VY: velocidad en el plano vertical
- Canales de aceleración: almacena la diferencia de velocidades, es decir, la variación de la velocidad de escritura a lo largo de toda la firma. La unidad de medida es $m/s^2$
    - AX: aceleración en el plano horizontal
    - AY: aceleración en el plano vertical
- Tiempo: tiempo transcurrido desde el comienzo de la toma de datos. La unidad de medida es el segundo
- Diferencia temporal: diferencia temporal de la muestra actual con respecto a la anterior. La unidad de medida es el segundo
- Presión: valor asociado a la presión con la que se actúa con el bolígrafo sobre la tableta. La unidad de medida es el Newton
- Estado : almacena “0” si el bolígrafo no está tocando la tableta y “1” si por el contrario hay contacto
- Orientación: mide la inclinación del bolígrafo con respecto a los ejes. La unidad de medida es el ángulo (º)
    - TX: inclinación del bolígrafo sobre el eje horizontal

- TY: inclinación del bolígrafo sobre el eje vertical

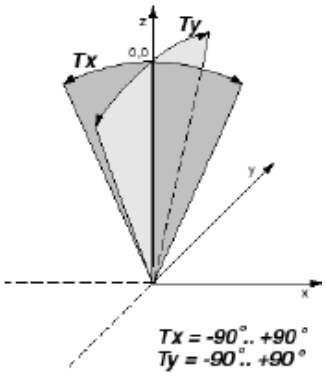



**Figura 6 - Ángulos de inclinación**

- Az: ángulo de azimut del bolígrafo. Puede almacenar valores entre 0 y 360, correspondientes al valor en grados del ángulo.

- El: ángulo de elevación del bolígrafo. Puede almacenar valores entre 0 y 90, correspondientes al valor en grados del ángulo.

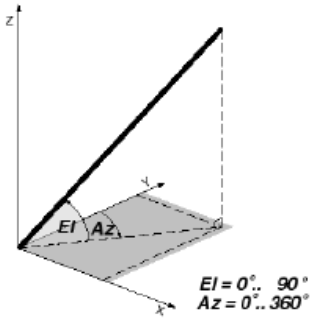



**Figura 7 - Ángulos de elevación y azimut**

- R: rotación del bolígrafo. Almacena valores comprendidos entre 0 y 360, que corresponde con el valor en grados del giro

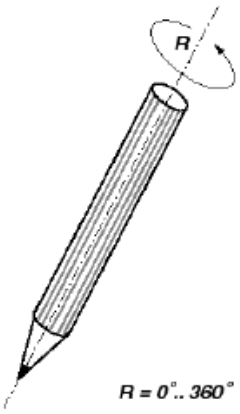



**Figura 8 - Rotación del bolígrafo**

En este estándar se definen dos versiones con diferentes formatos de almacenamiento de los datos. Por un lado el formato completo (full) y por otro el formato compacto (compact), que intenta reducir el tamaño de los datos almacenados mediante la supresión de la cabecera y el escalado de la información en un menor número de bytes.

#### 3.4.1.2 Formato completo

Contiene información más completa y específica, es el utilizado para uso general. Como desventaja con respecto al compacto cabe mencionar su mayor tamaño, que en bases de datos tan extensas como las tratadas puede suponer un problema. La estructura de almacenamiento de datos de este estándar se compone de dos bloques bien diferenciados: cabecera y cuerpo.

- **Cabecera**: contiene información sobre el bloque de datos biométricos, sobre los canales que se incluyen en la captura de datos y sus características. Se compone de los siguientes bloques:
    - *Identificador de formato* (Format identifier): se utiliza para identificar tipo de rasgo biométrico, en este caso firma manuscrita. Su valor son tres caracteres ASCII “SDI” seguidos por un cero como terminación.
    - *Número de versión* (Version number): se compone de cuatro bytes, los tres primeros con un valor ASCII “ 10” seguidos por cero como terminación, que corresponde a la versión uno del estándar.
    - *Descripciones de canal* (Channel descriptions): es un campo variable, que se compone de los siguientes subapartados:
        - Inclusión de canal (Channel inclussion): se compone de dos bytes, representando cada bit a uno de los canales. Si el bit tiene valor uno, indica que el canal ha sido incluido y se tiene en cuenta a la hora de almacenarse para la firma. Si por el contrario el bit tiene un valor cero significa que el canal no tiene influencia. Cabe mencionar que los canales X e Y tienen que incluirse obligatoriamente.

| Octet 1 | | | | | | | | Octet 2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

**Figura 9 - Ejemplo del campo inclusión de canal. En este caso los canales que se incluyen son X, Y, T, F, S, Az, El y R**

- Descripción de canal (Channel description): existe uno por cada canal y como su nombre indica contiene información descriptiva de los canales. Dicha información se divide en los siguientes puntos:
    - Preámbulo (Preamble): se compone de ocho bits con valor uno o cero en función de presencia o no del atributo del canal correspondiente al bit en cuestión
    - Valor de escalado (Scaling Values): se compone de dieciséis bits que almacenan el exponente (E) y la fracción (F). Su valor indica el número que permite realizar la conversión entre el formato de almacenamiento de datos del estándar y el formato real establecido.
        - El exponente se representa mediante los cinco bits más significativos y contiene un entero con signo que representa el exponente en base dos del valor de escalado dividido entre 16. Puede tomar valores en el rango -16 a 15 pero para codificarse debe añadirse la cantidad de 16 para obtener un entero sin signo
        - La fracción son los once bits restantes, que contiene el bit que encuentra a la derecha de la mantisa del valor de escalado. La mantisa se escala en el rango 1 ≤ mantisa < 2.

    El valor de escalado abarca un rango desde $2^{-16}$ a (1 + 2047/2048) * $2^{15}$, por tanto, se calcula como:

$$scaling\_value = mantissa \cdot 2^{\exp onente}$$

Siendo:

$$mantissa = 1 + \frac{F}{2^{11}}$$

$$\exp onente = E - 16$$

- Mínimo (Min): definición del valor mínimo que se puede almacenar en el canal. Se codifica mediante dos bytes como enteros sin signo. Puede tomar valores entre 0-65535 para los canales Z, T, DT, F, Az, El, R, mientras que para el resto de canales (X, Y, VX, VY, AX, AY, TX, TY) toma valores comprendidos entre -32768 y 32767. Para codificar éstos últimos se añadirá la cantidad de 32768. El canal S toma como valor 0 ó 1 y se codifica con un byte como entero sin signo.

- Máximo (Max): definición del valor máximo que se puede almacenar en el canal. Su codificación es igual que para el mínimo.

- Media (Mean): media aritmética de todos los valores del canal, se calcula como:

$$media = \frac{1}{N}\sum_{i=1}^{N} valor\_canal\_i$$

  Su codificación es igual que para mínimo

- Desviación estándar (std): desviación estándar de todos los valores del canal, se calcula como:

$$std = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(valor\_canal\_i - media)^2}$$

  Su codificación es igual que para mínimo

- *Reservado*: octeto reservado para revisiones futuras de esta especificación

A continuación se muestra una imagen que resume la estructura de la cabecera:

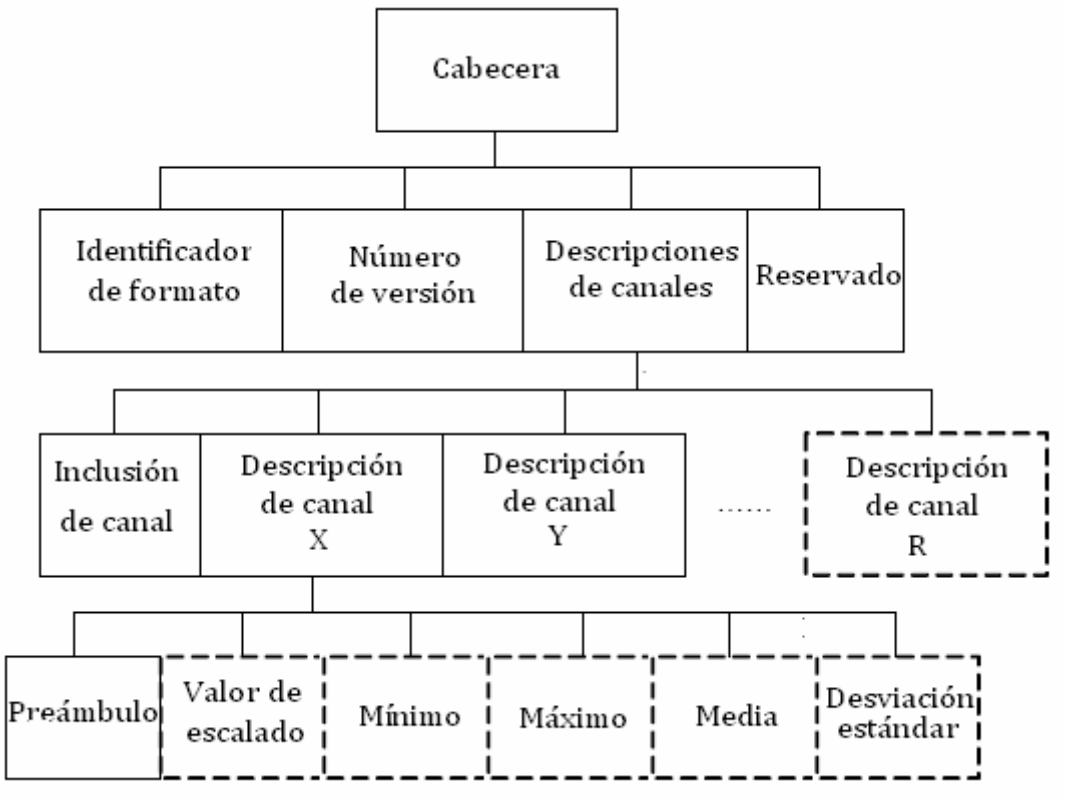



**Figura 10 - Cabecera del bloque de datos de series temporales para el formato completo**

Los bloques con líneas discontinuas son opcionales mientras que el resto son de obligada inclusión.

- **Cuerpo**: almacena las muestras capturadas de la firma, agrupándolas según los diferentes canales y el tipo de información. Se divide en:

  - *Preámbulo* (Preamble): indica la presencia o ausencia de los datos extendidos opcionales mediante la colocación de uno o cero en el bit más significativo del octeto.

  - *Número de muestras* (Number of sample points): tres bytes que indican el número de muestras totales capturadas mediante la tableta digitalizadora.

  - *Secuencia de muestras* (Sequence of sample points): se compone de todas las muestras capturadas, agrupando la información para cada una de ellas por canales. Cada uno de los canales se codifica mediante dos bytes como enteros sin signo. Puede tomar valores entre 0-65535 para los canales Z, T, DT, F, Az, El, R, mientras que para el resto de canales (X, Y, VX, VY, AX, AY, TX, TY) toma valores comprendidos entre -32768 y

32767. Para codificar éstos últimos se añadirá la cantidad de 32768. El canal S toma como valor 0 ó 1 y se codifica con un byte como entero sin signo.

- *Longitud de los datos extendidos* (Extended data length): dos bytes que guardan el número de bytes que se almacenan como datos extendidos opcionales
- *Datos extendidos* (Extended data): en este campo se almacenarán todos los datos opcionales, su longitud es variable.

A continuación se muestra una imagen que resume la estructura del cuerpo:

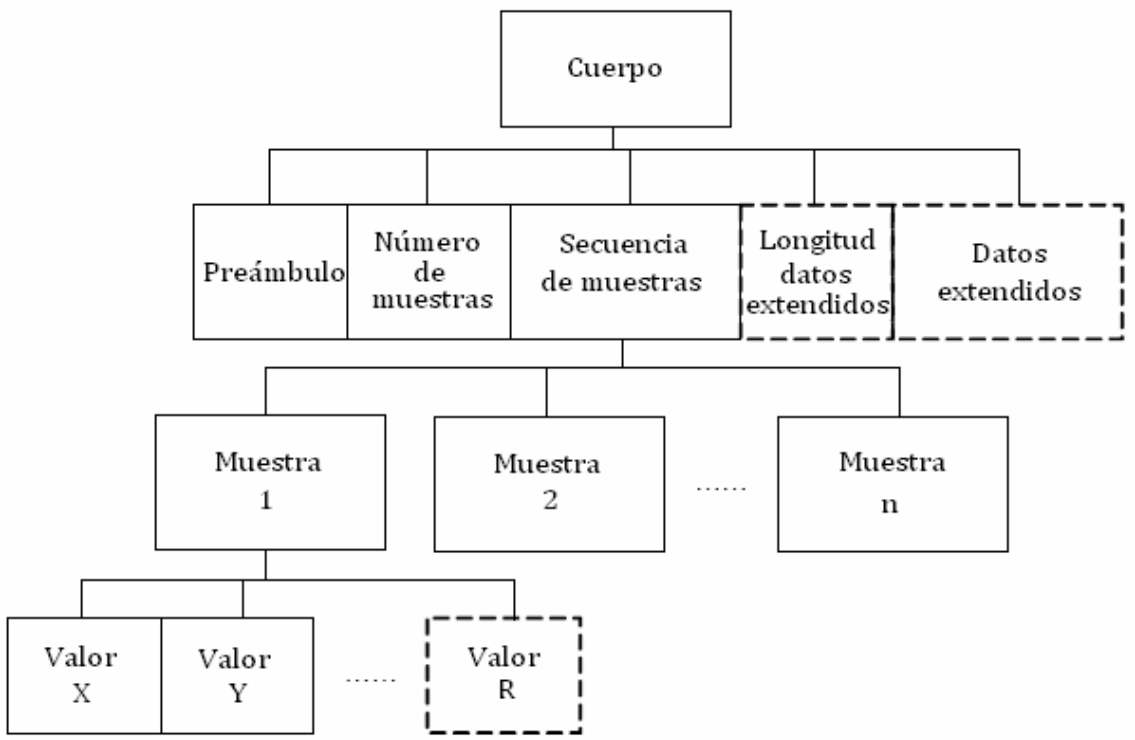



**Figura 11 - Cuerpo del bloque de datos de series temporales para el formato completo**

### 3.4.1.3 Formato compacto

En este caso se pretende reducir considerablemente el número de bytes almacenados con respecto al formato completo. Para ello, se excluye la cabecera y todos los campos del cuerpo a excepción de las muestras, que además representan los datos de forma mucho más reducida, almacenando la información con un número menor de bytes.

Cada uno de los canales se codifica mediante un byte como enteros sin signo. Puede tomar valores entre 0-255 para los canales Z, T, DT, F, Az, El, R, mientras que para el resto de canales (X, Y, VX, VY, AX, AY, TX, TY) toma valores comprendidos entre -128 y 127. Para codificar éstos últimos se añadirá la cantidad de 128. El canal S toma como valor 0 ó 1 y se codifica con un byte como entero sin signo.

La elevada importancia del tamaño de los datos en algunas aplicaciones lleva a la utilización de este formato reducido compensa el hecho de la pérdida de información.

La estructura de bloques del formato compacto sigue el modelo que se presenta a continuación:

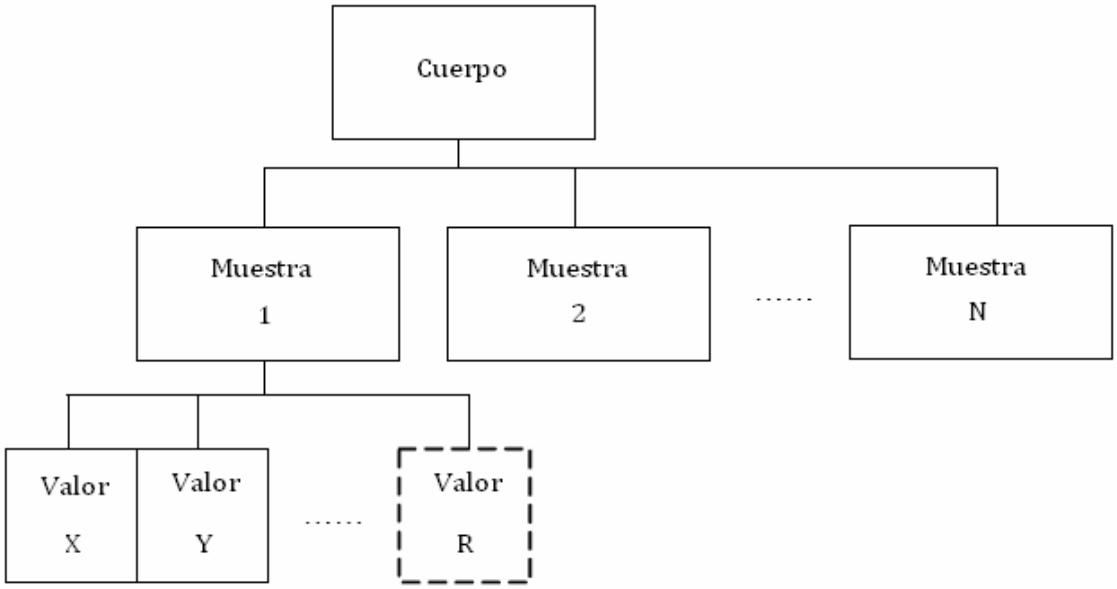



**Figura 12 - Cuerpo 19794-7 compacto**

### 3.4.2 19794-11

#### 3.4.2.1 Introducción

Es un nuevo formato creado como una compresión inteligente del formato 19794-7. Los datos almacenados son una secuencia de segmentos de firmas, precedidas por una cabecera que es común a todas ellas. Cada firma se representa como una secuencia de "pen strokes" y "pressure strokes", que es una forma de segmentar la firma en función de puntos singulares calculados, y se guardan datos como las posiciones de inicio y fin de los segmentos, presión, longitud del segmento, velocidades, aceleraciones y dirección.

El proceso seguido es la búsqueda de puntos singulares en la firma. Se define un punto singular cuando se produce un cambio de dirección en la coordenada "x", "y" o en la presión, o un evento de pen-down o pen-up.. Para los dos primeros casos (cambios de dirección en la coordenada X o Y) se tiene un "pen stroke" y para el último (cambio de dirección en la presión) un "pressure stroke". Para cada punto singular se almacena un resumen de los datos de las muestras contenidas en este segmento, de forma que no es necesario guardar todas las muestras, solamente un resumen de un grupo de muestras acotadas por puntos singulares. Se definen los siguientes puntos singulares:

- Pen-up: se produce cuando se levanta el bolígrafo del dispositivo de captura de datos
- Pen-down: tiene lugar cuando el bolígrafo toca el dispositivo de captura de datos, estando previamente levantado.
- Turning point: ocurre cuando se produce un cambio de dirección en las coordenadas X o Y o para la presión, esto es, cuando el movimiento del bolígrafo sobre el dispositivo cambia la dirección de escritura en alguno de los canales citados. Para presión el cambio de dirección quiere decir que se pase de aumentar presión al escribir a reducir presión o viceversa mientras que en los otros dos canales, implica un cambio de dirección izquierda-derecha o viceversa, o bien arriba-abajo o viceversa, respectivamente.

A diferencia del formato anterior, en este caso no es necesaria la creación de una versión compacta, debido a que el formato se define a si mismo como suficientemente compacto, debido a la compresión generada por la segmentación utilizada, manteniendo una buena calidad de los datos de la firma.

El esquema de bloques que se sigue para la generación de este formato es el mostrado a continuación. Como se mencionaba, a partir de 19794-7 se realiza un procesado

intermedio en el que se comprime la información mediante el cálculo de puntos singulares de la firma y posteriormente se almacenan los datos

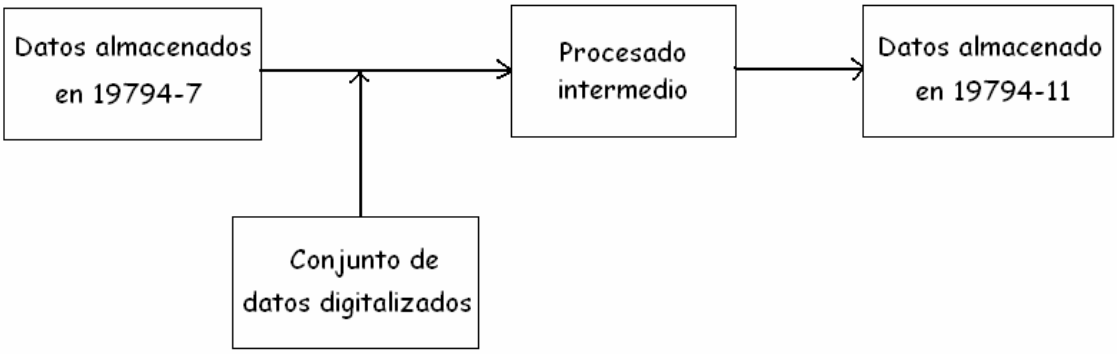



**Figura 13 - Proceso de creación de 19794-11**

### 3.4.2.2 Organización

La estructura de 19794-11 se puede esquematizar de la siguiente forma [9]:

- **Cabecera:** contiene información sobre todos los datos biométricos grabados. Se compone de los siguientes bloques:
    - *Identificador de formato* (Format identifier): se utiliza para identificar el bloque de datos. Su valor son tres caracteres ASCII “SPD” seguidos por un cero como terminación
    - *Número de versión* (Version number): se compone de cuatro bytes, los tres primeros con un valor ASCII “010” seguidos por cero como terminación.
    - *Longitud de grabado (Length of record):* número total de bytes del bloque de datos, incluyendo cabecera y cuerpo.
    - *Número de representaciones (Number of representations):* número de firmas almacenadas en la estructura
    - *Identificador del dispositivo de captura (Capture device vendor ID):* indica la identificación del dispositivo de captura de datos del distribuidor
    - *Identificador del tipo de dispositivo de captura (Capture device type ID):* tipo de producto que realiza la captura de datos.

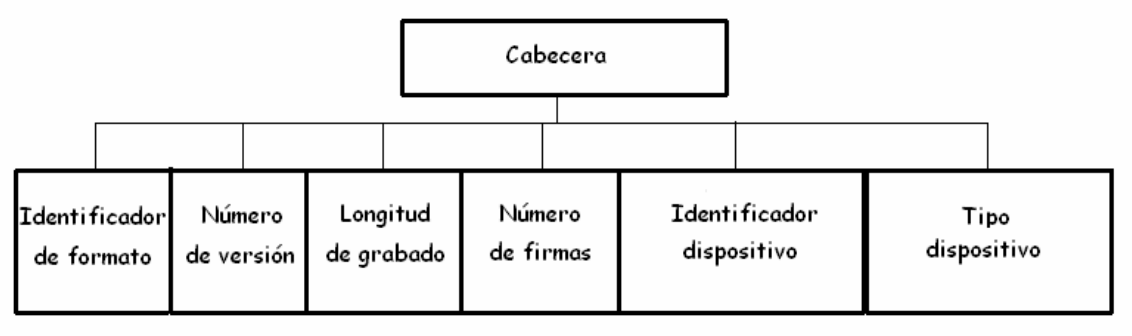



**Figura 14 - Cabecera 19794-11**

- **Cuerpo:** almacena las muestras capturadas de la firmas, agrupándolas por firmas y en función de los puntos singulares obtenidos, que dan lugar a los diferentes "pen strokes" y "pressure strokes" que contienen los resúmenes de las muestras del segmento correspondiente. Se divide en:

  - *Preámbulo (Preamble):* indica la presencia o ausencia de los datos extendidos opcionales mediante la colocación de uno o cero en el bit más significativo.

  - *Valores de escalado (Scaling values):* se compone de exponente y fracción. Su valor indica el número que permite realizar la conversión entre el formato de almacenamiento de datos del estándar y el formato real establecido

  - *Resolución (Sampling Resolution):* es la frecuencia de muestreo del dispositivo digitalizador medida en hercios (Hz)

  - *Secuencia de firmas (Sequence of signature representations):*

    - Longitud strokes (Pen stroke length): número de segmentos pen strokes y número de pressure strokes

      Longitud = num_pen_strokes + num_pressure_strokes

    - Pen strokes: se define como el movimiento del bolígrafo entre dos puntos singulares. Son segmentos compuestos de muestras de la firma cuyo inicio y final está marcado por un punto singular. Cabe destacar que la búsqueda de los puntos singulares se basarán en los cambios de dirección de las coordenadas X e Y.

      - Tipo stroke (PST value): identifica al tipo de stroke que se produce y pueden tomar los siguientes valores

1. Comienza con el punto singular pen-down y termina con el punto singular turning point
2. Comienza con turning point y termina con turning point
3. Turning point a pen-up
4. Pen-down a pen-up

- Valores XY (XY values): valores de las coordenadas indicadas para el punto singular inicial y final
- Valores temporales (Time values): valores del tiempo transcurrido para el punto singular y final
- Valores de presión (P values): valores máximo, mínimo y medio del segmento de muestras entre puntos singulares
- Longitud (L value): longitud del intervalo de muestras, corresponde a la siguiente expresión:

$$L = \sqrt{(X_{MAX} - X_{MIN})^2 + (Y_{MAX} - Y_{MIN})^2}$$

- Velocidades (V values): valores de velocidad máximo, mínimo y medio del intervalo entre puntos singulares para los canales X e Y. Se calculan mediante la expresión

$$V_{Xi} = \frac{X_{i+1} - X_i}{t_{i+1} - t_i} \qquad V_{Yi} = \frac{Y_{i+1} - Y_i}{t_{i+1} - t_i}$$

- Aceleraciones (A values): valores de aceleración máximo, mínimo y medio del intervalo entre puntos singulares para los canales X e Y. Se calculan mediante la expresión:

$$A_i = \frac{V_{i+1} - V_i}{t_{i+1} - t_i}$$

- Dirección (D value): el vector dirección se calcula en base a:

$$D = arctg\left(\frac{Y_1 - Y_0}{X_1 - X_0}\right)$$

- Pressure strokes: su significado es el mismo que pen stroke, con la particularidad de que en lugar de tener en cuenta las coordenadas X e Y, en este caso se basa en valores de presión.
    - Tipo stroke (PST value): identifica al tipo de stroke que se produce y pueden tomar los siguientes valores
        5. Pen-down a turning point
        6. Turning point a turning point
        7. Turning point a pen-up
        8. Pen-down a pen-up
    - Valores inicial y final de presión (PSF Values): valores de presión de la primera y última muestra del pressure stroke.
    - Valores de presión (P values): valores máximo, mínimo y medio del segmento de muestras entre puntos singulares
    - Valores XY (XY values): valores de las coordenadas indicadas para el punto singular inicial y final
    - Valores temporales (Time values): valores del tiempo transcurrido para el punto singular inicial y final
- Características globales de la firma (Overall features data):
    - Tiempo total (Total time): tiempo transcurrido desde el comienzo de la captura de datos hasta la última muestra tomada
    - Total puntos (Total points): número total de puntos capturados o número de bytes totales guardados
    - Valores medios (Mean values): media aritmética de las coordenadas X e Y de todas las muestras tomadas de la firma.
    - Valores desviación estándar (std values): media aritmética de las coordenadas X e Y de todas las muestras tomadas de la firma. Se calcula mediante la fórmula:

$$S = \sqrt{\frac{\sum (v-m)^2}{n-1}}$$

Siendo v = valores de X o Y, m= media aritmética de X o Y, n = número total de puntos

- Valor correlación (CC value): coeficiente de correlación calculado como:

$$R = 1000 \cdot \left\{ 1 + \frac{n\sum X \cdot Y - \sum X \sum Y}{\sqrt{\left[n\sum X^2 - \left(\sum X\right)^2\right] \cdot \left[n\sum Y^2 - \left(\sum Y\right)^2\right]}} \right\}$$

- *Longitud de los datos extendidos (Extended data length):* número de bytes que se almacenan como datos extendidos opcionales
- *Datos extendidos (Extended data):* en este campo se almacenarán todos los datos opcionales.

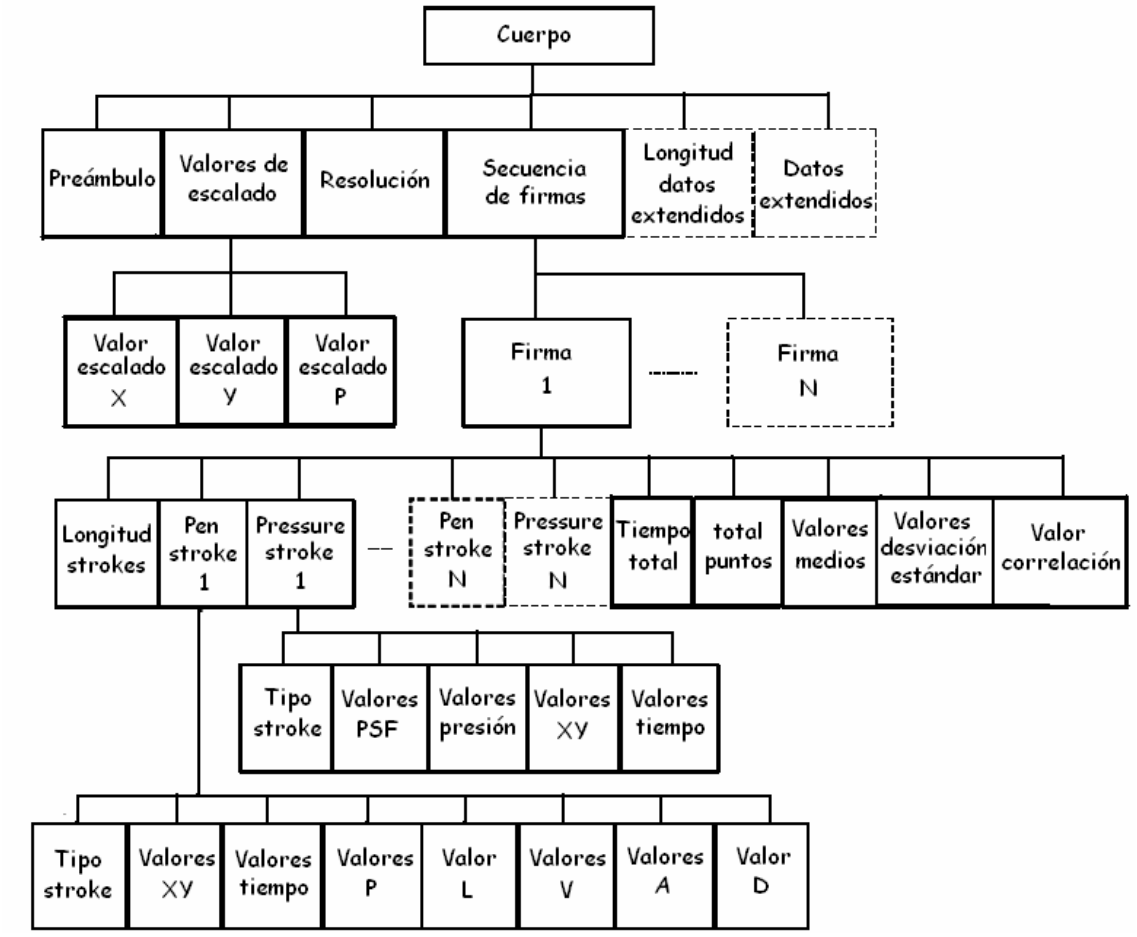



**Figura 15 - Cuerpo 19794-11**

# 4 Implementación de Normas de Firma

Para el desarrollo de las implementaciones de las normas de firma se ha utilizado el lenguaje de programación C, mediante la plataforma Visual Studio 2005. Las implementaciones realizadas han sido:

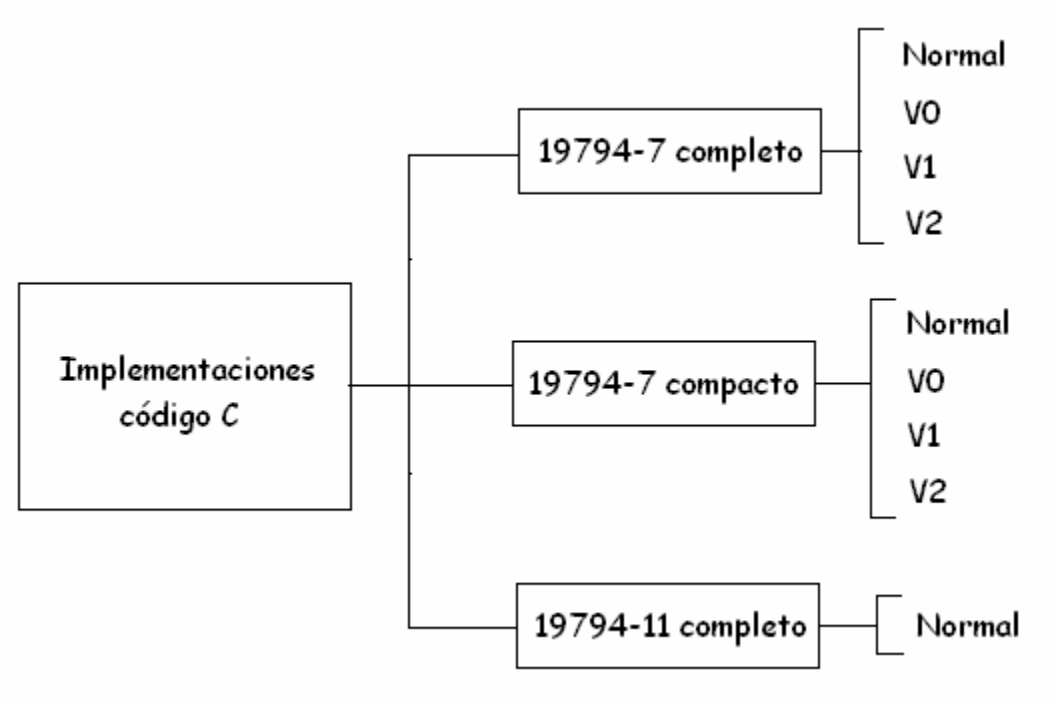



**Figura 16 - Implementaciones de normas de firmas**

Para la norma 19794-7 se definen dos versiones en el estándar, completa y compacta. La primera se compone de un bloque cabecera y otro para datos, mientras que el segundo no incluye cabecera y solamente contiene los datos correspondientes a la secuencia de muestras de los canales. Éste es una versión más compacta del formato completo, destinado a reducir la información que se desea guardar, almacenando con un menor tamaño de bytes las muestras a procesar por los dispositivos. Para el caso de 19794-11 solamente contiene un formato de datos.

## *4.1 19794-7 FULL*

### 4.1.1 Estructura de ficheros

La estructura de ficheros que se sigue en el proyecto es la siguiente:

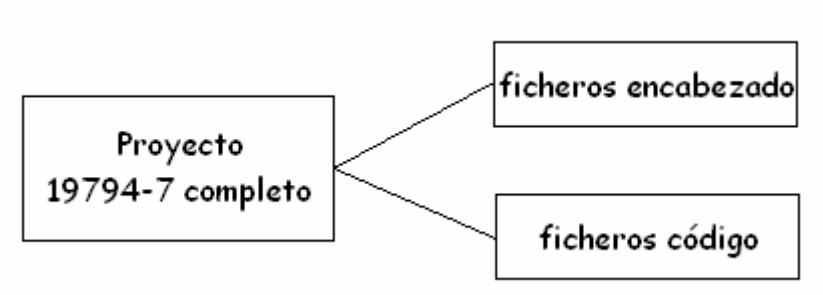



**Figura 17 - Estructura de ficheros de 19794-7 completo**

- Archivos de encabezado
  - Body.h: definición de la estructura de datos del cuerpo de la norma.
  - Channel.h: definiciones de los parámetros de los distintos canales, tales como valores máximos, mínimos, su inclusión o no en la lectura de datos…
  - Defines.h: constantes utilizadas en el desarrollo del programa, algunas de las más importantes son la que indica si el computador es big endian o little endian y las definiciones de los tipos de datos.
  - Fichero_iso.h: contiene la definición de las funciones de fichero_iso.c
  - Ficheros.h: definición de las funciones de ficheros.c
  - Firmlib.h: contiene las librerías utilizadas en el programa y los ficheros de definiciones
  - Fullformat.h: definición de la estructura global de la norma, contiene las llamadas a los ficheros de definiciones de estructuras de datos de cabecera y el cuerpo
  - Header.h: definición de la estructura de datos de la cabecera de la norma.
  - Iso_19794.h: definición de las funciones de iso_19794.c

- Mate.h: definición de las funciones mate.c

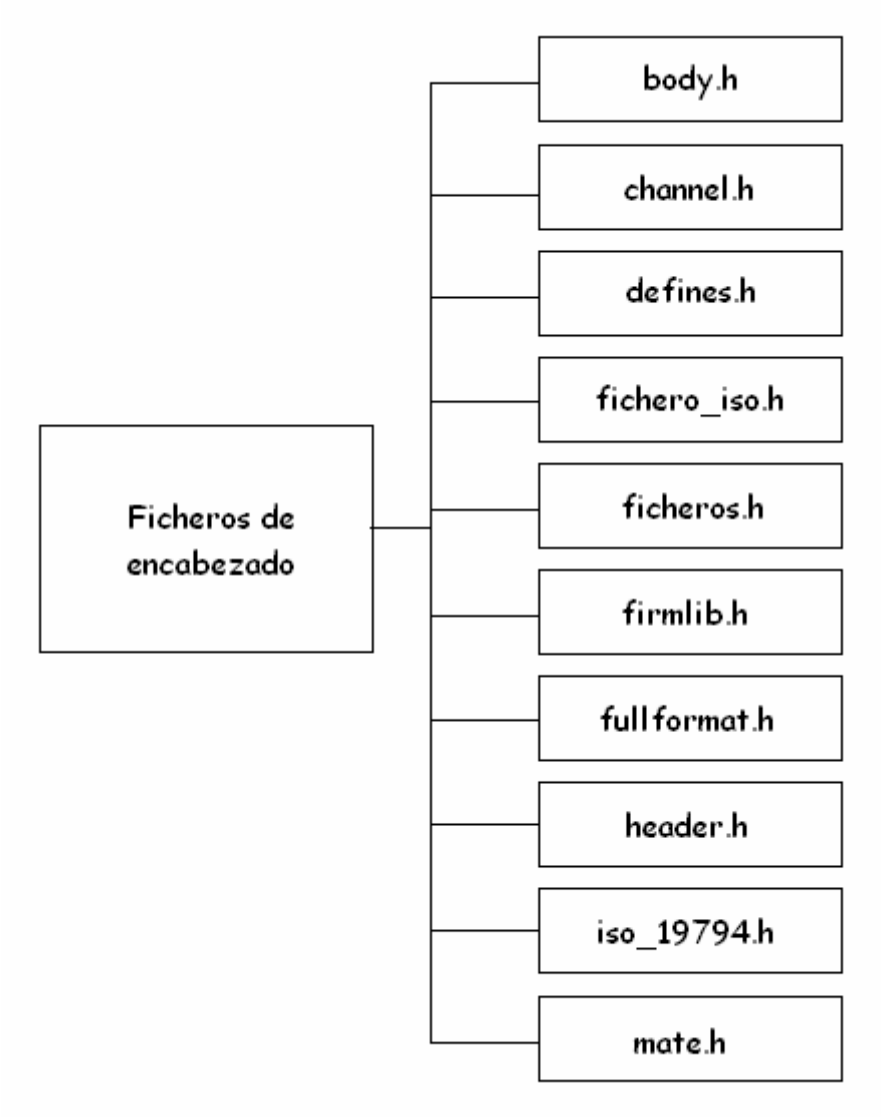



**Figura 18 - Ficheros de encabezado de 19794-7 completo**

- Archivos de código fuente

  - Entrega4.c: contiene el programa principal de la implementación, "main", desde el que se realiza las llamadas a las diferentes funciones que desarrollan la generación de las firmas

  - Fichero_iso.c: contiene las funciones:

    - MCyT2ISO19797_1_Normal: función encargada de la generación de los ficheros de firmas, desde él se llaman a las diferentes funciones que realizan cada uno de los puntos necesarios para la creación de las firmas

- firm2bin: escritura en ficheros de los datos recogidos de la base de datos MCyT
- writeHeader: escritura en fichero de la cabecera de la firma
- writeBody: escritura en fichero del cuerpo de la firma
- escribe: comprueba si el ordenador tiene arquitectura little o big endian, que es la forma en que se almacenan los bytes en el ordenador, y en función de ello, mediante una llamada a la función "fwrite" escribe los datos que se le indican mediante parámetro en el fichero correspondiente. La arquitectura big endian indica que se almacena en primer lugar los valores mayores o bytes más significativos, mientras que little endian almacena primero el dato menor o bytes menos significativos.

- Ficheros.c: las funciones que en él residen son:
  - String_txt: confecciona el path en el que se encuentra el fichero de texto de la base de datos MCyT del que se van a extraer los datos para crear la firma
  - String_iso: crea el path del fichero donde se guardará la firma
  - Lee_datos: lectura de datos del fichero correspondiente de la base de datos MCyT (mediante la función "fscanf"), comprobación de que cumplen con los requisitos, es decir, son correctos, y almacenamiento en una estructura para ser tratados y ordenados posteriormente
- Iso_19794.c: en él se implementan las funciones:
  - initHeader: inicialización de la cabecera de la norma de acuerdo a las constantes definidas.
  - getPreambleChannel: inicialización del preámbulo de los canales de acuerdo a las constantes indicadas
  - getChannelValue: inicialización de los valores de los canales de información
  - initBody: inicialización del cuerpo de la norma de acuerdo a las constantes definidas
  - createBdbHeader: creación de la cabecera de la norma a partir de los datos recogidos de los ficheros de la base de datos MCyT

- createBdbBody: creación del cuerpo de la norma a partir de los datos recogidos de los ficheros de la base de datos MCyT

- Mate.c: las funciones que contiene realizan cálculos matemáticos sobre los datos que se indican en los parámetros:
  - getE_F: obtención de los valores exponente y fracción de los campos de escalado de valores (scaling values).
  - meanFunc: cálculo del valor medio de los datos que se pasan como parámetros
  - stdFunc: cálculo de la desviación estándar de los datos pasados como parámetros.

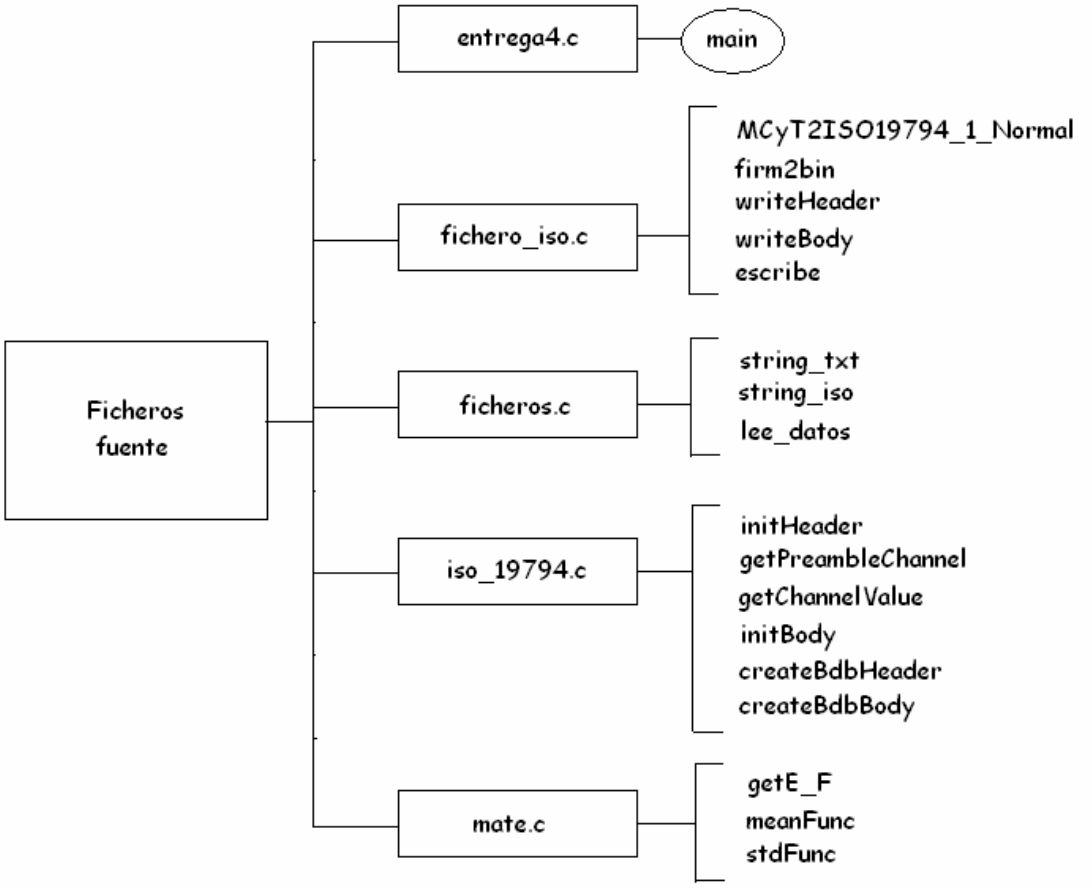



**Figura 19 - Ficheros fuente de 19794-7 completo**

### 4.1.2 Desarrollo del código

La parte principal del código se desarrolla en el fichero “entrega4.c”, que contiene la función “main”, dividida principalmente en dos bloques, uno de ellos destinado a la creación de los ficheros que contienen las firmas genuinas y el otro bloque para la creación de las falsificaciones.

Con el fin de convertir cada uno de los ficheros de la base de datos MCyT al formato 19794-7 Full, se sitúan ambos bloques dentro de dos bucles, uno destinado a recorrer cada uno de los usuarios de la base de datos (100 usuarios) y el otro bucle se encargará de cada una de las firmas realizadas por un usuario (25 firmas genuinas y 25 falsificaciones).

Cada uno de los bloques mencionados se divide a su vez en tres apartados, cada uno de ellos identificado con una función:

String_txt

Esta función es la encargada de crear la ruta del archivo de la base de datos MCyT que se convertirá al formato 19794-7 Full. Para ello, previamente se ha debido colocar dicha base de datos en un directorio del ordenador en el cual se va a ejecutar el programa e indicarle éste al programa introduciéndolo en una variable específica, después de lo cual, al ejecutar el proyecto, ésta función se encargará de almacenar en una variable el directorio en el que se almacena el fichero que se está tratando en cada caso, así como el fichero particular también, para después, poder abrirlo y actuar sobre él sacando la información para guardarla en el formato mencionado.

String_iso

Una vez se convierta el archivo de la base de datos al formato indicado, debe guardarse en un fichero en una ubicación específica. Para ello esta función almacena en una variable el directorio donde se almacenarán los ficheros convertidos con su nuevo nombre, para posteriormente poder guardar los archivos en el nuevo formato.

MCyT2ISO19797_11_Normal

En este apartado se lleva a cabo la conversión de los ficheros de la base de datos al formato que nos ocupa. Para ello se siguen los siguientes pasos:

1. Apertura como lectura del fichero de la base de datos MCyT que se desea convertir para poder acceder a los datos de la firma

2. Lectura de los datos de dicho fichero y almacenado en una variable para poder operar con ellos.
3. Inicialización de la estructura de datos que se va a crear
    - Inicialización de la cabecera
    - Inicialización del cuerpo
4. Creación de la estructura de datos que almacenará la firma en formato 19794-7
    - Creación de la cabecera
    - Creación del cuerpo
5. Creación del fichero y almacenamiento de la estructura en el nuevo formato.

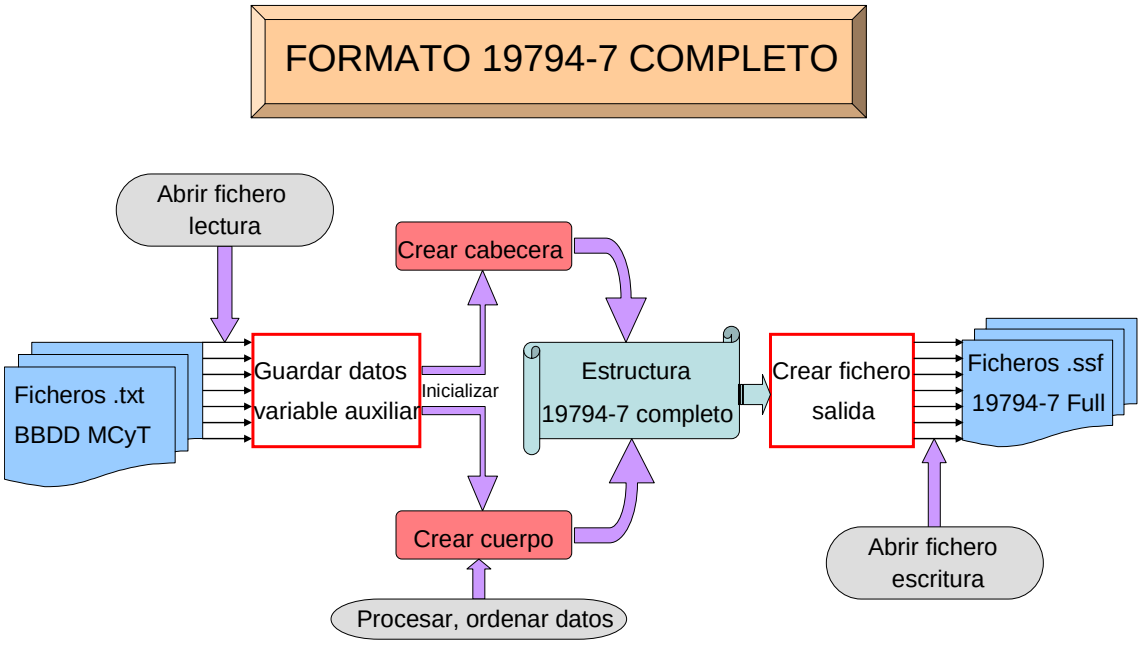



**Figura 20 - Diagrama de bloques de 19794-7 completo**

Cabe destacar que el programa lleva asociado un control de errores para cada una de las acciones que se llevan a cabo en él

### 4.1.3 Problemas encontrados

1. En el cálculo de los scaling values se tienen que almacenar en dos bytes el exponente y la fracción, el primero lo componen 5 bits y el segundo 11 bits. El problema reside en que había que hacer desplazamientos sobre una variable de 16 bits para colocar ambas.
2. A la hora de leer del fichero, se obtenían errores, bytes repetidos, valores mal introducidos…

### 4.1.4 Soluciones adoptadas

1. Se crean dos variables independientes para calcular cada exponente y fracción y, una vez calculados se almacenan en la variable de dos bytes mediante una máscara o desplazamiento para que quede cada campo en su lugar.
2. La apertura del fichero como lectura se realizaba como “rb” que no resultaba satisfactoria para el almacenamiento que se pretendía. El problema se arregló abriendo los ficheros como “r”

## *4.2 19794-7 COMPACTO*

### 4.2.1 Estructura de ficheros

La estructura de ficheros que se sigue en el proyecto es la siguiente:

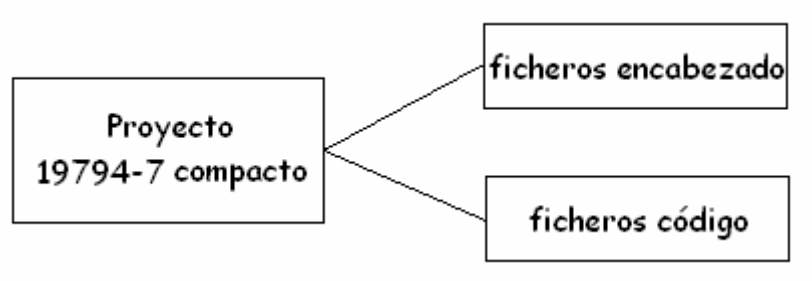



**Figura 21 - Estructura de ficheros de 19794-7 compacto**

- Archivos de encabezado
  - def_compacto.h: definición de las constantes y estructura de datos de la norma
  - funciones.h: definición de las funciones contenidas en el fichero “funciones.c”
  - mate.h: definición de las funciones matemáticas, implementadas en “mate.c”

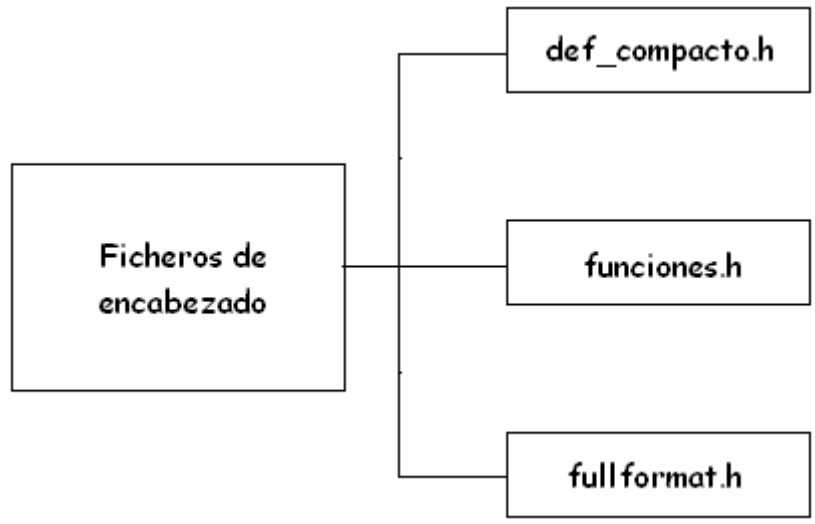



**Figura 22 - Ficheros de encabezado de 19794-7 compacto**

- Archivos de código fuente

  - compacto.c: contiene la función principal del programa, “main.c”, en la que se realizan las diferentes llamadas a las funciones para la generación de las firmas y su almacenamiento en ficheros.

  - funciones.c: contiene la implementación de las siguientes funciones:

    - LeeAlmacena_firma: lectura de datos de los ficheros de texto de la base de datos MCyT (mediante la función “fscanf”) y almacenamiento de los mismos en una estructura adecuada definida en los ficheros de definiciones.

    - GuardaFichero_7compacto: apertura del fichero donde se almacenará la firma (mediante la función “fopen”) y escritura de los datos en éste (con la función “fwrite”)

    - Convertir_muestras: mediante la llamada a la función “ConvierteCanal_2byte1” convierte los datos de los canales de dos bytes a un byte.

    - MCyT_2_19794_11: función general desde la que se lleva a cabo la llamada al resto de funciones auxiliares para la creación de las firmas

    - String_txt_MCyT: crea la ruta donde se almacenan los ficheros de la base de datos MCyT desde los que se obtendrán los datos para confeccionar las firmas

    - String_iso: crea la ruta de los ficheros donde se almacenan las firmas

    - escribe: función que se encarga de escribir en el fichero los datos que se le dan por parámetro mediante la función “fwrite” y en función de si el ordenador tiene una configuración big o little endian

  - mate.c:

    - max_vector: calcula el valor máximo de un vector dado por parámetro.

    - min_vector: calcula el valor mínimo de un vector dado por parámetro.

- ConvierteCanal_2byte1: realiza la conversión de una variable de dos bytes a un byte. Este proceso es necesario ya que la compresión de los canales de la versión completa a la compacta implica la reducción del número de bytes en que se almacenan sus valores de dos a uno, mediante la ponderación del valor a un rango de 256 ($2^8$).

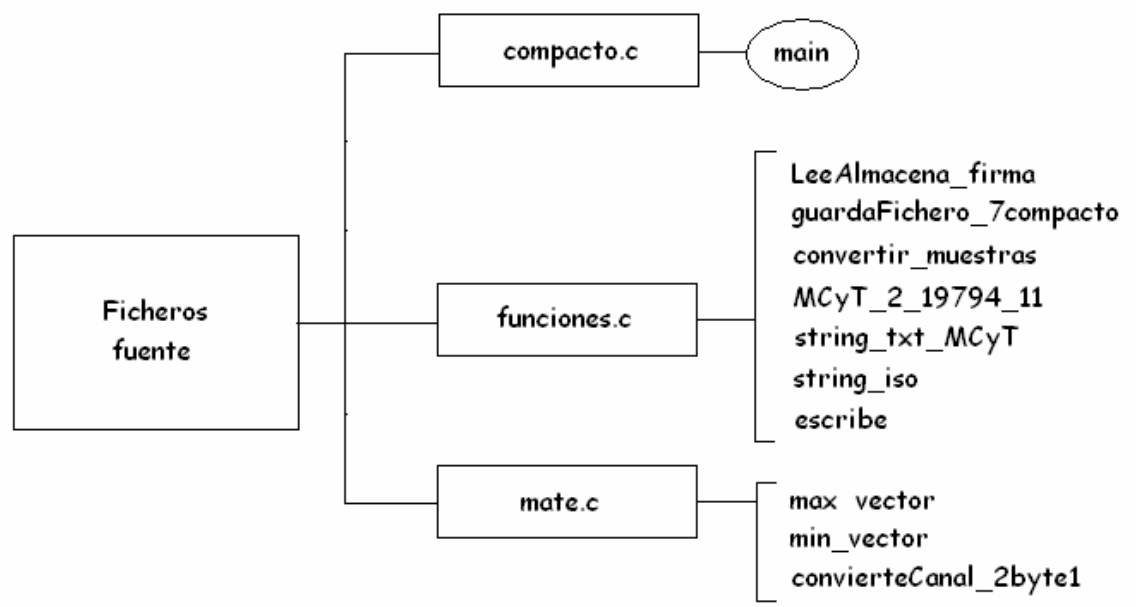



**Figura 23 - Ficheros fuente de 19794-7 compacto**

### 4.2.2 Desarrollo del código

La parte principal del código se desarrolla en el fichero "compacto.c", que contiene la función "main", dividida principalmente en dos bloques, uno de ellos destinado a la creación de los ficheros que contienen las firmas genuinas y el otro bloque para la creación de las falsificaciones.

Con el fin de convertir cada uno de los ficheros de la base de datos MCyT al formato 19794-7 Full, se sitúan ambos bloques dentro de un bucle destinado a recorrer cada uno de los usuarios de la base de datos (100 usuarios)

Cada uno de los bloques mencionados se divide a su vez en tres apartados, cada uno de ellos identificado con una función:

String_txt_MCyT

Esta función es la encargada de crear la ruta del archivo de la base de datos MCyT que se convertirá al formato 19794-7 Compacto. Para ello, previamente

se ha debido colocar dicha base de datos en un directorio del ordenador en el cual se va a ejecutar el programa e indicarle éste al programa introduciéndolo en una variable específica, después de lo cual, al ejecutar el proyecto, ésta función se encargará de almacenar en una variable el directorio en el que se almacena el fichero que se está tratando en cada caso, así como el fichero particular también, para después, poder abrirlo y actuar sobre él sacando la información para guardarla en el formato mencionado.

String_iso

Una vez se convierta el archivo de la base de datos al formato indicado, debe guardarse en un fichero en una ubicación específica. Para ello esta función almacena en una variable el directorio donde se almacenarán los ficheros convertidos con su nuevo nombre, para posteriormente poder guardar los archivos en el nuevo formato.

MCyT_2_19794_7Compacto

En este apartado se lleva a cabo la conversión de los ficheros de la base de datos al formato que nos ocupa. Para ello se siguen los siguientes pasos:

1. Lectura de fichero MCyT y almacenamiento de las muestras de la firma en una estructura con formato 19794-7 Compacto
2. Conversión de los valores obtenidos a la escala de un byte definida por la norma
3. Creación del fichero y almacenamiento de la estructura en el nuevo formato.

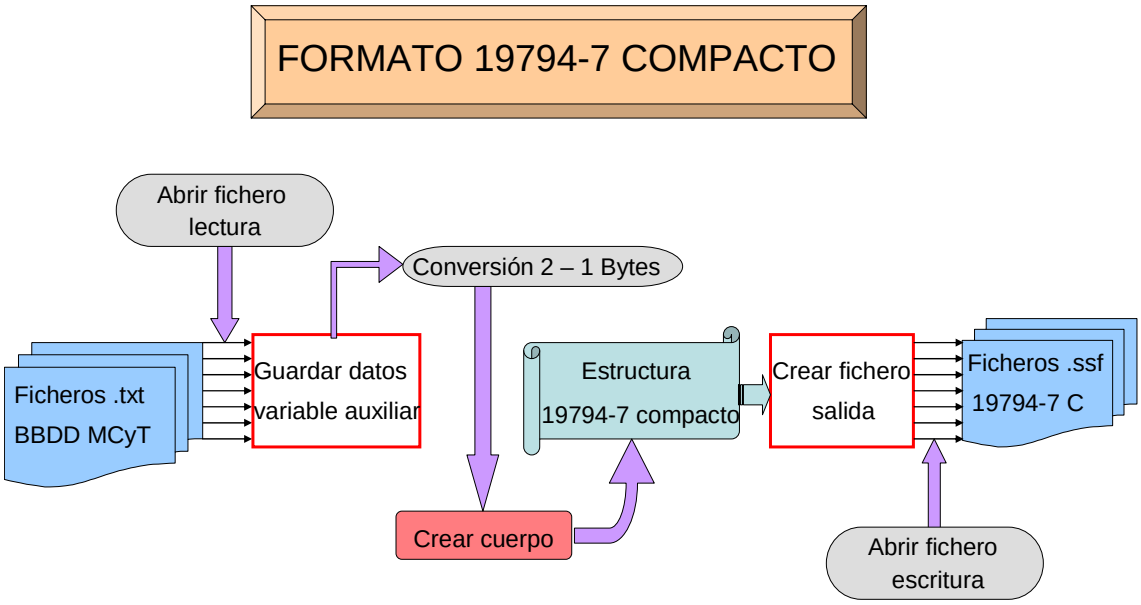



**Figura 24 - Diagrama de bloques de 19794-7 compacto**

Cabe destacar que el programa lleva asociado un control de errores para cada una de las acciones que se llevan a cabo en él

### 4.2.3 Problemas encontrados

1. En la conversión de dos bytes a un byte se debía buscar una función para calcular tanto los números con signo como los sin signo
2. El control de los decimales para aproximarlos a los enteros correctos.

### 4.2.4 Soluciones adoptadas

1. Se ponderan los valores para estar en el rango 0-255 y a los enteros con signo se le suma 128.
2. Para solucionar la aproximación de los decimales, en el cálculo se utilizan variables “float” que almacenan números decimales, y después de los cálculos se convierten a enteros.

## *4.3 19794-11*

### 4.3.1 Estructura de ficheros

La estructura de ficheros que se sigue en el proyecto es la siguiente:

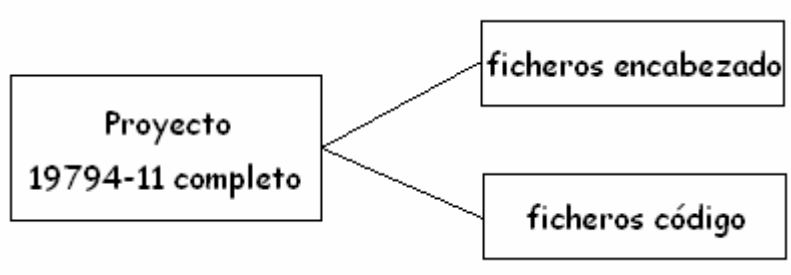



**Figura 25 - Estructura de ficheros de 19794-11**

- Archivos de encabezado
  - Def11normal.h: definiciones de las constantes y las estructuras de la norma.
  - Definiciones.h: definiciones de las constantes generales, los tipos de datos y una estructura auxiliar para almacenamiento de datos de 19794-7
  - Firmlib.h: contiene las librerías utilizadas y los archivos de definiciones

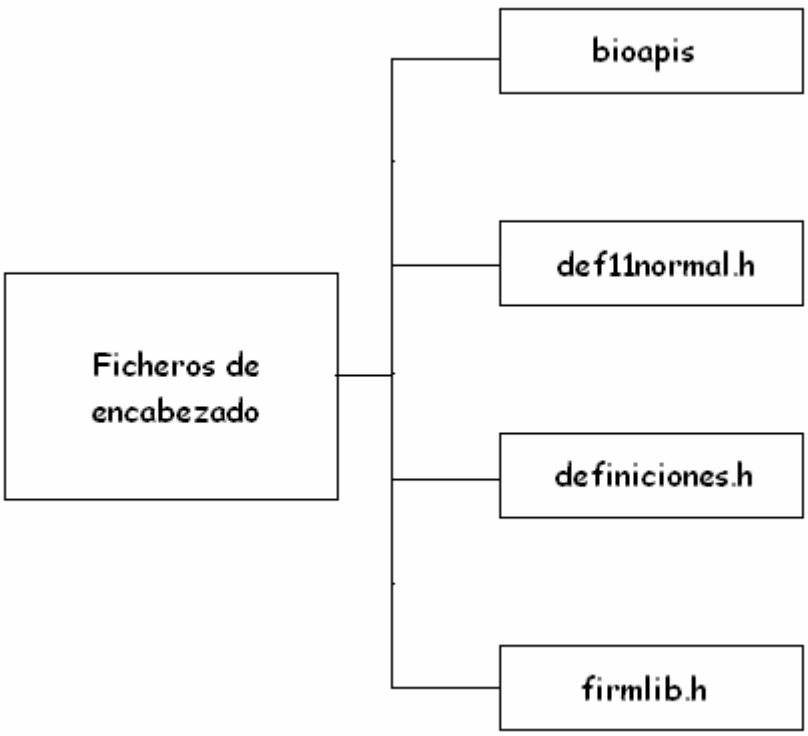



**Figura 26 - Ficheros de encabezado de 19794-11**

- Archivos de código fuente

  - Escribir11normal.c: contiene la implementación de la función principal del programa, "main", desde la que se realiza la llamada al resto de funciones que se encargarán del almacenamiento de las firmas en el formato que se dicta en la norma.

  - Funciones.c: compuesto por las funciones:

    - ISO_7full_to_11: función general desde la que se realizan llamadas al resto de las expuestas con el fin de convertir el fichero que contiene una firma en formato 19794-7 completo al formato 19794-11 y almacenarla en la estructura correspondiente. Se siguen los siguientes pasos

      - Definición e inicialización de variables

      - Lectura del fichero que contiene la firma en formato 19794-7 completo

      - Cálculo de velocidades y aceleraciones

    - Obtención de las posiciones de los puntos singulares, tanto de posición como de presión
    - Obtener y guardar los strokes de posición
    - Obtener y guardar los strokes de presión
    - Calcular y almacenar Overall Data (datos generales de la firma)
    - Con todos los anteriores, confeccionar el cuerpo de la estructura 19794-11
    - Confeccionar cabecera de la estructura 19794-11

- crear_ruta: confecciona la ruta, tanto para el formato 19794-11 como 19794-7 en la que se sitúan los ficheros del estándar correspondiente.
- leer_19794_7_full: lectura de los ficheros de firmas almacenados siguiendo este estándar y almacenamiento de los datos en una estructura adecuada para después operar con ellos y confeccionar los ficheros 19794-11 a partir de él
- leer_datos_nbytes: lectura de los bytes indicados por parámetro de un fichero también dado por el mismo medio.
- puntosing_xy: obtención de los puntos singulares del vector que se indica como parámetro, que debe ser el canal de las coordenadas X o Y. Para ello se comprueban los diferentes valores de los canales y las direcciones que van siguiendo.
- puntosing_f: realiza el mismo proceso descrito para la anterior función, sólo que en este caso para el canal de presión “f”
- guardar_pos_strokes: almacena en una estructura adecuada para ello, donde se guardan todos los strokes de posición, un stroke de este tipo, ya sea del canal X o Y.
- tipo_punto_sin_xy: obtención del tipo de punto singular X o Y. Una vez se encuentra un punto singular se analiza el tipo de éste, en este caso, para los canales X o Y
- tipo_ps_xy: obtención del tipo de stroke para los canales X o Y. Una vez se tiene un stroke, se calcula el tipo en función de los datos.

- guardar_pres_strokes: almacena los datos de presión de un stroke de este tipo en una estructura auxiliar donde se guardan todos los strokes de este tipo para después copiarlos en la estructura que marca el estándar.

- tipo_ps_p: obtención del tipo de stroke de presión.

- coeficiente_correlacion: cálculo del coeficiente de correlación en función de los datos analizados.

- calcular_tamano_muestra: calcula el número de bytes de la firma.

- guardar_fichero_11: almacena en un fichero la firma construida anteriormente. Ésta contiene todos los strokes, tanto de presión como de posición guardados previamente, los datos de cabecera y demás datos. Para ello se apoya en funciones auxiliares tales como “escribe” donde con la función “fwrite” se realiza la escritura de los bytes indicados.

- escribe: como se mencionaba anteriormente, mediante la función “fwrite” escribe los bytes que se le indican por parámetro comprobando previamente si el ordenador tiene una configuración big o little endian

- Lectura_11: lectura de los datos guardados en ficheros con formato 19794-11 y almacenamiento en una estructura marcada por la norma.

- Mostrar_11: una vez se ha creado la estructura con los datos en formato 19794-11, ésta se le pasa por parámetro a esta función y la función que se está describiendo, muestra por pantalla todos los datos de la firma.

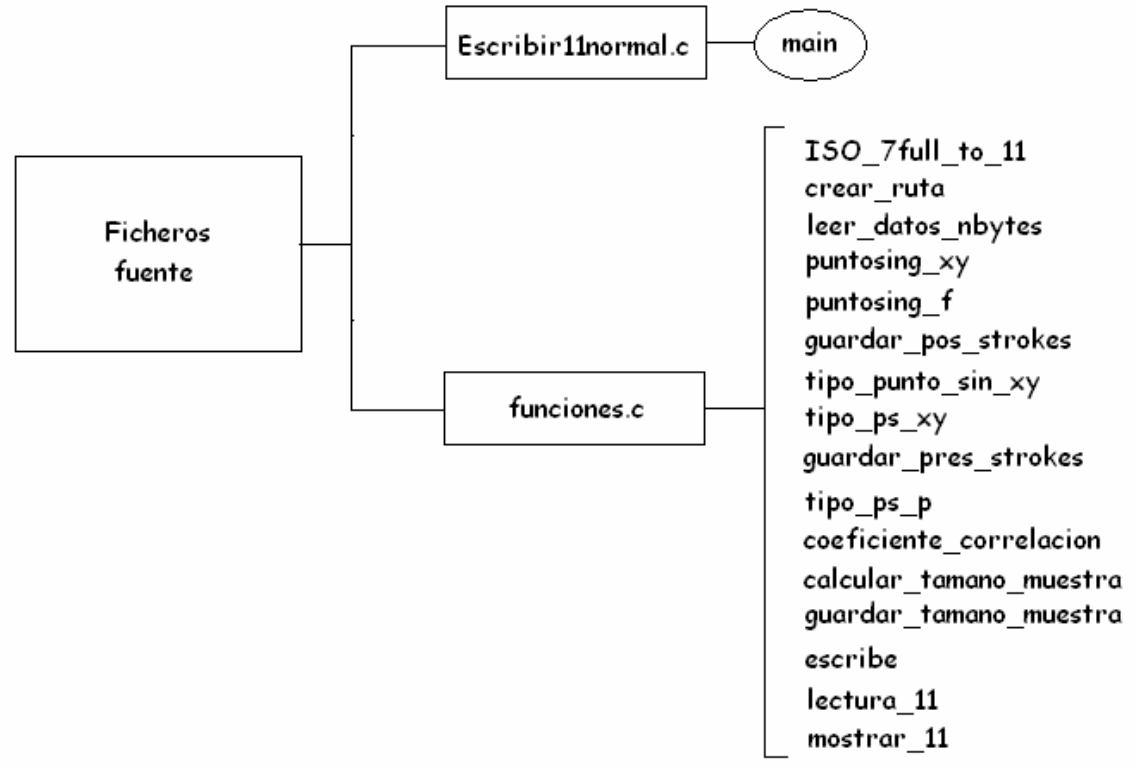



**Figura 27 - Ficheros fuente de 19794-11**

### 4.3.2 Desarrollo del código

La parte principal del código se desarrolla en el fichero "Escribir11normal.c", que contiene la función "main", dividida principalmente en dos bloques, uno de ellos destinado a la creación de los ficheros que contienen las firmas genuinas y el otro bloque para la creación de las falsificaciones.

Con el fin de convertir cada uno de los ficheros de la base de datos MCyT al formato 19794-11 Full, se sitúan ambos bloques dentro de un bucle destinado a recorrer cada uno de los usuarios de la base de datos (100 usuarios)

Cada uno de los bloques mencionados se divide a su vez en tres apartados:

Generación de la firma en formato 19794-11

1. Creación de la ruta donde se encuentran los ficheros en formato 19794-7 Full
2. Lectura de la firma en formato 19794-7 Full para operar con los datos contenidos en ella y confeccionar el nuevo formato 19794-11 Full
3. Cálculo de velocidades y aceleraciones

4. Obtención del los puntos singulares

    3.1. Para cada uno de los puntos singulares correspondientes a X o Y (pen strokes) almacenar datos en la estructura correspondiente

    3.2. Realizar la misma acción que en el caso anterior también para los pressure strokes

5. Cálculo de los datos Overall Data

6. Almacenamiento de la firma en un fichero con formato 19794-11 Full

    5.1. Guardar datos del cuerpo

    5.2. Guardar datos Overall Data

    5.3. Guardar cabecera

<u>Almacenamiento de la firma en fichero</u>

3. Creación de la ruta donde se almacenarán los ficheros en formato 19794-11 Full

4. Creación y apertura del fichero en el nuevo formato en el que se guardarán los datos.

5. Escritura de los datos en el fichero

    3.1. Guardar cabecera

    3.2. Guardar cuerpo

        3.2.1. Guardar pen strokes

        3.2.2. Guardar pressure strokes

        3.2.3. Guardar Overall Data

<u>Lectura del fichero y almacenamiento en estructura 19794-11</u>

1. Creación de la ruta donde se encuentran los ficheros en formato 19794-11 Full para ser leídos

2. Apertura del fichero para lectura de los datos contenidos en él.

3. Lectura de los datos del fichero

3.1. Leer cabecera

3.2. Leer cuerpo

3.2.1. Leer pen strokes

3.2.2. Leer pressure strokes

3.2.3. Leer Overall Data

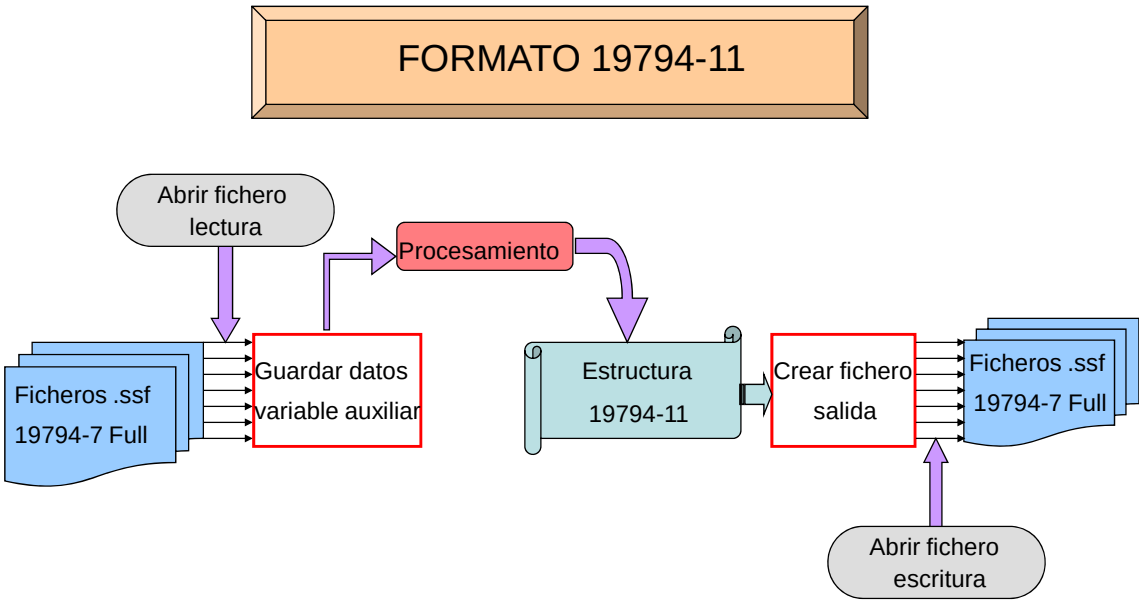



**Figura 28 - Diagrama de bloques de 19794-11**

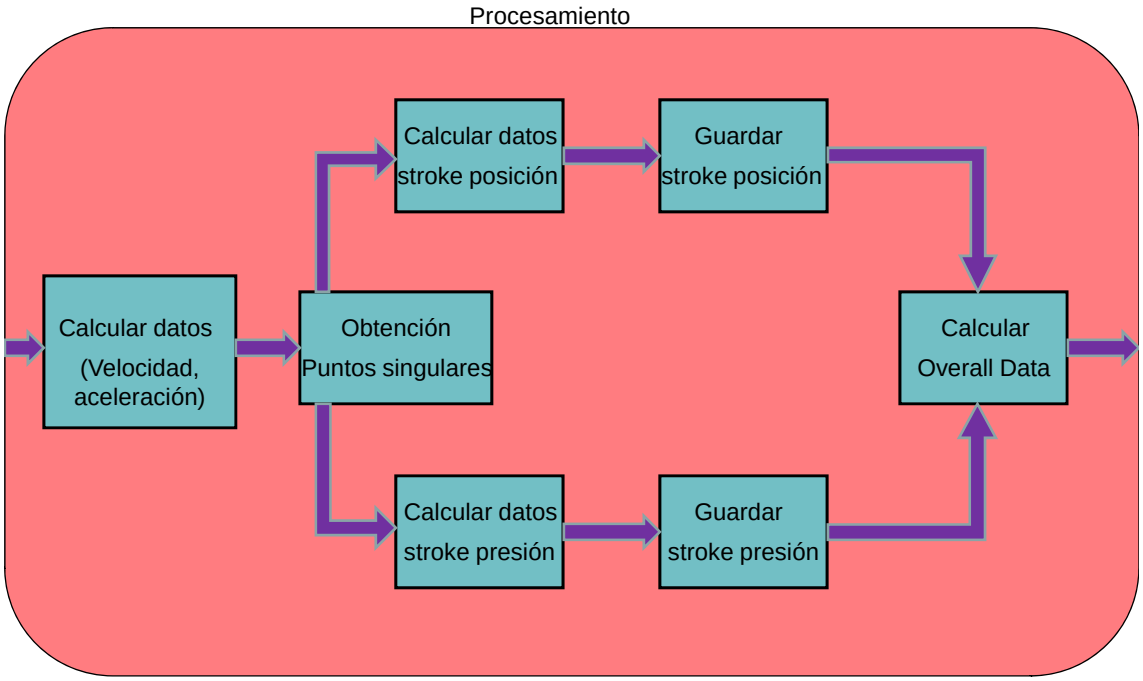



**Figura 29 - Bloque de procesamiento del diagrama de bloques de 19794-11**

Cabe destacar que el programa lleva asociado un control de errores para cada una de las acciones que se llevan a cabo en él

### 4.3.3 Problemas encontrados

1. Uno de los problemas que surge es que al leer la norma no se especifica con claridad el almacenamiento de los strokes de posición y presión, el orden en que deben guardarse. Ambos aparecen entrelazados (un stroke de posición seguido de uno de presión y así sucesivamente), lo cual añadido a que los dos terminan en “n” strokes lleva a pensar que se deben obtener el mismo número de strokes de cada tipo

2. La longitud de los strokes. Solamente hay un campo para indicar la longitud de los strokes, seguramente relacionado con lo comentado en el apartado anterior. De este modo se tiene que identificar la longitud de ambos conjuntos de strokes mediante un único campo.

3. Para la identificación de puntos singulares cuando se encuentran dos valores consecutivos iguales, surge el conflicto de si identificarlos como un cambio de dirección o no

4. Para el tipo de stroke, en el estándar se define un único campo que designa los strokes de presión y posición con número de 1-5. Los valores 1-4 serán asignados para strokes de posición, mientras que 5 será para presión. De esta forma se tienen 4 tipos de strokes de posición y 1 de presión.

5. Para el cálculo de algunos valores de Overall Data como el valor medio, o la desviación estándar se sobrepasa el valor mayor que se puede almacenar en la variable, de manera que no son válidos los cálculos.

6. El cálculo de las velocidades y aceleraciones se realiza por diferencias entre muestras, de modo que se tienen un menor número de muestras y se debe tener el mismo número.

7. Se detectan diversos aspectos que dificultan la implementación derivados de las estructuras de datos, como los mencionados en el punto 1, 2 y 4. Otros problemas surgen a la hora de almacenar los datos, como pueden ser los detallados en los puntos 3, 5 y 6.

### 4.3.4 Soluciones adoptadas

1. Se decide almacenar en primer lugar todos los strokes de posición y seguidamente todos los de presión, ya que no tiene porqué coincidir el número de uno con el del otro, y de esta forma, si se continuaran entrelazando, llegará un momento en que no se podrá debido al distinto número de strokes de uno y otro tipo.

2. En un primer momento se realiza una implementación utilizando dos campos, uno para indicar la longitud de los strokes de presión y otro para los de posición, pero finalmente se decide seguir con un único campo, que indica el número total de strokes, suma de los de posición y presión. Para cada stroke se comprueba el tipo para realizar el conteo.

3. Se adopta la solución de considerar los valores iguales consecutivos como de la misma dirección, de modo que no se produce un stroke.

4. Se decide identificar los strokes por separado, es decir, utilizar un campo para los tipos de strokes de presión y otro campo para los de posición, cada uno identifica las diferentes posibilidades que pueden ocurrir en función de los cambios de dirección. Se siguen teniendo 4 opciones para posición y se amplían a 4 las opciones de presión.

5. Se realiza el cálculo por pasos y se utilizan variables que permiten un mayor rango de almacenamiento. Después se convierte a las variables correspondientes.

6. Se colocan ceros como valores iniciales o finales de las velocidades y aceleraciones, de modo que se considera reposo al inicio o final.

7. Estos factores se le comunican al comité de desarrollo del estándar y éste decide modificar la norma.

# 5 Base de datos biométrica bimodal MCyT

La necesidad actual de grandes bases de datos para evaluar sistemas de reconocimiento biométrico automático ha motivado el desarrollo de bases de datos con gran cantidad de información. Para el desarrollo de las pruebas sobre firmas manuscritas que se llevan a cabo en este proyecto se elige la base de datos bimodal MCYT como fuente de la información de los individuos. Se tienen en consideración una gran cantidad de personas, con significado estadístico, en un procedimiento multimodal, e incluyendo diferentes fuentes de variabilidad que existen en entornos reales.

Uno de los principales problemas que pueden surgir en el desarrollo, testeo y evaluación de sistemas de reconocimiento biométrico, tanto para el modo de identificación como para verificación, es la ausencia de grandes bases de datos biométricos de acceso público formadas a partir de datos reales. Ello es debido a la necesidad de un alto grado de cooperación por parte de los individuos, además de que deben participar un gran número de ellos para obtener datos significativos.

Algunas de estas bases de datos públicas son:

- DB 4 NIST Fingerprint Image Groups
- FFFVC200x fingerprint databases
- Philips Research Laboratorios Signatura Database
- MilDea
- Biosecure

En este contexto, el Biometric Research Laboratory (ATVS) de la Universidad Politécnica de Madrid, ha promovido el proyecto MCYT, en el que se ha realizado el diseño y adquisición de una gran base de datos, incluyendo rasgos de huellas dactilares y firmas. En el proceso de adquisición de datos han participado:

- Universidad Politécnica de Madrid (UPM)
- Universidad de Valladolid (UVA)
- Universidad del País Vasco (EHU)
- Escuela Universitaria Politécnica de Mataró (EUPMt)

La base de datos MCYT incluye huellas dactilares y firmas on-line de cada individuo, obteniendo un número significativo de muestras de cada tipo de acuerdo a diferentes niveles de control, que se definen en función de la variabilidad de cada característica.

El ámbito de utilidad de la base de datos MCYT implica principalmente la evaluación en el diseño de sistemas de reconocimiento automático en aplicaciones civiles, comerciales y forenses, permitiendo el desarrollo y evaluación de algoritmos de

reconocimiento biométrico basados en características biométricas simples, además de la fusión de ellas en un sistema de reconocimiento bimodal.

Uno de los principales objetivos que se pretende lograr en el diseño de la base de datos MCYT es que los datos contenidos en ella sean estadísticamente significativos. Para ello se tienen en cuenta los siguientes aspectos:

- Número de individuos implicados
- Número de modalidades por individuo
- Número de realizaciones (muestras) por cada individuo

A medida que cada uno de los anteriores se hace mayor, también crecerá la generalización de los datos, de modo que normalmente la tendencia es obtener enormes bases de datos biométricos.

Para conseguir lo anterior, en el diseño de la base de datos MCYT se persigue:

- Maximizar el número de individuos manteniendo el número de muestras recogidas por cada usuario dentro de un margen lógico.

- Maximizar el número de muestras recogidas por cada usuario acotando en este caso el número de individuos en una cantidad adecuada

El cuerpo base MCYT, se confecciona por la adquisición de datos sobre 330 individuos en las cuatro instituciones participantes en el proceso. En particular EUPMt, UVA, EHU y UPM han realizado la adquisición de 35, 75, 75 y 145 individuos respectivamente.

El cuerpo base MCYT se divide en dos subcuerpos:

- MCYT_Fingerprint: para cada individuo, se adquieren 12 muestras de cada dedo utilizando dos sensores diferentes (óptico y capacitivo). Por tanto, el número de muestras incluidas será:

$$\mathrm{M} = 330\,\mathrm{indiv} \times 10\,\mathrm{dedos} \times 12\,\mathrm{muestras} \times 2\,\mathrm{sensores} = 79200\,\mathrm{muestras}$$

- MCYT_Signature: se obtienen por cada individuo 25 firmas verdaderas y 25 falsificaciones. Se incluye en la base de datos tanto la información estática (imagen de la firma escrita) como la dinámica (trayectoria, presión y azimut/altitud del bolígrafo). Por tanto, el número de muestras que se almacenará es:

$$\mathrm{M} = 330\,\mathrm{indiv} \times (25\,\mathrm{verdaderas} + 25\,\mathrm{falsificaciones}) = 16500\,\mathrm{muestras}$$

Para cada individuo, la sesión de captura de las características dinámicas de la firma se realiza después de que las huellas digitales se registran en la base de datos. Partiendo de que la adquisición de cada firma on-line se completa de manera dinámica, será

necesaria una tableta gráfica: el dispositivo de adquisición usado es una tableta WACOM, modelo INTOUS A6 USB, cuyas características se exponen a continuación:


**Figura 30 - Dispositivo WACOM**

- Resolución de 2540 líneas por pulgada (100 líneas/mm)
- Precisión de +/- 0.25 mm
- La máxima altura de detección es 10 mm, de modo que también se detectan los movimientos pen-up (levantar el bolígrafo de la tableta)
- Área de captura de 127 x 97 mm
- Rangos de cada secuencia

  - Posición en eje x: $x_t$ [0 - 12700], corresponde a 0 – 127 mm
  - Posición en eje y: $y_t$ [0 - 9700], corresponde a 0 – 97 mm
  - Presión bolígrafo: $p_t$ [0 - 1024]
  - Ángulo azimut bolígrafo-tableta: $\gamma_t$ [0 - 3600] ($0^o$ - $360^o$)
  - Ángulo altitud bolígrafo-tableta: $\rho_t$ [300 - 900] ($300^o$ - $900^o$)

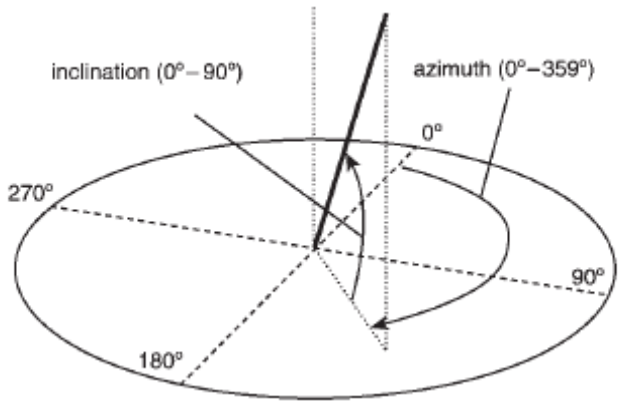



**Figura 31 - Ángulos de azimuth e inclinación del bolígrafo con respecto al plano de la tarjeta gráfica Intuos de Wacom**

La frecuencia de muestreo de las señales adquiridas se coloca a 100 Hz, Cada individuo realiza 25 firmas genuinas (en grupo de 5 muestras) y 25 firmas falsas (5 por cada usuario desde n-1 hasta n-5). Se procede de igual forma para los primeros individuos, pero realizando las firmas falsas los usuarios posteriores.

La detección y segmentación de la muestra de entrada (firma) se realiza de forma automática por el software de adquisición. El comienzo de la firma se produce cuando se detecta el contacto del bolígrafo con la tableta, es decir, la primera muestra con valor de presión distinto de cero. La firma finaliza cuando durante 3 segundos la presión es cero (se produce pen-up), el proceso de captura se detiene, y almacena la firma.

En las siguientes imágenes se exponen algunos ejemplos de firmas. Para cada fila, las tres firmas de la parte izquierda corresponden a un individuo y las tres firmas de la derecha a otro distinto. Dentro de un mismo individuo las dos firmas de la izquierda son genuinas mientras que la de la derecha es una falsificación. Debajo de cada una de las firmas expuestas se encuentra una gráfica que muestra la información on-line almacenada en la base de datos.


**Figura 32 - Ejemplos de firmas del subcuerpo MCYT_Signature**

# 6 Resultados de compresión para formatos 19794-7 Compacto y 19794-11

Como se comentaba en secciones anteriores, a partir del formato 19794-7 completo se generan los formatos 19794-7 compacto y 19794-11, con la intención de almacenar los datos en ficheros con menor número de bytes.

Para la forma compacta el proceso seguido es la eliminación de la cabecera y algunos datos, además de reducir los datos a la mitad de su tamaño mediante ponderación a un rango menor, con lo que se tienen los mismos datos, pero representados con un menor número de bytes y escalados (aquellos canales definidos como 2 bytes de resolución son transformados a 1byte de resolución).

Con respecto a 19794-11, en esta se agrupan los datos mediante bloques o strokes, tanto de presión como de posición, de forma que cada bloque no requiere incluir todos los datos en el fichero, sino que se realizan unos cálculos sobre ellos, de forma que sigan siendo representativos, y así reducir el tamaño de los ficheros. El proceso resulta más tedioso pero como se comprueba a continuación se consigue cierta compresión, aunque con bastante pérdida de información.

A continuación se muestran los resultados del análisis realizado basado en el tamaño medio de cada una de estas versiones y se puede ver como, efectivamente, se consigue el propósito de la reducción del tamaño de los ficheros.

| TAMAÑO EN BYTES | | | |
|---|---|---|---|
| | 7 COMPLETO | 7 COMPACTO | 11 COMPLETO |
| Número de bytes | 4643 | 2109 | 3649 |

**Tabla 2 - Comparativa del número de bytes de las normas**

De forma gráfica se pueden presentar los resultados obtenidos de la siguiente forma:

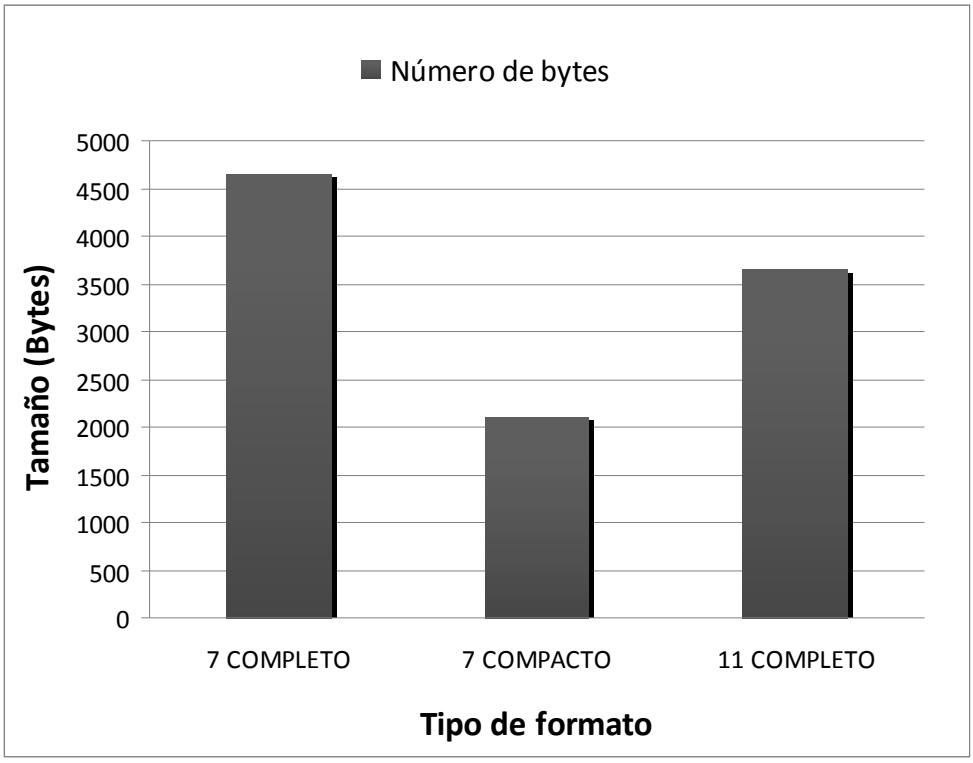



**Figura 33 - Número de bytes de las normas**

Se observa como para el caso de la versión compacta el número de muestras se reduce a más de la mitad, mientras que para el otro caso la compresión obtenida no es demasiado grande con respecto a la información perdida

El formato 19794-7 compacto, como se comentaba en párrafos anteriores, se va a reducir a más del 50% ya que además del tamaño de los datos a la mitad, se tendrán que descontar los bytes de la cabecera y cuerpo eliminados.

Para 19794-11, la reducción del número de bytes se basa en las particularidades de la firma realizada por el usuario en cuestión, ya que los cambios de dirección y presión de la escritura serán los que determinen el número de strokes de presión y posición, y por tanto, el número de bytes que se almacenarán en el fichero, ya que la cabecera y demás datos del cuerpo mantienen constante su longitud. La compresión no es muy alta, se pierde mucha información (dos canales de inclinación, movimiento del bolígrafo en el aire, información entre puntos singulares). Además, se guardan muchos datos en cada stroke teniendo algunos campos repetidos, por ejemplo los de posición inicial y final de los strokes.

Los porcentajes o tasas de compresión se calculan en base a la fórmula:

$$T = \frac{num\_medio\_bytes\_original - num\_medio\_bytes\_comprimido}{num\_medio\_bytes\_original}$$

Se realiza el cálculo para 19794-7 compacto y 19794-11 con respecto a 19794-7 completo, obteniéndose los siguientes resultados:

| COMPRESIÓN ENTRE NORMAS (%) | | |
|---|---|---|
| | 7 COMPACTO | 11 |
| % Compresión | 54,58 | 21,41 |

**Tabla 3 - Tasa de compresión 19794-7 compacto y 19794-11**

De forma gráfica se pueden presentar los resultados obtenidos de la siguiente forma:

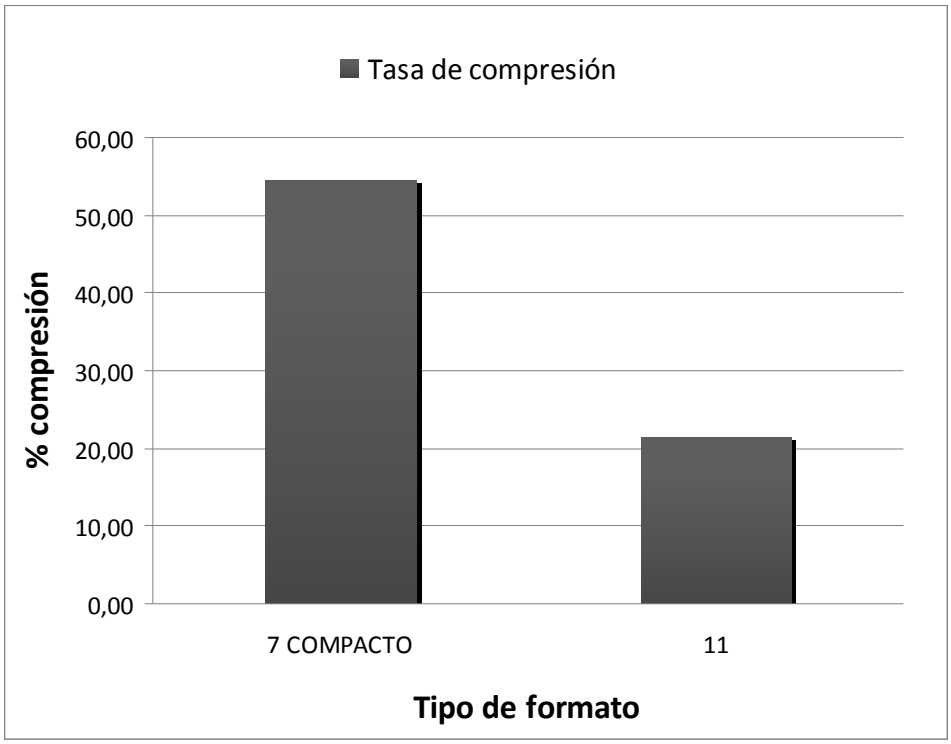



**Figura 34 - Tasa de compresión 19794-7 compacto y 19794-11**

Las tasas de compresión obtenidas son aceptables para el caso compacto, mientras que para 19794-11 no son demasiado buenas. En ambos casos se pierde información en la compresión, en el caso compacto se pierde resolución en las señales, y en el caso de 19794-11 los puntos intermedios y el movimiento del lápiz en el aire

A priori se podría desprender la conclusión de que la implementación compacta es la mejor ya que nos permite representar la misma información con un número mucho menor de bytes, pero esto no es del todo cierto, ya que puede influir de forma negativa la pérdida de resolución por la conversión de 2 bytes a 1 byte si afecta al rendimiento de los algoritmos.

# 7 Introducción a la compresión de datos

## *7.1 Introducción*

El término compresión de datos se define como el proceso que se lleva a cabo para reducir el tamaño de los datos tratables y representar de manera eficaz una cierta información. El objetivo principal de cualquier esquema de compresión es describir la misma información con un menor número de datos, eliminando los elementos redundantes [11].

Cuando se realiza el proceso de compresión se debe tener en cuenta la información que se va a tratar, en base a que los datos pueden ser:

- Redundantes: se refiere a los datos repetitivos o previsibles, que podrán ser eliminados en el proceso de compresión sin ocasionar pérdida de información. Por ejemplo largas cadenas que se repiten de manera periódica en un fichero.

- Irrelevante: no afecta al contenido y no aporta información, ya sea porque no se puede apreciar o porque carece de interés. Su eliminación no afecta al significado de los datos. Como ejemplo, mencionar el envío de sonidos a una frecuencia no perceptible por el oído humano.

- Relevante: es la información que debe ser incluida si no se quieren perder datos. En caso de suprimirla ocasiona que el nuevo fichero carezca de la totalidad de la información original, lo cual implica la consecuente pérdida de calidad con respecto al inicial.

La importancia de la compresión reside principalmente en dos aspectos

- Reducir el espacio de almacenamiento: en la actualidad existen dispositivos con un gran volumen de almacenamiento de información (discos duros principalmente) donde guardar los datos, pero también es cierto que la velocidad de crecimiento de estos dispositivos es similar a la de crecimiento de la información. Por otro lado, existe la necesidad del transporte de los datos, mediante medios de menor capacidad (memorias flash, DVDs…) para lo cual es de gran interés la compresión de la información, ya que de esta forma se podrán transportar un mayor volumen de datos.

- El segundo aspecto nace como consecuencia de los avances tecnológicos, que han llevado a grandes redes de comunicaciones, como por ejemplo Internet. El envío y recepción de archivos por estas redes se ha convertido en un trabajo diario y el tiempo en que estas acciones se producen dependen

directamente del tamaño de los ficheros que se tratan. Para reducir este tiempo, emplearlo de forma más eficiente y reducir costes, la compresión es una herramienta imprescindible.

Un proceso de compresión lleva asociado un proceso de descompresión por parte del receptor, con el que se pretende recuperar los datos originales para ser utilizados. En base a cómo se realiza el proceso, los compresores se pueden clasificar de la siguiente forma [12]:

- Según el tiempo necesario para comprimir/descomprimir
    - Simétricos: el proceso de descompresión y compresión consumen una cantidad de tiempo similar. Se utiliza para transmisiones en tiempo real, como por ejemplo videoconferencias, voIP o aplicaciones de telemetría.
    - Asimétricos: la diferencia temporal entre ambos procesos es muy grande, siendo el tiempo de compresión mucho mayor que el de descompresión. Su utilización se basa en el tratamiento de grandes volúmenes de datos, con la particularidad de que el proceso de compresión se realiza una única vez, mientras que el de descompresión se lleva a cabo en múltiples ocasiones. Un ejemplo de ello es la compresión de vídeo (MPG, MPEG, DivX…), que es un proceso lento pero una vez finalizada la compresión se puede reproducir tantas veces como se desee, siendo la descompresión de mucha menor complejidad.
- Según el modelo utilizado
    - No adaptativo: se define el modelo con el que se va a operar antes del proceso de compresión y se mantiene constante, con lo cual no es necesario el reconocimiento previo de los datos. Su característica principal es la rapidez, pero no se logran buenas razones de compresión.
    - Semi-adaptativo: se realiza un análisis previo de los datos para definir posteriormente un modelo basado en este estudio. Tiene el inconveniente de que los datos deben estar almacenados para poder analizarlos, con lo que el tiempo crece y es necesario mayor espacio.
    - Adaptativo: en este caso el modelo se va actualizando a medida que se realiza la compresión de datos, teniendo diferentes modelos en función de la entrada que cambian de manera dinámica. Con éste se obtienen mejores razones de compresión y no es necesario el almacenamiento previo de la información. Por otro lado es más lento y su implementación es más compleja.

- En función de la información transmitida:

    - Compresión con pérdida de información: una vez terminado el proceso de compresión, los datos generados no son los mismos que al inicio, se ha producido un deterioro o pérdida de algunos de ellos, resultando imposible su recuperación, con lo cual la integridad de la información original se ve alterada. Se eliminan elementos relevantes.

    - Compresión sin pérdida de información: la integridad de los datos iniciales queda inalterada, no se produce pérdida alguna, a pesar de la reducción del tamaño del fichero resultante. Se elimina información redundante o no relevante.

El modelo general seguido en un proceso de compresión de datos se puede resumir mediante el esquema de bloques [13]:

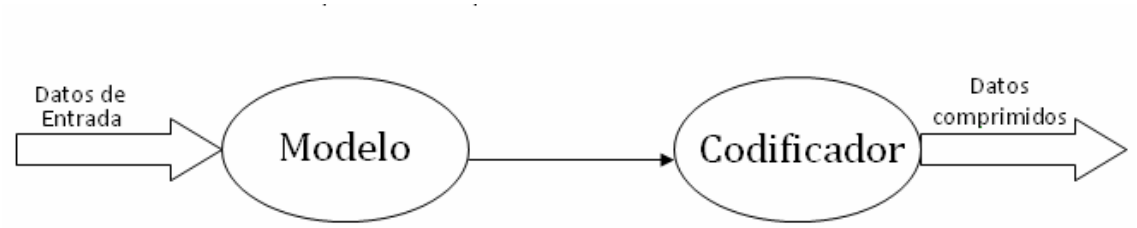



**Figura 35 - Esquema de codificación**

Hay una gran diversidad de algoritmos de compresión, cada cual diseñado a medida para cada tipo de dato. El resultado es un fichero en un formato específico, con un tamaño igual o menor que el original, dependiendo del algoritmo empleado. Cabe reseñar que la aplicación de un algoritmo no adecuado para los datos tratados puede llevar a un mayor tamaño del archivo comprimido, ya que para el caso tratado puede no resultar eficiente su empleo, siendo inútil el trabajo desarrollado.

## 7.2 Algoritmos de compresión con pérdida de información

Se denominan algoritmos de compresión con pérdida (lossy compression algorithm) a cualquier procedimiento de codificación que tenga como objetivo representar cierta cantidad de información utilizando una menor cantidad de la misma, resultando imposible una reconstrucción exacta de los datos originales. Dicha pérdida de datos se compensa con la obtención de muy altas tasas de compresión.

El uso de este tipo de compresión se reduce a los casos en los que no es imprescindible el conjunto total de los datos originales para que la información tenga sentido. La información reconstruida es una aproximación de la original, lo cual implica una reducción de la calidad de los archivos comprimidos.

Su uso está muy extendido para vídeo, imágenes y sonido, ya que su alta tasa de compresión reduce en gran medida el gran tamaño que ocupan y la pérdida de calidad apenas es perceptible. Algunos ejemplos son MPEG, JPEG y MP3 respectivamente.

Existen dos técnicas comunes de compresión con pérdida de información [14]:

- Transformación: se realiza una reducción de los datos mediante la transformación de los originales, resultando imposible la obtención de los iniciales de nuevo

- Predictivo: se realiza un análisis de los datos originales con el objetivo de predecir el comportamiento de los mismos. Seguidamente se hace un estudio comparativo de la predicción realizada contra los datos iniciales, para después realizar un análisis de errores y la información requerida para la reconstrucción.

## *7.3 Algoritmos de compresión sin pérdida de información*

Se denomina algoritmo de compresión sin pérdida (lossless compression algorithm) a cualquier procedimiento de codificación que tenga como objetivo representar cierta cantidad de información sin utilizar una menor cantidad de la misma, siendo posible una reconstrucción exacta de los datos originales [15]. De esta forma se reduce el volumen de datos con respecto al original pero no se pierde información. La desventaja es que la tasa de compresión obtenida tiene un límite asociado a la entropía o redundancia de datos de la fuente inicial.

Este tipo de compresión se vuelve necesaria cuando se requiere conservar íntegramente la información original, por ejemplo en archivos ejecutables o tablas de bases de datos, siendo reversible el proceso de compresión.

Este sistema de compresión se usa en compresores de archivo (RAR, Gzip, Bzip, zip, 7z, ARJ, LHA) y de disco, también en imágenes (PNG, RLE) y en algún formato de audio (FLAC, WAV)

En el presente documento se estudian los algoritmos descritos a continuación, que serán los que se apliquen a los datos de las firmas generadas por el formato 19794-7, tanto en el formato completo como el compacto, para la realización del estudio de los resultados de compresión de la información de la firma

### 7.3.1 RLE (Run-Length Encoding)

El algoritmo RLE (Run-Length Encoding) es una forma de compresión basada en el análisis de repetición de información. Una secuencia de datos iguales se representa como el propio dato en cuestión y el número de repeticiones del mismo.

De lo mencionado con anterioridad se puede extraer la conclusión de que el algoritmo será tanto más efectivo cuanta mayor repetitiva sea la información a comprimir. Si este no es el caso, incluso puede obtenerse un mayor tamaño en lugar de una reducción, de modo que su uso debería ser exclusivamente para datos muy repetitivos y de esta forma se logre un uso eficiente.

A continuación analizan dos cadenas de caracteres para ilustrar el uso de dicho algoritmo:

Cadena 1: “RRRRRROOOOOOBBBBBBBBEEEEEEERRRRRTTTTTTOOOO”

Cadena 2: “ROBERTOPIZARROSANTOS”

Si se aplica el algoritmo RLE a ambas cadenas se obtienen los siguientes resultados:

Compresión cadena 1: R6O6B8E7R5T6O4

Compresión cadena 2: R1O1B1E1R1T1O1P1I1Z1A1R2O1S1A1N1T1O1S1

Para el primer caso se obtiene un resultado positivo, debido a la repetitividad de los datos a comprimir. La primera cadena tiene un tamaño total de 42 caracteres y una vez comprimida su extensión alcanza los 14 caracteres, lo cual implica una gran compresión y un uso bastante eficiente del algoritmo.

Por otro lado, en el segundo caso queda de manifiesto su falta de efectividad ante datos muy diversos. La longitud de la segunda cadena es de 20 caracteres, mientras que después del proceso de compresión el número de caracteres ha crecido de manera notable, alcanzando el valor de 38. De esta forma se comprueba que para datos dispersos, no se comprime sino que se realiza el proceso contrario, no obteniéndose el propósito que se persigue.

La ganancia de compresión viene dada por la expresión [16]:

$$G = \frac{Núm\_caract\_antes\_compresión - Núm\_caract\_después\_compresión}{Núm\_caract\_antes\_compresión}$$

Para el caso del ejemplo se obtiene:

Cadena 1: $G = \frac{42-14}{42} = 0.667 = 66.7\%$

Cadena 2: $G = \frac{20-38}{20} = -0.9 = -90.0\%$

A la vista de los resultados, para la cadena 1 se logra aproximadamente una ganancia de compresión del 70% mientras que para el segundo caso en lugar de comprimir, lo que se consigue es aumentar el tamaño de la cadena a estudio en un 90% (de ahí el signo negativo), siendo la longitud de ésta casi el doble de la original.

Otro análisis que puede hacerse es el emitido por el número de bytes del fichero original por cada byte comprimido (N:1) o cuántos caracteres puedo representar con un solo carácter codificado:

$$N = \frac{Núm\_caract\_antes\_compresión}{Núm\_caract\_después\_compresión}$$

Para el caso del ejemplo se obtiene:

Cadena 1: $N = \frac{42}{14} = 3$

Cadena 2: $N = \frac{20}{38} = 0.53$

Como se observa en el primer caso, por cada tres caracteres del mensaje original, se obtiene uno codificado, es decir con un solo carácter podemos sustituir a tres del inicial. Para el siguiente, se necesitan aproximadamente dos caracteres codificados para representar uno del mensaje inicial, lo cual desprende la no efectividad del algoritmo para éste último.

Del estudio previo realizado se desprenden las siguientes ventajas:

- Fácil de implementar
- Rapidez en el proceso de compresión/descompresión
- Baja memoria temporal necesaria

Como inconvenientes se detallan los siguientes

- Sólo es eficiente para información con bastantes datos repetitivos consecutivos
- Las razones de compresión son tanto más altas cuanto más repetitiva es la información, pero por regla general no se consiguen tasas tan altas como en el caso de algoritmos de compresión con pérdidas

### 7.3.2 LZSS (LZ77)

Es un algoritmo desarrollado por James Storer y Thomas Szymanski en 1982 basado en el LZ77 descrito por Abraham Lempel y Jacob Ziv en 1977. Como se aprecia, los nombres de ambos tienen su origen en las iniciales de sus creadores y el año de implantación

Éste último es un algoritmo de codificación sin pérdidas basado en diccionario, lo cual quiere decir que para codificar una cadena determinada lo hace mediante una referencia a una localización del diccionario de la misma cadena. La codificación consta de dos elementos, un desplazamiento sobre el diccionario y la longitud a contar a partir de este desplazamiento, aún cuando no existan repeticiones. También suele escribirse el siguiente carácter a ser analizado. De esta forma para codificar una cadena, se busca su coincidencia dentro del diccionario, si ésta se encuentra, se codifica como el lugar dentro del diccionario en el que se produce y la longitud de dicha cadena coincidente.

El diccionario contiene los últimos “N” caracteres codificados/decodificados. A medida que se van analizando caracteres nuevos, el diccionario va actualizando sus valores deslizando su contenido, por este motivo se le conoce como ventana deslizante.

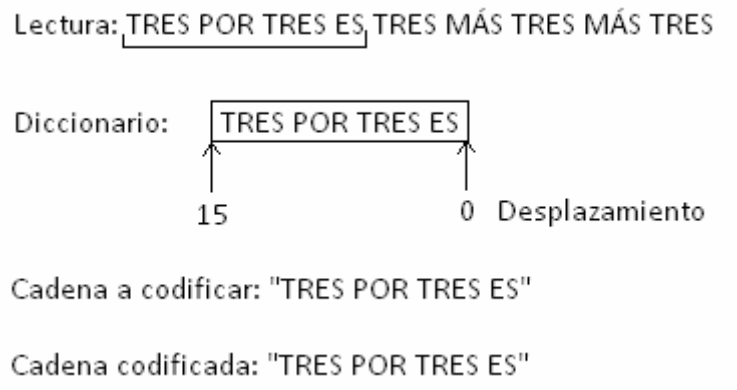
Un defecto reseñable del algoritmo LZ77 es que puede ocurrir el caso de que la referencia al diccionario no siempre es más corta que la cadena sin codificar, lo cual lleva a obtener un mayor tamaño de los datos comprimidos que los originales, no obteniendo los resultados esperados en ese caso.

La evolución del LZ77, llamado LZSS introduce como mejora principal el hecho de que en ningún caso la referencia al diccionario va a ser mayor que la cadena sin codificar, obteniendo en cualquier caso un resultado positivo al comprimir, entendiendo por tal, una reducción del tamaño comprimido con respecto al inicial.

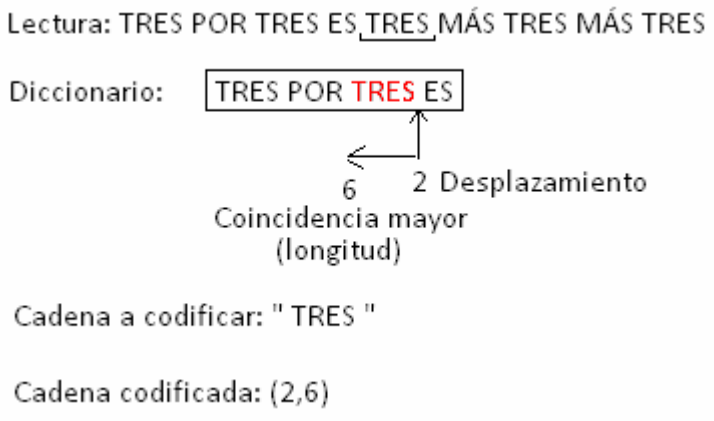
LZSS corrige el hecho de que cuando hay un único carácter que no se encuentra repetido, éste, al ser codificado ocupa mayor espacio que sin codificar. Para ello introduce una bandera, indicando si lo que viene a continuación es un carácter o una representación de codificación, de forma que se evita el problema anterior [17].

Por ejemplo, para codificar la cadena “TRES POR TRES ES TRES MÁS TRES MÁS TRES” y suponiendo que en el diccionario es de cuatro bits, el proceso seguido sería:

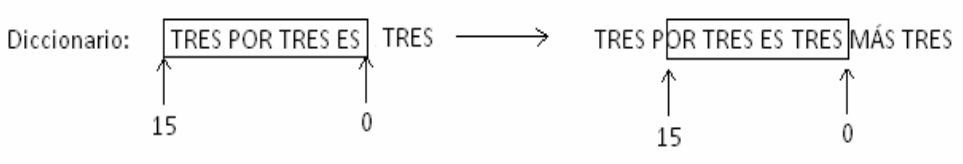
- Lectura de los 16 primeros caracteres de la cadena (4 bits) para inicializar el diccionario. Para codificar estos caracteres se dejan literalmente como están.



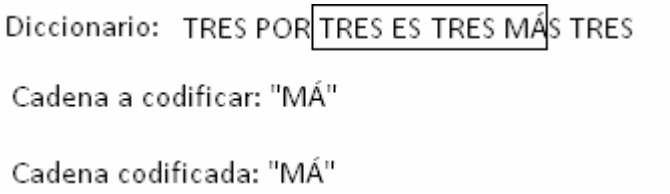

- Se continúan leyendo caracteres y se busca la coincidencia más larga dentro del diccionario. Se codifica por el desplazamiento dentro del diccionario y la longitud de la coincidencia, es decir (2,6)



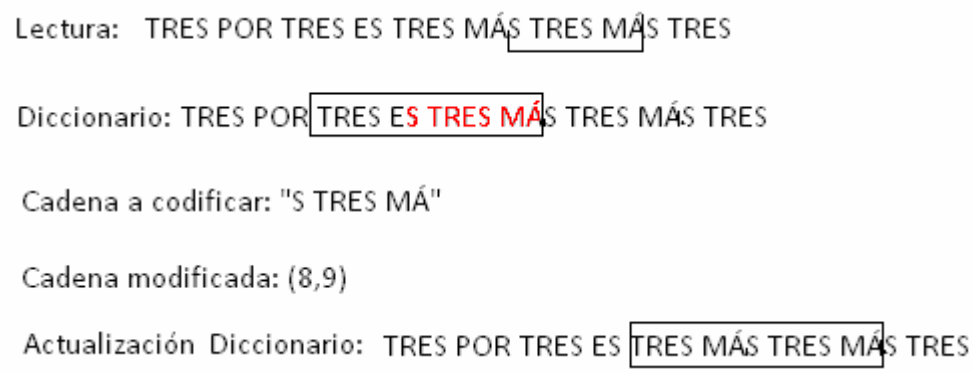

- Actualización del diccionario mediante ventana corredera:



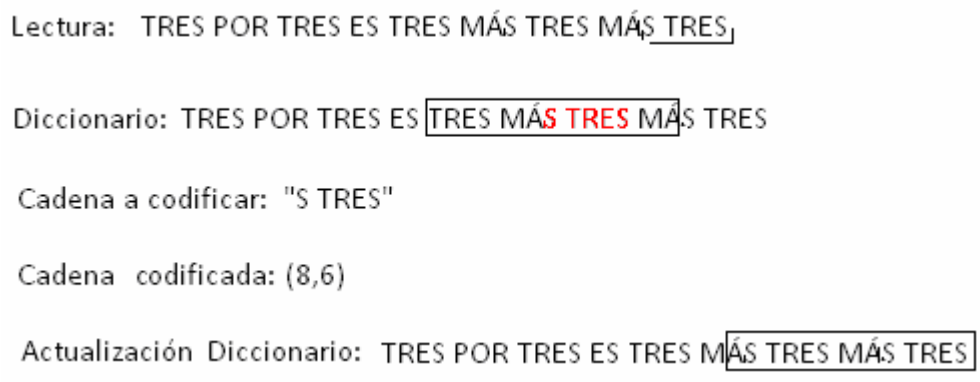

- Los siguientes dos caracteres (“MÁ”) no aparecen en el diccionario de forma que se codifican literalmente y se incluyen en el diccionario.




- Siguiendo con el proceso:



- Y para terminar:




La cadena codificada: “TRES POR TRES ES(2,6)MÁ(8,9)(8,6)”

Para codificar cada dupla (d,l) se necesitan un byte (4 bits para cada uno) al igual que para cada carácter. Además de ello se debe sumar un bit de bandera por cada símbolo para distinguir entre si es un carácter literal o un par codificado. Para la cadena de inicialización no es necesario sumar este bit. De modo que el tamaño de la cadena de codificación será de 21 bytes + 5 bits o 21*8 + 5 = 173 bits

Por otro lado, la cadena de entrada original ocupa un tamaño de 39 bytes o 39*8 = 312 bits.

Para el cálculo de la ganancia de compresión, como se indicaba anteriormente:

$$G = \frac{bits\_antes - bits\_después}{bits\_antes} = \frac{312 - 173}{312} = 0.445 = 44.5\%$$

Se obtiene un resultado de 44.5 % de ganancia de compresión, si la cadena hubiera tenido mayor longitud, manteniéndose aproximadamente la frecuencia de coincidencias, el porcentaje hubiera crecido notablemente, ya que la inicialización del diccionario hace que no se compriman los 16 primeros bytes, sino que se quedan igual, de modo que pierde eficiencia al inicio. Para cadenas largas mejorará esta eficiencia.

Utilizando la expresión descrita en el apartado anterior para el número de bytes del fichero original por cada byte comprimido (N:1) o cuántos caracteres puedo representar con un solo carácter codificado:

$$N = \frac{bits\_antes\_compresión}{bits\_después\_compresión} = \frac{312}{173} = 1.8$$

Como ventajas de este algoritmo se pueden mencionar las siguientes:

- Es muy usado
- Fácil de implementar
- Bastante eficiente.

Los inconvenientes en este caso son:

- Requiere un proceso de búsqueda para encontrar la cadena coincidente más larga en cada caso.
- Necesita almacenamiento extra para guardar el diccionario
- El tiempo aumenta debido al procesamiento para actualizar el diccionario y la búsqueda de cadenas en él

### 7.3.3 LZW (LZ78)

Es un algoritmo desarrollado por Terry Welch en 1984 basado en el LZ78 descrito por Abraham Lempel y Jacob Ziv en 1978. Al igual que en el caso anterior, los nombres de ambos tienen su origen en las iniciales de sus creadores y el año de implantación. También es un algoritmo basado en diccionario pero en este caso se evita el problema que generaba la ventana deslizante, en la que se tenían en cuenta los últimos N caracteres para el diccionario. Ahora se elimina ésta y se construye un diccionario en el que se almacenan progresivamente las cadenas.

Para codificar una cadena con LZ78 se comprueba si está en el diccionario, si es así, se codifica mediante el identificador asignado por el diccionario y el siguiente carácter de la cadena de entrada (i, C). Después de esto, se introduce la cadena formada por la dupla anterior en el diccionario como una nueva entrada y con nuevo identificador.

La evolución de LZ78, LZW, introduce un nuevo concepto para mejorar el rendimiento. Éste consiste en eliminar "C", de forma que cada cadena introducida en el diccionario se representa únicamente mediante un identificador. Para lograr esto, en primer lugar se debe inicializar el diccionario, de forma que contenga una entrada por cada carácter posible, a cada uno de los cuales se le asigna un identificador

El diccionario se irá incrementando mediante la introducción de nuevas entradas conforme se vaya analizando los diferentes caracteres de la cadena a comprimir, por la concatenación de ellas. Las cadenas que se vayan construyendo y no estén añadidas se introducen en el diccionario y se le van asignando nuevos identificadores [18].

Si se quiere codificar por ejemplo la cadena: "ROBEPROBE", suponiendo que las posibles entradas son el alfabeto español, en letras mayúsculas, es decir un total de 27 caracteres:

- Inicialización del diccionario: una entrada por cada carácter en mayúsculas del alfabeto, es decir 27 caracteres con los identificadores como sigue: 1=A, 2=B …. 27=Z

- Comienza el proceso de codificación:

  Lectura: primer carácter (“R”). Ya está contenido en el diccionario, no genera un nuevo código.

- Lectura de “O”, se concatena a “R” y forma la cadena “RO”, que se añade al diccionario con identificador 28. Se codifica “R” como 19.

- Lectura de “B”, se concatena a “O” y forma la cadena “OB”, que se añade al diccionario con identificador 29. Se codifica “O” como 16.

- Lectura de “E”, se concatena a “B” y forma la cadena “BE”, que se añade al diccionario con identificador 30. Se codifica “B” como 2.

- Lectura de “P”, se concatena a “E” y forma la cadena “EP”, que se añade al diccionario con identificador 31. Se codifica “E” como 5.

- Lectura de “R”, se concatena a “P” y forma la cadena “PR”, que se añade al diccionario con identificador 32. Se codifica “P” como 17.

- Lectura de “O”, se concatena a “R” y se forma la cadena “RO” que ya se encuentra en el diccionario, por tanto no se incluye, se siguen añadiendo caracteres a ésta para formar una cadena mayor y entonces sí que se añadirá. No se codifica “R” porque ya está codificado, se buscará coincidencia mayor.

- Lectura de “B”, se concatena a “RO” y forma la cadena “ROB”, que se añade al diccionario con identificador 33. Se codifica “RO” como 28.

- Lectura de “E”, se concatena a “B” y forma la cadena “BE”, que ya se encuentra en el diccionario, no se añade. No se codifica.

- Se codifica la última cadena “BE” como 30

- La salida obtenida es “(19)(16)(2)(5)(17)(28)(30)”. Se compone de 7 caracteres mientras que la entrada tenía 9, ya se puede comprobar como al ser una cadena pequeña se obtiene compresión. Al crecer la longitud de la cadena se vuelve más eficiente, al encontrar un mayor número de cadenas en el diccionario.

El proceso seguido se puede resumir en:

| Entrada (Lectura) | Cadena a buscar | ¿Dentro de diccionario? | Cadena guardar diccionario | Cadena codificada |
|---|---|---|---|---|
| R | R | SÍ | - | - |
| O | RO | NO | RO (28) | R (19) |
| B | OB | NO | OB (29) | O (16) |
| E | BE | NO | BE (30) | B (2) |
| P | EP | NO | EP (31) | E (5) |
| R | PR | NO | PR (32) | P (17) |
| O | RO | SÍ | - | - |
| B | ROB | NO | ROB (33) | RO (28) |
| E | BE | SÍ | - | - |
| - | - | - | - | BE (30) |

**Tabla 4 - Proceso seguido para codificación LZW**

Ventajas

- Es muy utilizado debido a sus buenos resultados.
- Es muy eficiente y para cadenas largas va ganando más eficiencia cada vez.
- El diccionario no es de ventana deslizante sino que va guardando todas las posibles cadenas, lo cual implica que a medida que va creciendo se tendrán más probabilidades de mayor compresión.

Inconvenientes

- El tamaño del diccionario crece con el tamaño de los datos a comprimir. Llegará un momento en que se deba acotar su tamaño, ya que para cadenas muy largas puede crecer de manera indefinida.
- El tiempo de búsqueda de las coincidencias en el diccionario, depende del tamaño de éste, que crece como se comentaba en el punto anterior, de forma que se puede incurrir en búsquedas que consuman una gran cantidad de tiempo.
- Dependiendo del tamaño del diccionario, se codificará cada identificador con un mayor o menor número de bits, lo cual provoca que se pueda perder capacidad de compresión para cadenas cortas, ya que se representan con muchos bits.

# 8 Implementación de la compresión

## *8.1 Modificaciones introducidas a 19794-7*

Para mejorar los resultados de la compresión de datos, se manejan diferentes formas de almacenamiento de la información generada a partir de la base de datos MCyT y confeccionada en función a las diferentes normas de firmas.

Estas diferentes versiones, permiten un análisis del aprovechamiento de los diferentes algoritmos de compresión en base al orden de los datos dentro de los ficheros. Éstos contienen los mismos datos para todas las versiones, variando solamente el orden en que se almacenan, además puede variar el rendimiento de los algoritmos de compresión dependiendo de la manera en que se ordenen dichos datos

Se realiza la composición de cada una de las versiones, además de la versión tal y como se define en el estándar, tanto para el formato 19794-7 completa como para el compacto, pero estas versiones sólo se confeccionan para realizar la compresión de los datos mediante algoritmos de compresión sin pérdidas que se analizarán posteriormente. La cabecera no se comprime, tan sólo los datos. De modo que las firmas de estas versiones estarán formadas por la cabecera intacta y los datos comprimidos

La implementación de cada una de las anteriores es un proyecto distinto en lenguaje C, en el que sólo varían los directorios de almacenamiento, los nombres de los ficheros de salida y la función que guarda los datos en los ficheros, en función al orden seguido seguirá un proceso u otro de ordenación.

Las modificaciones introducidas tanto a la versión completa como a la compacta de la norma 19794-7 (ya que para 19794-11 no se realizan cambios debido a que no se le aplica compresión a los datos) son tres, denominadas como V0, V1 y V2

### 8.1.1 Versión V0

Los datos se almacenan en ficheros conforme se indica en el estándar, es decir, se guarda una secuencia de puntos, cada uno de los cuales contiene los valores de todos los canales incluidos correspondientes a dicho punto. Se tienen tantos bloques como muestras tenga la secuencia de puntos, y dentro de la misma tantos datos como canales incluidos

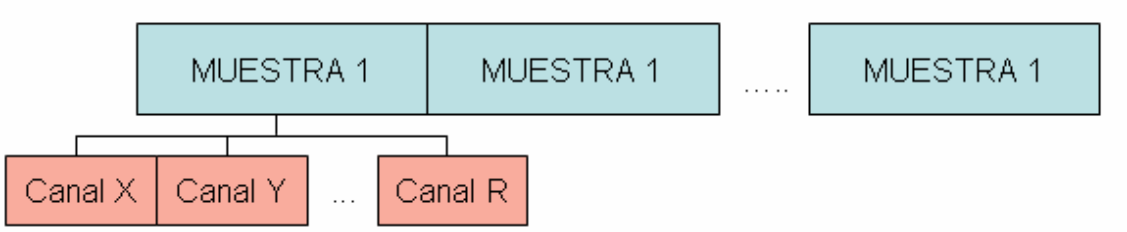



**Figura 36 - Esquema de almacenamiento de datos con versión V0**

### 8.1.2 Versión V1

Se almacena cada canal de forma independiente, es decir, los datos se ordenan por canales, todos los datos de un mismo canal se agrupan y se almacenan, y así se sigue el proceso con todos los demás, uno a continuación de otro. Se tienen tantos bloques como canales incluidos, y cada uno de ellos se compone de tantos valores como contenga la secuencia de puntos.

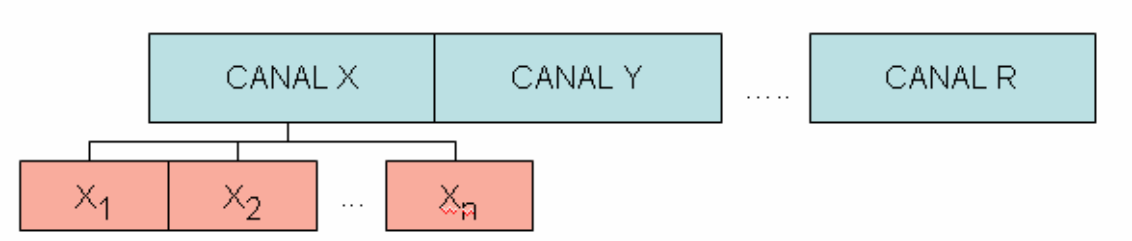



**Figura 37 - Esquema de almacenamiento de datos con versión V1**

### 8.1.3 Versión V2

En este caso se siguen considerando los canales de forma independiente como el caso anterior, pero con la particularidad de que no se no se guardan los datos en sí, sino que se almacena la diferencia entre muestras consecutivas del mismo canal, es decir, la diferencia de la muestra siguiente con respecto a la actual. Se sigue el proceso para todos los canales incluidos, y se colocan uno tras otro. Cabe mencionar que el primer valor de cada canal será él mismo y los demás se obtendrán como se ha explicado, ya que al obtener diferencias tendríamos un valor menos, es decir, si tenemos "n" muestras, tendríamos "n-1" valores.

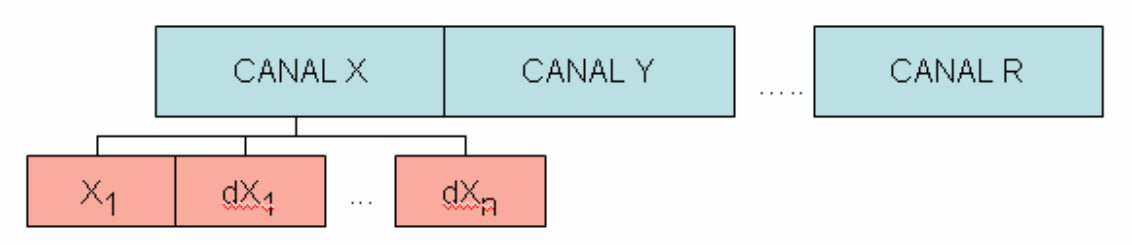



**Figura 38 - Esquema de almacenamiento de datos con versión V2**

## 8.2 Proceso de compresión de ficheros

En el desarrollo de este proyecto se han incluido tres algoritmos de compresión diferentes: RLE, LZSS y LZW. La implementación en código C se ha descargado de la página web de libre distribución de Michael Dipperstein [19].

Las implementaciones mencionadas se desarrollan bajo el entorno Unix, por lo que para su compilación y ejecución se utilizará un entorno (Cygwin) que simula el sistema operativo Linux, bajo una máquina Windows.

Para la ejecución de la compresión se deberán seguir los siguientes pasos:

- Instalar el programa de libre distribución "Cygwin" [20] para crear un entorno virtual de Linux y de este modo poder ejecutar los programas que implementan los algoritmos de compresión realizados bajo este sistema operativo. Cabe destacar que durante la instalación de dicho programa se deberán añadir a la instalación los paquetes "gcc" y "make" situados dentro del grupo "Devel" para la correcta compilación y ejecución. El programa se instalará en C:\cygwin
- Copiar el directorio "compresión" en el directorio "C:\cygwin\home\"
- Ejecutar "Cygwin" y seleccionar el directorio en el cuál se encuentran los archivos para compilar y ejecutar el algoritmo de compresión correspondiente. Por defecto el programa comienza en el directorio "C:\cygwin\home\usuario", "usuario" representa el usuario del ordenador, de modo que para mayor comodidad:
    - Colocar las carpetas con los ficheros de los algoritmos de compresión (rle-0.3, lzss-0.6 y lzw-0.5) en "C:\cygwin\home\usuario\algoritmos_compresion"
    - Las carpetas que contienen los archivos a ser comprimidos (19794_7_V0, 19794_7_V1 y 19794_7_V2) en el directorio "C:\cygwin\home\usuario\ficheros"

  Por tanto para entrar en los directorios de los algoritmos de compresión desde el inicio de cygwin, se ejecutarán los siguientes comandos (según corresponda en cada caso)
    - Si queremos entrar en el algoritmo Run-Length Encoding: "cd algoritmos_compresion/rle-0.3/"
    - Para LZSS (LZ77): "cd algoritmos_compresion/lzss-0.6/"
    - Para LZW (LZ78): "cd algoritmos_compresion/lzw-0.5/"

- Una vez dentro de cada carpeta de los algoritmos, para cada uno de ellos:
    - teclear el comando “make” para compilar y generar el ejecutable
    - Antes de ejecutar el Shell (archivo “.sh”), abrir el archivo con el programa “ultraedit” y revisar que los directorios son los correctos (variables “fich_ent” y “fich_sal”) y sino adaptarlos a cada ordenador
    - Ejecutar el programa Shell correspondiente a cada algoritmo mediante el comando “./comprimir_XXX.sh”, donde “XXX” tendrá los valores
        - “rle”: para el algoritmo Run-Length Encoding [21]
        - “lzs”: para algoritmo LZ77(LZSS) [22]
        - “lzw”: para algoritmo LZ78 (LZW) [23]

      Cabe destacar que cuando se ejecuta el Shell, por defecto crea los ficheros comprimidos para la versión “V0” (datos guardados como en la norma). Si se quiere obtener la compresión de la versión “V1” o “V2”, se deberá modificar el Shell cambiando en las variables “dir_datos7” y “dir_comprimido” el “0” final por “1” o “2” respectivamente.
- Una vez realizados los pasos anteriores, tendremos los ficheros comprimidos en el directorio “C:\cygwin\home\usuario\ficheros_comprimidos\XXX”, donde XXX puede tomar los valores “RLE”, “LZS” o “LZW”. Dentro de cada carpeta se encuentran otras tres carpetas con nombres “19794_7_VX”, donde X puede tomar los valores “0”, “1” o “2” dependiendo de la versión de los datos comprimidos

# 9 Resultados de Compresión

Como se mencionaba en apartados anteriores se aplican tres algoritmos de compresión sin pérdidas (RLE, LZSS y LZW) sobre el estándar 19794-7, tanto para su implementación completa como compacta. Para cada uno de estos algoritmos e implementaciones se realizan tres versiones, en función al almacenamiento de los datos MCyT en fichero tras el proceso de conversión a los estándares:

- Versión V0: los datos se almacenan conforme se indica en la norma tratada
- Versión V1: ordenando por canales, todos los datos de un mismo canal se agrupan y se almacenan, así con todos los demás, uno a continuación de otro.
- Versión V2: igual que el anterior, solo que en esta ocasión no se guardan los datos en sí, sino que se almacenan las diferencias entre muestras, es decir, la diferencia de la muestra siguiente con respecto a la actual.

Cabe mencionar que en estas tres versiones que serán comprimidas, sólo se guardan los datos, excluyendo la cabecera, ya que ésta permanece intacta y únicamente se reducen los datos.

En este apartado se analizan diversos aspectos como el número de muestras o las tasas de compresión, lo cual dará una visión de la compactación de los ficheros para los algoritmos y versiones tratadas [24].

## *9.1 Tamaño de ficheros*

Un primer estudio es el realizado para el tamaño de los ficheros, para cada uno de los casos expuestos. El proceso seguido es la realización de un programa Matlab para el cálculo del número de bytes de los ficheros por cada usuario (100 usuarios) y por cada una de las firmas genuinas (25 firmas), no incluyendo las falsificaciones, realizando la media de todas ellas. Los resultados obtenidos son los siguientes:

| TAMAÑO EN BYTES | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7F (sin comprimir) | 7V0 | 7V1 | 7V2 | 7C (sin comprimir) | 7CV0 | 7CV1 | 7CV2 |
| RLE | 4643 | 3249 | 2786 | 2937 | 2109 | 2190 | 1503 | 1613 |
| LZSS | | 3009 | 2877 | 1759 | | 1801 | 1479 | 1175 |
| LZW | | 3088 | 3153 | 1587 | | 1784 | 1430 | 990 |

**Tabla 5 - Tamaño de ficheros en bytes para las diferentes versiones y algoritmos de compresión**

En las filas se exponen los algoritmos de compresión aplicados, y en las columnas, las diferentes versiones V0, V1 y V2 tanto para el formato 19794-7 completo (7V0, 7V1, 7V2) como para el compacto (7CV0, 7CV1, 7CV2), así como los formatos completo y compacto según se definen en la norma sin comprimir. El tamaño de los ficheros, expresado en bytes, incluye la cabecera, que no se comprime al quedar inalterada pero sí que se suma al tamaño final.

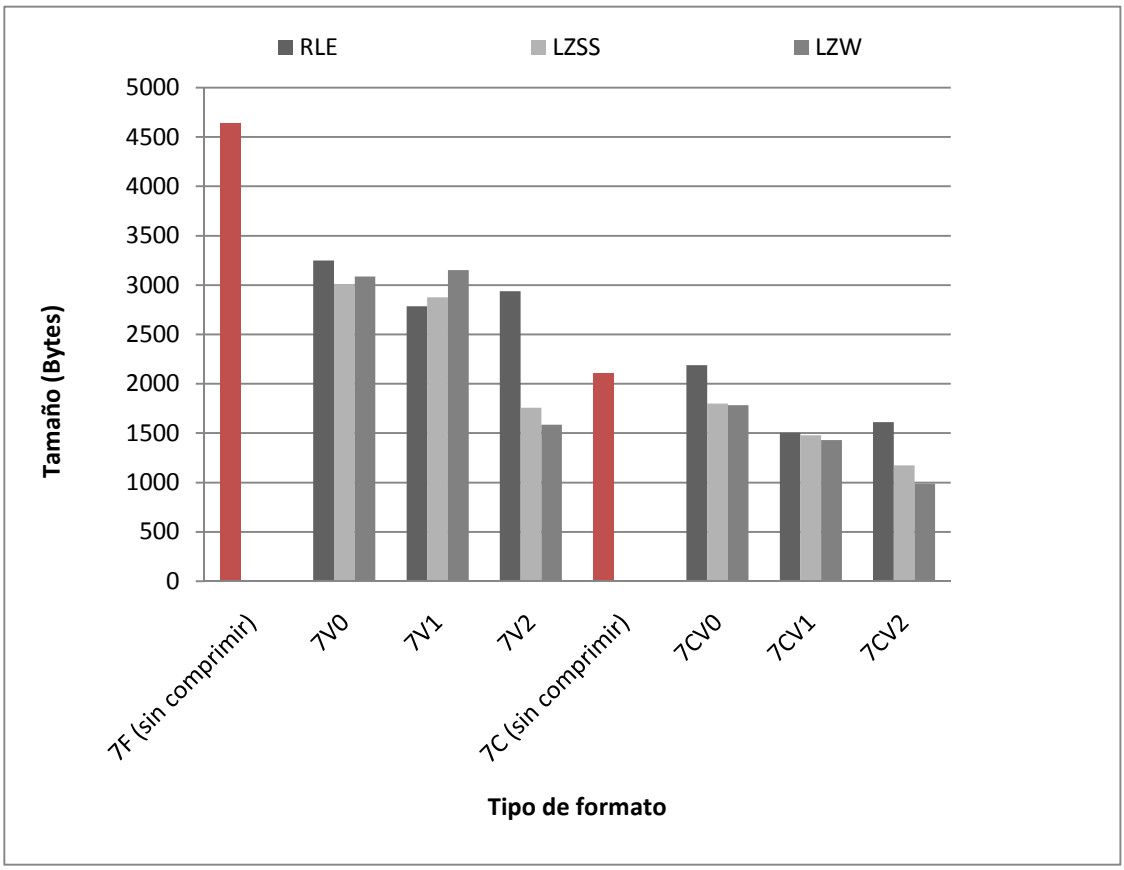



**Figura 39 - Tamaño de ficheros en bytes para las diferentes versiones y algoritmos de compresión**

En una primera vista, y como era lógico, se obtiene un menor número de bytes para las versiones de la norma 19794-7 compacto, que carece de cabecera y ya de por sí reducen los datos con respecto a la versión completa de dos bytes a un solo byte.

En un análisis más exhaustivo se puede apreciar que la versión V2 es la más comprimida, seguida por la versión V1 y por último la regida por la norma. Esto se debe a la repetitividad de los datos de un mismo canal, de manera que al agruparlos se tendrán más coincidencias consecutivas, en caso de algoritmos de diccionario también es más eficiente y se consigue una mayor tasa de compresión. Con respecto a la diferencia entre las versiones V1 y V2, no es apreciable en el algoritmo RLE debido a que la repetitividad consecutiva será la misma al tener éstos la misma localización pero

representados de forma diferente. Sin embargo, se observa que en los algoritmos basados en diccionario, los resultados obtenidos son bastante mejores para la versión V2 con respecto a V1.

Seguidamente se exponen los resultados particularizados para cada uno de los algoritmos, para extraer conclusiones más particularizadas.

### 9.1.1 Algoritmo RLE (Run Length Encoding)

En la siguiente tabla se ilustra el número de bytes para cada uno de los casos expuestos para el algoritmo RLE:

| TAMAÑO EN BYTES | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7F (sin comprimir) | 7V0 | 7V1 | 7V2 | 7C (sin comprimir) | 7CV0 | 7CV1 | 7CV2 |
| RLE | 4643 | 3249 | 2786 | 2937 | 2109 | 2190 | 1503 | 1613 |

**Tabla 6 - Tamaño de ficheros en bytes aplicando algoritmo de compresión RLE a las diferentes versiones**

Como se desprendía de los resultados globales, las diferencias para este algoritmo son principalmente entre la versión V0 con respecto a las versiones V1 y V2, debido a que el ordenamiento de los datos de las últimas es la misma, realizada por canales, mientras que para la primera es diferente a éstas y la información repetitiva está más dispersa. De lo anterior se deduce un menor tamaño en bytes de las versiones uno y dos, obteniéndose entre ellas diferencias menores provocadas por las mayores coincidencias consecutivas de las muestras almacenadas en la versión V2 con respecto a la versión V1.

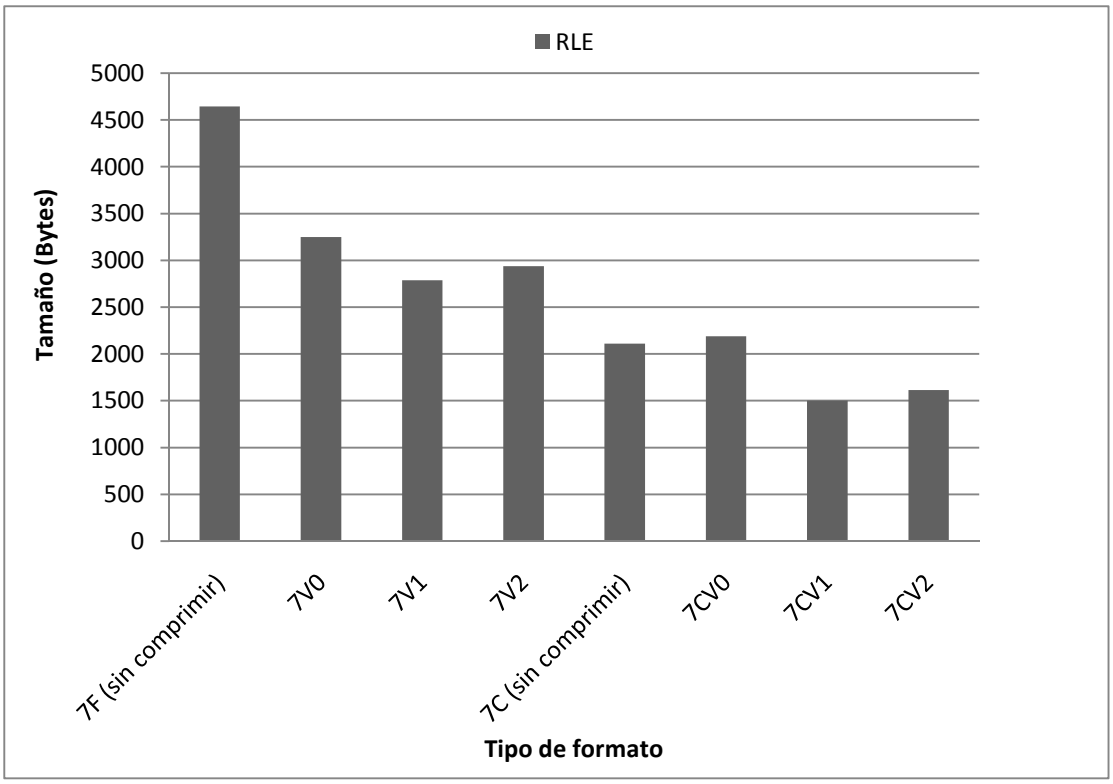



**Figura 40 - Tamaño de ficheros en bytes aplicando algoritmo de compresión RLE a las diferentes versiones**

### 9.1.2 Algoritmo LZSS (LZ77)

En la tabla mostrada a continuación se puede observar la gran diferencia en número de bytes dependiendo de las diferentes normas y versiones:

| TAMAÑO EN BYTES | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7F | 7V0 | 7V1 | 7V2 | 7C | 7CV0 | 7CV1 | 7CV2 |
| LZSS | 4643 | 3009 | 2877 | 1759 | 2109 | 1801 | 1479 | 1175 |

**Tabla 7 - Tamaño de ficheros en bytes aplicando algoritmo de compresión LZSS a las diferentes versiones**

Para el caso completo se aprecia una enorme diferencia de la versión V2 con respecto al resto, mientras que para el caso compacto las diferencias se mantienen entre los diferentes tipos y en unos niveles similares a los obtenidos con la versión V2, lo cual indica la gran reducción obtenida para ésta, ya que iguala y supera en algún caso a la compacta. Es especialmente significativo lo que ocurre para la versión 7V2, que tiene un menor número de bytes que la compacta, de modo que sin perder resolución, con la compresión se logra reducir más bytes aún. Para verlo de forma más clara se adjunta la gráfica que sigue:

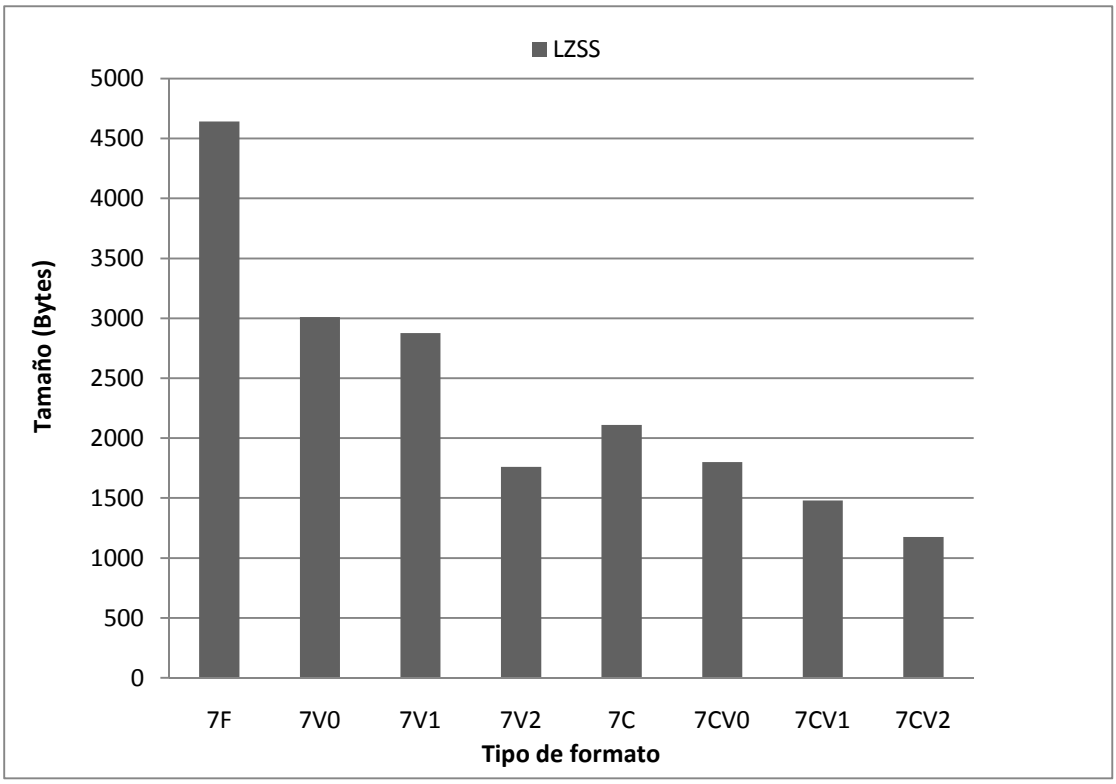



**Figura 41 - Tamaño de ficheros en bytes aplicando algoritmo de compresión LZSS a las diferentes versiones**

### 9.1.3 Algoritmo LZW (LZ78)

Los resultados en este caso se mantienen en los niveles, aproximadamente, del caso anterior, aunque sufriendo algunas variaciones:

| TAMAÑO EN BYTES | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7F (sin comprimir) | 7V0 | 7V1 | 7V2 | 7C (sin comprimir) | 7CV0 | 7CV1 | 7CV2 |
| LZW | 4643 | 3088 | 3153 | 1587 | 2109 | 1784 | 1430 | 990 |

**Tabla 8 - Tamaño de ficheros en bytes aplicando algoritmo de compresión LZW a las diferentes versiones**

Se vuelve a mostrar la gran diferencia en número de bytes de la versión V2 con respecto a las otras dos en el formato completo, lo cual implica un uso mucho más eficiente para ésta, estando en el rango aproximadamente de las versiones compactas, y ocupando la mitad que las otras versiones de su formato. Para el formato compacto se mantienen las diferencias, siendo de nuevo el caso con menor longitud la versión V2, que se destaca siempre como la de menor tamaño. En la gráfica de barras posterior se aprecia con mayor claridad:

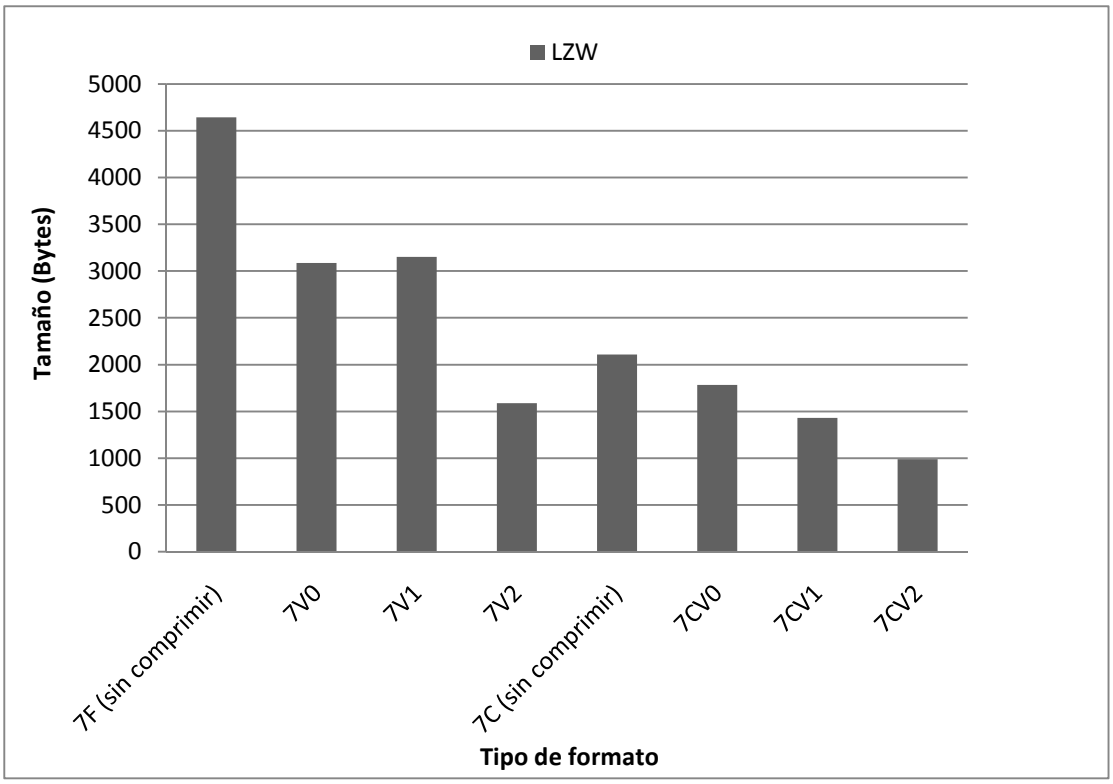



**Figura 42 - Tamaño de ficheros en bytes aplicando algoritmo de compresión LZW a las diferentes versiones**

La significativa diferencia entre el número de bytes entre el formato completo y compacto, se debe al menor número de datos a comprimir, ya que el compacto no tiene cabecera ni algunos datos del cuerpo, por lo demás los datos son los mismos, pero representados con menor número de bytes, lo cual es el mayor factor de influencia.

## *9.2 Tasa de compresión*

Otro objeto de análisis es la tasa de compresión, que se calcula como el cociente entre la diferencia entre el tamaño medio en bytes de los archivos originales y de los comprimidos, dividido por el tamaño original:

$$T = \frac{num\_medio\_bytes\_original - num\_medio\_bytes\_comprimido}{num\_medio\_bytes\_original}$$

Se consideran tres casos, el primero de ellos es la tasa de compresión del formato 19794-7 completo con respecto a las tres versiones realizadas sobre él. El segundo es igual que el anterior pero para el caso compacto. El último refleja una comparación entre el formato completo y las versiones compactas.

### 9.2.1 Formato completo

En la siguiente tabla se exponen en las columnas los diferentes algoritmos de compresión y en las filas las tasas de compresión para cada caso, del formato completo con respecto a las versiones V0, V1 y V2:

| RATIO DE COMPRESION 7F -V0, V1, V2 | | | |
|---|---|---|---|
| | 7F - V0 | 7F - V1 | 7F - V2 |
| RLE | 30% | 40% | 37% |
| LZSS | 35% | 38% | 41% |
| LZW | 33% | 32% | 66% |

**Tabla 9 - Ratio de compresión de ficheros para la versión 19794-7 completa con respecto a sus modificaciones**

Las mayores tasas de compresión se obtienen para la versión V2, como ya se deducía del estudio realizado para el número de bytes de los ficheros. La mejor de todas es para el algoritmo LZW, muy destacado con respecto al resto, obteniendo casi el doble que para los otros. En los casos restantes las tasas se mantienen más o menos con diferencias de cómo mucho un 10%. Cabe mencionar que el algoritmo que menores tasas obtiene es RLE y particularmente el menor de todos es para la versión V0, debido a su poca repetitividad por la no ordenación por canales. En la gráfica de barras siguiente se pueden apreciar estos resultados:

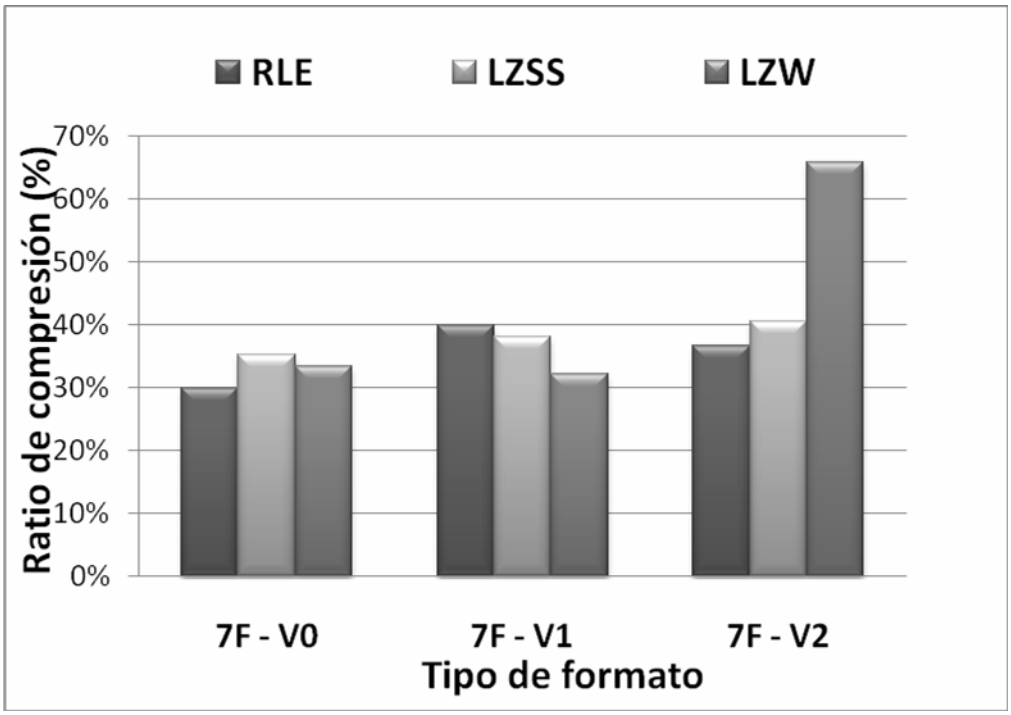



**Figura 43 - Ratio de compresión de ficheros para la versión 19794-7 completa con respecto a sus modificaciones**

### 9.2.2 Formato compacto

Como se observa se siguen manteniendo las conclusiones que se van derivando de secciones anteriores, la versión V2 es la que mejores resultados ofrece, destacándose positivamente para el caso LZW. El algoritmo RLE sigue siendo el peor de los tres, pero ahora sus tasas son aproximadamente la mitad del resto, siendo bastante peor e incluso, para la versión V0 se obtiene un resultado negativo. Para los otros dos algoritmos se obtienen tasas dentro de un rango del 10% excepto la versión comentada al inicio.

| RATIO DE COMPRESION 7C - CV0, CV1, CV2 | | | |
|---|---|---|---|
| | 7C - CV0 | 7C - CV1 | 7C - CV2 |
| RLE | -4% | 29% | 24% |
| LZSS | 15% | 30% | 44% |
| LZW | 15% | 32% | 53% |

**Tabla 10 - Ratio de compresión de ficheros para la versión 19794-7 compacta con respecto a sus modificaciones**

En este apartado se produce un caso particular, para el algoritmo RLE y la versión V0 no se logra comprimir nada, al contrario, el fichero comprimido es un 4% mayor que el original, de forma que el uso del algoritmo no es eficiente. La gráfica pone de manifiesto los resultados expuestos anteriormente:

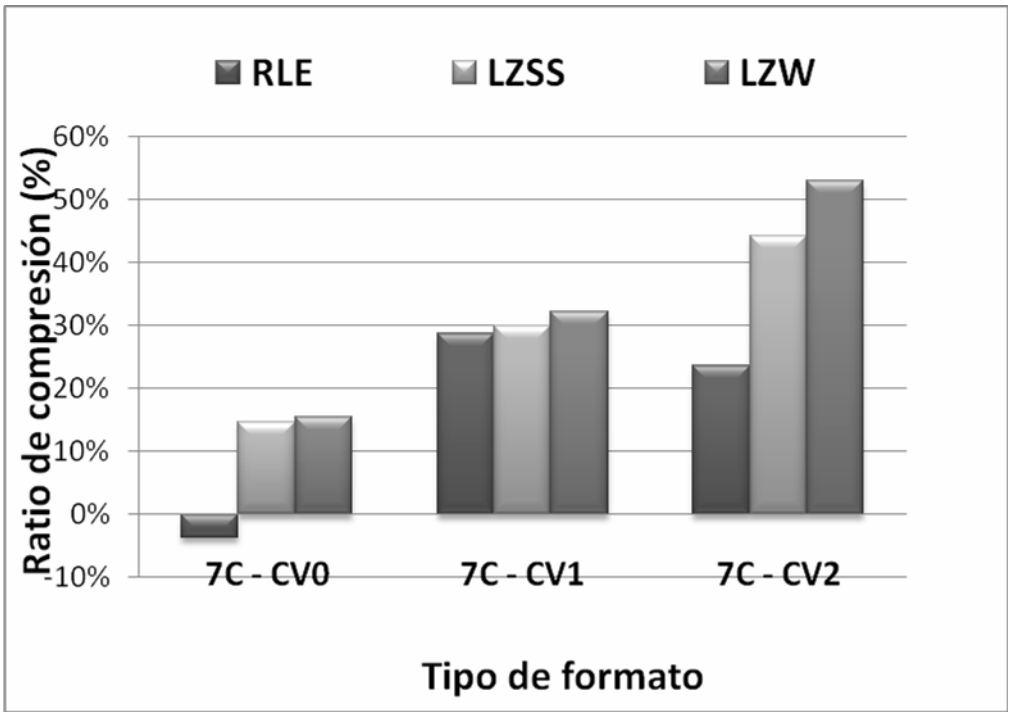



**Figura 44 - Ratio de compresión de ficheros para la versión 19794-7 compacta con respecto a sus modificaciones**

### 9.2.3 Formato completo con respecto a compacto

Por último se calculan las tasas de compresión del formato completo con respecto a las versiones compactas y así se da una visión del ratio global desde el formato inicial hasta el compacto, que se compone a partir de la compresión del primero. Como es de esperar, las tasas de compresión son mucho mayores que en los casos anteriores, por el hecho de que en los casos precedentes se analizaban el formato con sus respectivas versiones, y en este caso el formato con mayor longitud de datos, con las versiones con menor longitud de bytes, sin incluir cabecera y algunos datos.

| RATIO DE COMPRESION 7F - CV0, CV1, CV2 | | | |
|---|---|---|---|
| | 7F - CV0 | 7F - CV1 | 7F - CV2 |
| RLE | 53% | 68% | 65% |
| LZSS | 61% | 68% | 75% |
| LZW | 62% | 69% | 79% |

**Tabla 11 - Ratio de compresión de ficheros para la versión 19794-7 completa con respecto a las modificaciones de 19794-7 compacto**

Las tasas obtenidas, son, en muchos casos, más del doble de las obtenidas en apartados anteriores, destacando como la mejor la versión V2 para el algoritmo LZW con casi un 80%. Para las versiones V1 y V2 se mantienen aproximadamente las tasas en un rango del 10%, bajando ligeramente para la versión V0 y cayendo de manera significativa para el algoritmo RLE de ésta. En la gráfica posterior aparece la suave pendiente de ascenso de las tasas de compresión de una versión y algoritmo a otro, es decir, las tasas de compresión crecen, siendo menores para RLE, subiendo para LZSS y las mayores se obtienen para LZW.

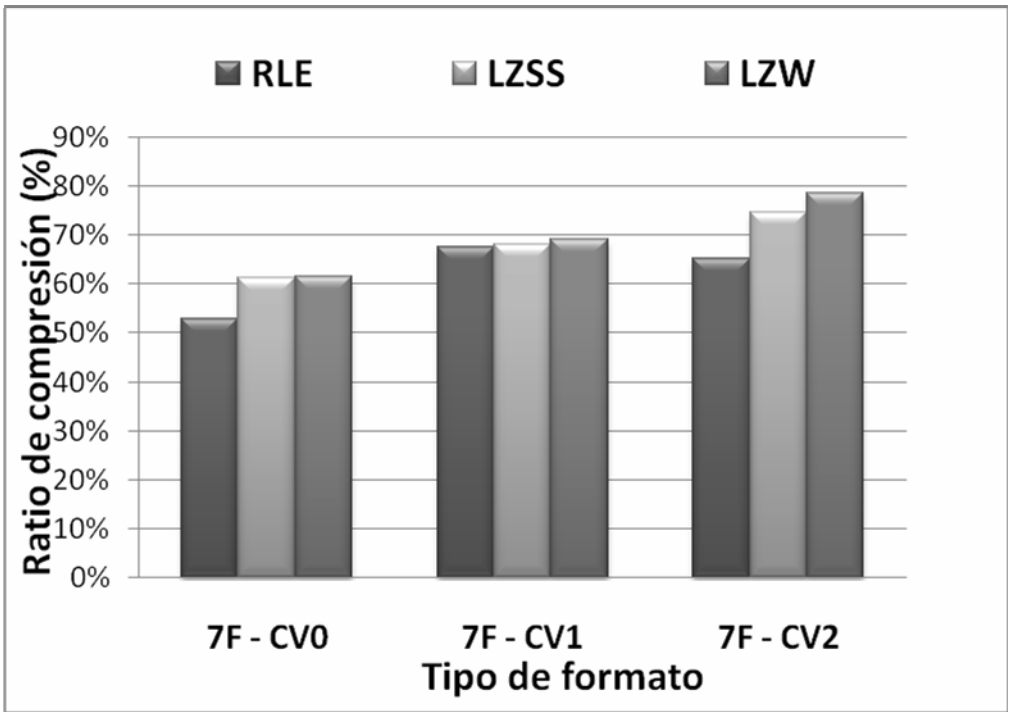



**Figura 45 - Ratio de compresión de ficheros para la versión 19794-7 completa con respecto a las modificaciones de 19794-7 compacto**

# 10 Conclusiones y Trabajos Futuros

Del trabajo y estudios desarrollados se pueden extraer las siguientes conclusiones:

- Con respecto a 19794-7 se observa una gran reducción de datos de la versión compacta con respecto a la completa, aunque como punto negativo de ésta es la pérdida de resolución de pasar de 2 bytes a 1 byte.
- Con respecto al formato 11 se observa que su desarrollo es más complejo debido a la creación de los strokes de presión y posición. Se alcanza cierto grado de compresión, pero no compensa con la pérdida de información ya que se pierden los datos de inclinación y azimuth, movimiento del bolígrafo en el aire, además de que ahora sólo se tienen los puntos singulares.
- Se encuentran diferentes imprecisiones en el estándar y aspectos que pueden ejecutarse de diferente forma. El comité de desarrollo del estándar decide cambiarle para alcanzar una mayor compresión de datos.
- Como se deriva del análisis gráfico de los apartados de compresión de datos, para datos repetitivos consecutivos el algoritmo RLE consigue buenos resultados, pero como norma general, los algoritmos basados en diccionario mejoran la compactación de datos, reduciendo en mayor manera la información a almacenar
- El algoritmo LZW, evolución de LZ78, como norma general y excepto en algunos casos, es el algoritmo que mayor compactación de datos y por tanto mejores tasas de compresión consigue para los diferentes formatos de datos, pero como nota negativa cabe mencionar que es la implementación más compleja de las tres empleadas
- La implementación de algoritmos basados en diccionario es más compleja que RLE, además de necesitar mayor tiempo de cómputo para la creación y búsquedas en el diccionario así como mayor espacio de almacenamiento.

Como trabajos futuros y posibles líneas de investigación se comentan:

- Desarrollo de un nuevo estándar que evite los problemas surgidos en torno a 19794-11, pero que siga en la misma línea que éste, ya que las intenciones pretendidas eran de gran interés para reducir el tamaño de los datos, a pesar de su complejidad de implementación con respecto a 19794-7

- Además de la compresión de los datos mediante el estándar, sería recomendable un estudio de un algoritmo de compresión adecuado para compactar aún más los datos resultantes, que no dificulte mucho el proceso de obtención de firmas y que aporte un alto grado de reducción de almacenamiento de datos.

- Prueba con más algoritmos de compresión sin pérdidas, como por ejemplo PNG, GIF, Huffman y con algoritmos de compresión con pérdidas de series temporales, audio, etc… algunos de los cuales pueden ser MP3, JPEG, WMA con lo que se lograría una mayor compresión de los datos aunque implicarían la pérdida de información.

- Construcción de nuevas bases de datos reales con información de diferentes personas de sexos y edades diferentes, que permitan un juego de pruebas de los estándares mayor. Sobre todo más bases de datos con firmas de de distintos estilos: occidentales, anglosajonas, españolas, orientales (chinas, japonesas, koreanas), islámicas…

# 11 Referencias


[1] Biometría, 2009. http://es.wikipedia.org/wiki/Biometr%C3%ADa

[2] Biometrics: biometría y seguridad personal, 2009. http://biometrics-on.com/es/

[3] Tutorial sobre biometría, 2009. http://tutorial-biometria.galeon.com/

[4] http://www.alambre.info/2003/12/08/origenes-de-la-firma-autografa/

[5] http://www.iec.csic.es/criptonomicon/articulos/expertos70.html

[6] Ortega García, Javier. Firma manuscrita On-line (http://arantxa.ii.uam.es/~jortega/FirmaManuscritaOnLine_ASAL.pdf). Universidad Autónoma de Madrid. 2009

[7] La estandarización en el campo de la identificación biométrica, 2009. http://www.shsconsultores.es/jornadas/0202_-_la_estandarizacion_en_el_campo_de_la_identificacion_biometrica.pdf

[8] Information Technology—Biometric Data Interchange Formats—Part 7: Signature/Sign Time Series Data, ISO Standard ISO/IEC FCD 19794-7, 2006.

[9] Information Technology—Biometric Data Interchange Formats—Part 11: Signature/Sign Processed Dynamic Data, ISO Standard ISO/IEC CD 19794-11, 2009.

[10] J. Ortega-Garcia, J. Fierrez-Aguilar, D. Simon, J. Gonzalez, M. Faundez-Zanuy, V. Espinosa, A. Satue, I. Hernaez, J.-J. Igarza, C. Vivaracho, D. Escudero and Q.-I. Moro, "*MCYT baseline corpus: a bimodal biometric database"*, IEEE Proc.-Vis. Image Signal Process., Vol. 150, No. 6, pp 395-401. December 2003.

[11] http://es.wikipedia.org/wiki/Compresi%C3%B3n_de_datos

[12]http://gpsis.utp.edu.co/omartrejos/descargas/Proy%20Tecnicas%20de%20Compresion.pdf

[13] http://ccc.inaoep.mx/%7Ecferegrino/cursos/comprcrip/Compre1.pdf

[14]http://es.wikipedia.org/wiki/Algoritmo_de_compresi%C3%B3n_con_p%C3%A9rdida

[15]http://es.wikipedia.org/wiki/Algoritmo_de_compresi%C3%B3n_sin_p%C3%A9rdida

[16] http://es.kioskea.net/contents/video/compimg.php3

[17] http://es.wikipedia.org/wiki/LZSS

[18] http://ditec.um.es/cgi-bin/dl/compress.pdf

[19] http://michael.dipperstein.com/

[20] http://www.cygwin.com/setup.exe

[21] http://michael.dipperstein.com/rle/index.html

[22] http://michael.dipperstein.com/lzss/

[23] http://michael.dipperstein.com/lzw/

[24] Miguel-Hurtado, O.; Mengibar-Pozo, L.; Tomeo-Reyes, I.; Liu-Jimenez, J.; *"Analysis en compact data formats for the performance of Handwritten Signature Biometrics"*. ICCST 2009.