Escuela Politécnica Superior

Construcción de un corpus de referencia para investigación en reconocimiento automático de partituras musicales

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Autor:
María Alfaro Contreras

Tutor/es:
José Manuel Iñesta Quereda – Jorge Calvo Zaragoza

Junio 2018

Universitat d'Alacant
Universidad de Alicante

Construcción de un corpus de referencia para investigación en reconocimiento automático de partituras musicales

Autor
María Alfaro Contreras

Directores
José Manuel Iñesta Quereda
Departamento de Lenguajes y Sistemas Informáticos
Jorge Calvo Zaragoza
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Escuela
Politécnica
Superior

ALICANTE, 21 de junio de 2018

Resumen

La música es una de las mayores manifestaciones culturales del ser humano. Es por ello que, a lo largo de toda la historia, siempre ha existido la necesidad de conservar los documentos musicales de la manera más minuciosa posible con el fin de favorecer la preservación del patrimonio musical, así como su acceso, estudio y distribución. Para amenizar este trabajo, se ha puesto un gran esfuerzo en desarrollar diferentes herramientas que permitan generar de manera automática la transcripción de partituras, un trabajo tedioso incluso para los más profesionales.

El Reconocimiento Óptico de Música es la rama de la inteligencia artificial que desarrolla sistemas que sean capaces de extraer el contenido musical de una imagen de una partitura musical y transcribirlo a un formato que permita procesarlo fácilmente por un ordenador. La tendencia para el desarrollo de estos sistemas es usar técnicas de aprendizaje automático. Estas técnicas son capaces de inferir la transcripción a partir de ejemplos correctos de la tarea, es decir, conjunto de pares (imagen, transcripción). Dada la complejidad de la música, para que estas técnicas arrojen resultados satisfactorios es necesario utilizar un conjunto muy grande. Es por ello que una buena medida es desarrollar sistemas que sean capaces de generar estos pares automáticamente, de forma que se pueda obtener un conjunto potencialmente ilimitado de ejemplos. Ahí es donde entra en juego el presente proyecto.

Por todo lo expuesto anteriormente, este trabajo de fin de grado construye un corpus de referencia para investigación en reconocimiento automático de partituras musicales con el fin de proporcionar a los sistemas de reconocimiento automático una base de datos de ejemplos ilimitados. Para conseguirlo, se ha desarrollado un sistema de generación automática de datos etiquetados, el cual emite dos salidas: por un lado, la transcripción esperada de la partitura generada; por otro lado, la partitura en formato PDF. Con ambas salidas, se estarían obteniendo los pares necesarios para el algoritmo de aprendizaje automático. El sistema etiqueta música monofónica pero se ha desarrollado de manera que pueda editarse para en un futuro ampliar el etiquetado de datos a música polifónica, con la intención de que pueda ser utilizado por otros investigadores como base para la creación de sus propias bases de datos.

Abstract

Music is one of the greatest cultural manifestations of the human being. That is why, throughout history, there has always been the need to preserve musical documents in the most thorough way possible in order to support the preservation of musical heritage, as well as its access, study and distribution. In order to help develop this work, a great effort has been put into developing different tools that allow the automatic transcription of music scores, a tedious task even for the most professional ones.

Optical Music Recognition is the branch of artificial intelligence that develops systems that are able to extract the musical content of an image from a musical score and transcribe it into a format that allows it to be easily processed by a computer. The tendency for the development of these systems is to use automatic learning techniques. These techniques are able to infer the transcription from correct examples of the task, that is, set of pairs (image, transcription). Given the complexity of music, for these techniques to produce satisfactory results it is necessary to use a very large set. That is why a good measure is to develop systems that are capable of generating these pairs automatically, so that a potentially unlimited set of examples can be obtained. That is where the present project plays its part.

For all the above, this project builds a reference corpus for research in automatic recognition of musical scores in order to provide automatic recognition systems with a database of unlimited examples. To achieve this, a system of automatic generation of tagged data has been developed. This system has two outputs: on the one hand, the expected transcription of the generated score; on the other hand, the score in PDF format. With both outputs, the necessary pairs for the machine learning algorithm would be obtained. The developed system only tags monophonic music, but it has been developed so that it can be edited in the future to expand the labeling of data to polyphonic music, with the intention that it can be used by other researchers as a basis for the creation of their own databases.

Agradecimientos

Con este trabajo finaliza mi etapa como estudiante del Grado en Ingeniería en Sonido e Imagen en Telecomunicación de la Universidad de Alicante y es por ello, que quiero agradecer en primer lugar a mi tutor, José Manuel Iñesta. Muchísimas gracias por toda la ayuda prestada, desde el primer momento hasta el último. Gracias por guiarme a lo largo de este proyecto, teniendo siempre abiertas las puertas de tu despacho para ponerle solución a cualquier problema encontrado. Este trabajo no habría visto la luz sin la ayuda de Jorge Calvo, quien propuso el tema y ha estado ejerciendo como mi segundo tutor. Gracias por tu preocupación y ayuda, de valores incalculables. De todo corazón, mis mejores gracias a ambos por acogerme desde el primer momento y hacerme sentir tan cómoda no solo a lo largo del desarrollo de este proyecto, sino desde el primer momento que comencé mi aventura con vosotros con las prácticas en empresa.

A mis padres tengo que agradecerles todo en esta vida. Su apoyo incondicional y confianza a ciegas en mí han sido mi mejor combustible. Gracias por, simplemente, dejarme ser, por dejarme volar con los pies en la tierra. A mi hermano, por ser el mejor compañero de aventuras. Gracias por amar y disfrutar de la música de la manera en que lo haces. Ahora te toca a ti iniciar esta aventura universitaria y estoy segura de que te va a ir genial. Os quiero muchísimo. También quiero dar las gracias al resto de mi familia por estar siempre ahí, alegrándose por mis éxitos como si fueran suyos.

A mis amigos, por darme la dosis necesaria de risas en cada momento. Pero sobretodo a Ana, por ser mi amiga y mi compañera en esta aventura que ha sido el grado. Eres el mayor regalo que me han hecho estos cuatro años y volvería a vivirlos de nuevo si fuera contigo.

A todos, espero que disfrutéis la lectura tanto como yo he disfrutado redactándola.

María Alfaro Contreras

San Vicente del Raspeig, Junio de 2018

Después del silencio,
lo que más se acerca a
expresar lo inexpresable
es la ***música****.*

Aldous Huxley.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

La música es el lenguaje universal. Un arte que lleva traspasando fronteras desde sus inicios, siendo una de las principales manifestaciones culturales de la humanidad. Es por ello que, a lo largo de toda la historia, siempre ha existido la necesidad de conservar los documentos musicales de la manera más minuciosa posible en catedrales, bibliotecas o archivos históricos. No obstante, no siempre ha sido posible ese pulcro cuidado, quedando algunos de ellos dañados, limitando así su acceso ya que un uso continuo de esos documentos podría terminar por dañarlos de una manera irrevocable. Esto implica que una importante parte de este patrimonio permanezca alejada del estudio musicológico.

Para favorecer la preservación de la música, así como su acceso, estudio y distribución, se ha puesto un gran empeño en la transcripción de partituras a formato digital en los últimos años. Para ello se han desarrollado diferentes herramientas. Por ejemplo, las aplicaciones de edición de partituras están cada vez más extendidas. Mediante simples acciones con el ratón o el teclado, estas aplicaciones son capaces de crear partituras en formato digital. Otra herramienta usada para la transcripción de partituras es el uso de instrumentos digitales (por ejemplo, un piano MIDI). Estos pueden ser conectados a un ordenador, de manera que la información musical es transferida automáticamente a través de su interpretación. Desafortunadamente, este proceso no permite captar siempre todos los matices que se pueden llegar a encontrar en una partitura.

Por otra parte, la digitalización masiva de documentos musicales ha hecho que los algoritmos de Extracción y Recuperación de Información Musical cobren una gran importancia a la hora de amenizar el largo y tedioso proceso de transcripción de partituras.

La percepción de notación musical por parte de los ordenadores forma un campo de investigación en constante crecimiento llamado Reconocimiento Óptico de Música (*Optical Music Recognition*, OMR). El objetivo principal de los sistemas de OMR es decodificar e interpretar automáticamente los símbolos de notación musical a partir de imágenes escaneadas. Los resultados son representados en un formato digital adecuado, como MusicXML, MIDI o MEI, para almacenar la información semántica.

La investigación de OMR comenzó en 1966, cuando Pruslin intentó por primera vez el reconocimiento automático de partituras [Novotný and Pokorný, 2015]. Su sistema fue capaz de reconocer cabezas de notas y acordes. En 1970, Prerau introdujo el concepto de

segmentación de imágenes para detectar elementos primitivos de notación musical. Con la disponibilidad de escáneres ópticos de bajo costo, la investigación de OMR se extendió a finales de los años ochenta. Una contribución interesante fue WABOT-2, un robot antropomórfico, desarrollado en 1984, capaz de leer una partitura e incluso tocar una melodía de mediana dificultad con un sintetizador. Los primeros productos comerciales de OMR y los primeros intentos de manejar partituras manuscritas aparecieron a principios de la década de 1990.

A pesar de que los sistemas de OMR se han investigado minuciosamente durante las últimas décadas e incluso existen varias herramientas comerciales, los resultados prácticos aún están lejos de ser ideales. Las tecnologías actuales no permiten asegurar una transcripción libre de errores y puede que nunca lo hagan.

1.1. Contextualización

La manera de trabajar de los sistemas de OMR se divide, generalmente, en cuatro etapas [Novotný and Pokorný, 2015]:

1. **Preprocesamiento de imagen**.
2. **Segmentación**.
3. **Reconocimiento de objetos**.
4. **Reconstrucción semántica**.

La terminología usada para describir estas cuatro fases no es siempre la misma: la fase de segmentación también es conocida como *detección primaria o localización de objetos musicales* mientras que la fase de reconocimiento de objetos es llamada, en ocasiones, *clasificación de características musicales* [Bainbridge, 1997]. Además, para cada una de las cuatro fases, existe una amplia variedad de métodos con los que desarrollarlas.

1.1.1. Preprocesamiento de imagen

El objetivo principal de esta fase es ajustar la imagen escaneada con el fin de lograr un proceso de reconocimiento robusto y eficiente. Técnicas como realce, desenfoque, operaciones morfológicas, eliminación de ruido, *deskewing* y binarización son algunas de las más usadas en el preprocesamiento de imágenes. A continuación, se introduce únicamente la técnica de binarización pues es considerada crucial para la mayoría de los sistemas de OMR.

Binarización

El proceso de binarización es el primer proceso presente en la mayoría de los sistemas de OMR: la imagen escaneada de entrada es convertida en una imagen binaria, es decir, la imagen de entrada es analizada con el fin de decidir qué es útil (los símbolos musicales) y qué no (el fondo, el ruido). Esto está motivado por el hecho de que las partituras musicales tienen una naturaleza intrínsica binaria (los colores no guardan ninguna información musical en la notación musical).

Se trata de un proceso, generalmente automático, que facilita el trabajo de las tareas posteriores al reducir la cantidad de información que necesitan procesar. La dificultad de diseñar algoritmos de detección de pentagramas o reconocimiento de símbolos es menor si las imágenes son en blanco y negro que en color.

Por lo general, hay dos planteamientos del proceso de binarización. El primero engloba a los métodos de umbral global, los cuales, como su propio nombre indica, aplican un único umbral a toda la imagen. El método de Otsu [Otsu, 1979] a menudo se considera el mejor y más rápido [Sezgin and Sankur, 2004]. Este tipo de métodos funciona bien siempre y cuando se extraen objetos de fondos uniformes, sino tiende a fallar. Aún así, es usado en numerosos artículos de investigación de sistemas de OMR gracias a su simplicidad y a su eficiencia. El segundo planteamiento engloba a las técnicas adaptativas de binarización, las cuales determinan el umbral a usar para cada píxel usando información de su vecino. Estos métodos son capaces de separar la información necesaria de fondos no uniformes pero conllevan un mayor tiempo de procesamiento. Uno de los métodos más populares es el de Niblack [Niblack, 1985], el cual calcula un umbral para un píxel en función de la media y la desviación estándar del entorno del píxel.

1.1.2. Segmentación

En esta etapa, se analizan las partituras musicales con el fin de descomponerlas en elementos más primitivos. La primera decisión a tomar es el grado de segmentación al que se va a llegar, estableciendo la cantidad de notación musical a procesar. Este es un paso importante antes de cualquier reconocimiento de forma. Las líneas de pentagrama son una característica fiable de la notación musical utilizada para estimar dos valores de referencia importantes: el grosor de la línea de pentagrama y la altura del espacio de pentagrama, ambos utilizados para deducir el tamaño de otros símbolos musicales. La forma más común de su aproximación se basa en la codificación *run-length* (*Run-Length Enconding*), que es una forma muy simple de compresión de datos. Por ejemplo, supongamos la secuencia binaria {1 1 1 0 0 1 1 1 1 0 0 0 0}. Puede ser representada en RLE como {3 2 4 4} (suponiendo que la secuencia original comienza en 1, de lo contrario, el primer número en la secuencia codificada sería 0). Las partituras de música binarizada

pueden ser codificadas columna por columna con RLE, para así luego estimar fácilmente las longitudes relativas: la secuencia más "negra" (0) aproxima el grosor de la línea de pentagrama mientras que la secuencia más "blanca" (1) estima la altura del espacio de pentagrama. Sin embargo, existen aproximaciones más robustas [Cardoso and Rebelo, 2010].

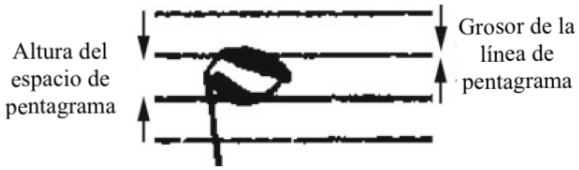


Figura 1.1: Características de las líneas de pentagrama. De [Cardoso and Rebelo, 2010].

Líneas de pentagrama

La detección de líneas de pentagrama es un proceso fundamental en OMR porque el pentagrama crea un sistema de coordenadas de dos dimensiones esencial para entender la notación musical común. Desafortunadamente, las líneas de pentagrama no son siempre perfectamente horizontales o con un grosor uniforme en imágenes escaneadas (incluso en partituras impresas). La detección precisa de estas es, a día de hoy, un problema difícil todavía no resuelto.

Los algoritmos más simples usan proyecciones horizontales [Fujinaga, 1988, Fujinaga, 2004]. Una proyección horizontal mapea una imagen binaria en un histograma acumulando el número de píxeles negros en cada fila. Si las líneas son rectas y horizontales, el pentagrama se puede detectar como cinco picos distintos consecuentes (máximos locales) en el histograma.

La figura 1.2 muestra un fragmento de una partitura musical y su proyección horizontal. En la práctica, se calculan varias proyecciones horizontales con ángulos de rotación ligeramente diferentes para tratar aquellas líneas de pentagrama que no son completamente horizontales. La proyección con los máximos locales más altos es la elegida.

Otras estrategias usan líneas de escaneo vertical [Carter, 1992] o la transformada de Hough o la agrupación de columnas verticales [Rebelo et al., 2012]. A pesar de las numerosas técnicas de detección de líneas de pentagrama, todas cuentan con ciertas limitaciones.

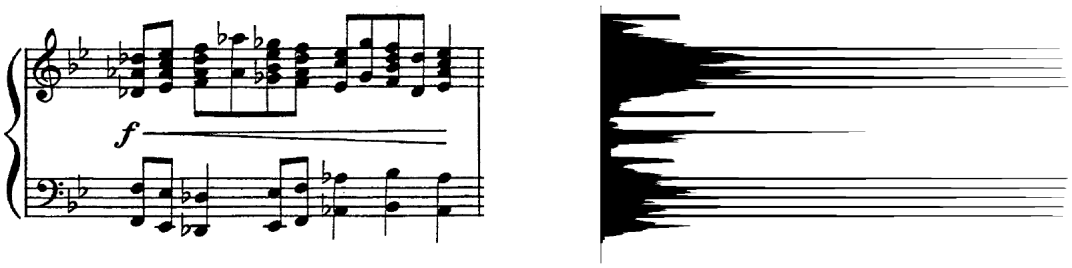


Figura 1.2: Proyección horizontal de un fragmento musical.

Segmentación de símbolos

Una vez que se han identificado las líneas de pentagrama, los elementos musicales primitivos deben localizarse y aislarse. Esto se puede lograr de dos maneras: eliminando las líneas de pentagrama o ignorándolas. Aunque la mayoría de los investigadores prefieren eliminarlas con el fin de aislar los símbolos musicales como componentes conectados, hay algunos autores que sugieren lo contrario (por ejemplo, [Bellini et al., 2001] o [Göcke, 2003]).

El algoritmo de eliminación de líneas de pentagrama más simple elimina la línea por partes, siguiéndola y reemplazando los píxeles negros de la línea con píxeles blancos a menos que haya evidencia de un objeto a cada lado de la línea [Bainbridge and Bell, 2001]. Este proceso debe ser muy minucioso para así no romper ningún símbolo musical. No obstante, los algoritmos a menudo causan fragmentación, especialmente a aquellos objetos que tocan tangencialmente las líneas del pentagrama. La partitura es entonces dividida en regiones de interés con el fin de localizar y aislar los elementos musicales primitivos. El mejor enfoque es la descomposición jerárquica [Rebelo et al., 2012]. Para comenzar, se analiza la partitura musical y se descompone en pentagramas. Luego, se extraen los símbolos musicales primitivos (cabezas de notas, plicas, corchetes, silencios, etc.). Este procedimiento varía según el sistema, por ejemplo, algunos enfoques consideran que las cabezas de notas, las plicas y los corchetes son objetos separados, mientras que otros consideran estas primitivas como un objeto completo que representa una sola nota. Para un análisis más detallado acerca de estos procedimientos, se recomienda leer [Rebelo et al., 2010].

1.1.3. Reconocimiento de objetos

Los símbolos segmentados son procesados con el fin de que el clasificador los reconozca, asignándoles una etiqueta correspondiente a un grupo previamente definido. Desafortunadamente, las formas musicales son inherentemente complejas; a menudo formadas por la superposición de varios componentes. Además, la eliminación del pentagrama puede

romper algunos elementos (a veces ya están fragmentados debido a la calidad de la propia partitura musical). Por lo tanto, la fase de reconocimiento de objetos es muy delicada y generalmente se combina con el paso de segmentación [Rebelo et al., 2012].

Los objetos son clasificados de acuerdo a sus características distintivas. Algunos autores sugieren la clasificación utilizando perfiles de proyección [Fujinaga, 1988], otros aplican la coincidencia de plantillas para reconocer símbolos [Göcke, 2003] o proponen un proceso de reconocimiento totalmente impulsado por gramáticas que formalizan el conocimiento de la música [Coüasnon and Camillerapp, 1994]. Los métodos de clasificación estadística que utilizan máquinas de vectores de soporte (*Support Vector Machines*, SVMs), redes neuronales (*Neural Networks*, NNs), el método de los k vecinos más próximos (*k-nearest neighbors*, knn) y los clasificadores de modelos ocultos de Markov (*Hidden Markov Model*, HMM) fueron investigados por Rebelo et al. [Rebelo et al., 2010].

Los símbolos musicales manuscritos a veces se segmentan y reconocen utilizando la morfología matemática, aplicando una técnica de *esqueletización* y un algoritmo de detección de bordes [Ng et al., 1999]. A pesar de la cantidad de técnicas de reconocimiento disponibles, la investigación sobre la segmentación y el reconocimiento de símbolos sigue siendo importante y necesaria, ya que todos los sistemas de OMR dependen de ella.

1.1.4. Reconstrucción semántica

La tarea inevitable de todos los sistemas de OMR es reconstruir la semántica musical de los elementos primitivos previamente reconocidos y almacenar la información en una estructura de datos adecuada. Esto necesariamente requiere una interpretación de las relaciones espaciales entre los objetos encontrados en la imagen. Las relaciones en la notación musical común son esencialmente bidimensionales y la información posicional es muy crítica. Por ejemplo, un punto puede cambiar la duración de una nota si está colocado a la derecha de la cabeza de la nota o puede alterar su articulación si se coloca encima de la nota.

Estas reglas musicalmente sintácticas se pueden formalizar usando las gramáticas [Bainbridge, 1997], [Coüasnon and Camillerapp, 1994] o [Fujinaga, 1988]. Las reglas de gramática especifican eventos de notación musical semánticamente válidos y una forma de cómo se deben segmentar los elementos gráficos primitivos. Las técnicas alternativas construyen la reconstrucción semántica basándose en un conjunto de reglas y heurísticas (por ejemplo, [Ng et al., 1999]).

El último y fundamental aspecto de los sistemas OMR es la transformación de las partituras semánticamente reconocidas en un formato de codificación que es capaz de modelar y almacenar información musical. Existen numerosos formatos digitales pero ninguno de ellos ha sido aceptado como estándar. Los formatos más conocidos son: MI-

DI (*Musical Instrument Digital Interface*), NIFF (*Notation Information File Format*), SMDL (*Standard Music Description Language*) y MusicXML. MIDI se usa principalmente como un formato de intercambio entre instrumentos digitales y ordenadores. Aunque su capacidad de modelar partituras musicales es muy limitada (por ejemplo, las relaciones entre símbolos no se pueden representar), la mayoría de los editores de música pueden operar con archivos MIDI. NIFF fue desarrollado en 1994 para intercambiar datos entre diferentes programas de notación musical y es capaz de describir aspectos visuales y lógicos de la música, sin embargo, hoy en día se considera obsoleto. SMDL separa estrictamente los sitios visuales y lógicos y es más bien un esquema formal estandarizado que un formato de archivo práctico. MusicXML está diseñado especialmente para compartir y archivar partituras musicales. Cubre la estructura lógica y también aspectos gráficos de las partituras musicales. Cada vez es más popular y su objetivo es convertirse en el formato abierto estándar para el intercambio de partituras digitales. Para una revisión y comparación más detallada de los distintos formatos de archivos de notación musical se recomienda consultar [Bellini and Nesi, 2003].

En resumen, el objetivo principal de esta etapa es la construcción de un modelo de notación musical que pueda ser usado como una descripción simbólica de la partitura musical.

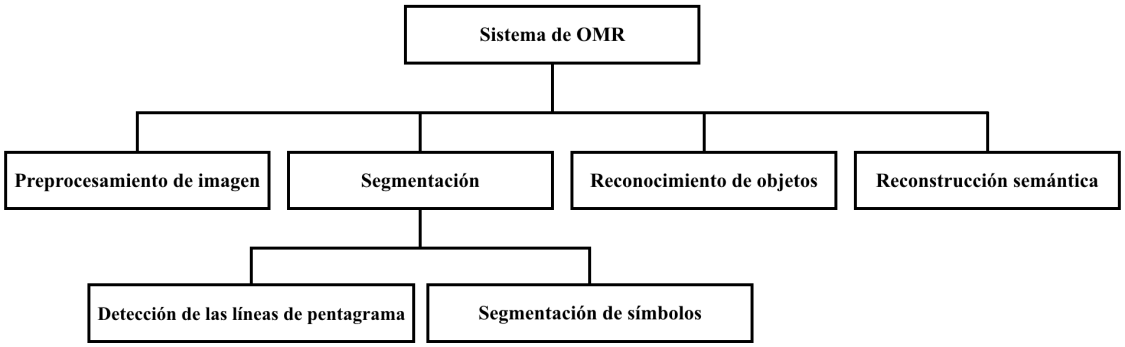


Figura 1.3: Estructura de un sistema de OMR.

1.2. Problemas presentes

La tendencia para el desarrollo de sistemas de OMR es usar técnicas de aprendizaje automático. Estas técnicas son capaces de inferir la transcripción a partir de ejemplos correctos de la tarea, es decir, conjunto de pares (imagen, transcripción). Dada la complejidad de la música, para que estas técnicas arrojen resultados satisfactorios es necesario utilizar un conjunto muy grande. Aquí es donde aparece uno de los principales problemas de la investigación de OMR: la falta de una base de datos a usar como punto de referencia. Tal conjunto de datos contendría una gran cantidad de partituras musicales

de diferentes tipos (escaneos limpios, fotocopias, manuscritos degradados, etc.) junto con su correspondiente representación en un formato de notación uniforme.

La creación de dicho corpus es una tarea extremadamente lenta debido a que la mayoría de los ejemplos disponibles son manuscritos. Es por ello que una buena solución es desarrollar sistemas que sean capaces de generar estos pares automáticamente, de forma que se pueda obtener un conjunto potencialmente ilimitado de ejemplos. Ahí entra en juego el presente proyecto. El propósito de este trabajo de fin de grado es la creación de un corpus de referencia para investigación en reconocimiento automático de partituras musicales. El objetivo principal es desarrollar un generador automático de partituras musicales que emita dos salidas: por un lado, la transcripción esperada de la partitura generada; por otro lado, la partitura en formato PDF. Con ambas salidas, se estarían obteniendo los pares necesarios para el algoritmo de aprendizaje automático.

2 Conceptos previos

En este capítulo, se explicarán una serie de conceptos acerca de la notación musical necesarios para poder entender el desarrollo, el funcionamiento y la utilidad del generador automático de partituras musicales propuesto. No se pretende realizar un manual de referencia sobre notación musical, pues algunos de los conceptos explicados no serán necesarios para poder trabajar con la aplicación en modo usuario, pero sí para entender parte de la implementación. La intención es explicar estos conceptos desde un punto en el que el lector pueda entenderlos e identificarlos sin la necesidad de tener un conocimiento musical elevado. Además, se hará una explicación detallada acerca de dos de las herramientas principales usadas en el desarrollo de este trabajo: el lenguaje musical *kern* de Humdrum y la librería musical Verovio.

2.1. Notación musical

Desde la Antigüedad, la humanidad ha hecho intentos por resguardar las manifestaciones musicales de manera gráfica. Los neumas, primitivos grafismos musicales, ofrecen una idea acerca del fraseo y la intención del canto, pero no indican alturas ni ritmos exactos. El conocimiento de la melodía, la cual era transmitida oralmente, era necesario para poder aplicar en ella el sentido musical que ofrecían los neumas [Arbonés and Milrud, 2014]. Por tanto, era muy grande la necesidad de un sistema de notación musical que permitiera transmitir y conservar las obras musicales a lo largo del tiempo, permitiendo que estas fueran interpretadas por músicos que nunca antes las habían escuchado.

El primer indicio de notación musical procede de las culturas del Creciente Fértil, en concreto de una tablilla datada hacia 2000 a.C. y encontrada en la localidad sumeria de Nippur, en el actual Iraq, que describe una pieza musical mediante instrucciones, dadas en escritura cuneiforme[1], de cómo debe ser representada [Arbonés and Milrud, 2014]. A pesar de que hace más de 4.000 años de las primeras manifestaciones de sistemas de notación musical, la notación usada actualmente cuenta con poco más de 400 años.

[1] Referido a ciertos caracteres de forma de cuña o de clavo, que algunos pueblos de Asia usaron antiguamente en la escritura.

El documento sobre el cual se plasma el sistema de notación musical y que por tanto, contiene la transcripción de una obra musical es conocido como partitura.

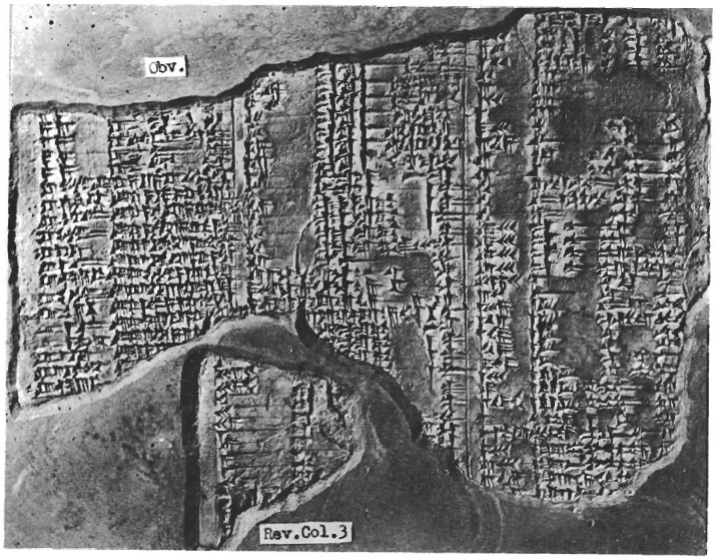


Figura 2.1: Tableta cuneiforme de Nippur. Primer ejemplo histórico de notación musical.

La partitura informa sobre aquellos aspectos técnicos de la música pero su interpretación cuenta con una parte subjetiva, la cual depende del músico quien, con sus conocimientos, sus decisiones y su expresión, dotará a la pieza musical de un mensaje único. Basta con oír diferentes versiones de una misma obra interpretada por distintos músicos para apreciar la diversidad de miradas.

2.1.1. Pentagrama

El modo gráfico de representar la música y por tanto, el elemento principal de una partitura, es el pentagrama (*staff*). Se trata de un conjunto de cinco líneas paralelas, horizontales y equidistantes entre sí sobre las que se escriben los signos musicales. Quedan así determinadas cinco líneas y cuatro espacios que se numeran de abajo a arriba:

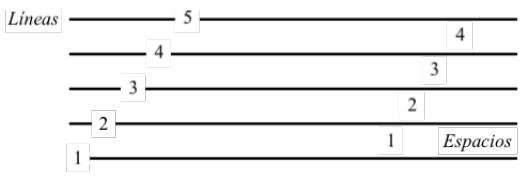


Figura 2.2: Disposión de las líneas y espacios de un pentagrama.

Los símbolos musicales son colocados de izquierda a derecha del pentagrama según el orden aparición y pueden representarse sobre las propias líneas del pentagrama o sobre los espacios que existen entre ellas. Las notas más agudas se sitúan en las posiciones más altas del pentagrama mientras que las más graves lo hacen en las más bajas del pentagrama. Con el fin de consignar notas más agudas o más graves se agregan líneas adicionales y por consecuente, espacios adicionales (figura 2.4). Los sonidos alineados verticalmente son simultáneos mientras que los alineados horizontalmente tienen la misma altura.

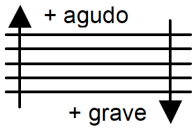


Figura 2.3: Altura de una nota según su posición en el pentagrama.

2.1.2. Representación de la altura

Las notas denotan una altura (*pitch*) concreta. Los nombres de las notas naturales en notación latina son DO, RE, MI, FA, SOL, LA y SI, siendo su equivalencia en notación anglosajona C, D, E, F, G, A y B, respectivamente. La altura de un sonido viene representada por la posición de este sobre el pentagrama, ya sea en una línea o en un espacio. Esta información es parcial: para conocer la altura absoluta de un sonido es necesario recurrir a la clave utilizada.

El primer elemento que aparece en un pentagrama es la clave (*clef*). Esta determina unívocamente la altura de los sonidos, dotando a las notas de cada línea o espacio con un nombre único. La clave más utilizada es la clave de sol (*treble clef* o *G-clef*). Esta clave atribuye SOL_4 a la nota que se escribe en la segunda línea. Sin embargo, también es común encontrarnos la clave de fa en cuarta o la clave de do en tercera. La clave de fa (*bass clef* o *F-clef*) atribuye a la cuarta línea la nota FA_3 y la clave de do en tercera (*alto clef* o *C3-clef*) atribuye a la tercera línea la nota DO_4. Así, una misma posición de una figura corresponde con distintas notas según la clave utilizada.

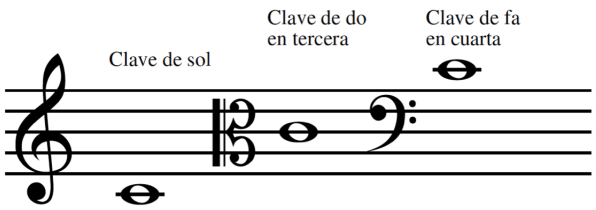


Figura 2.4: DO_4 representado en las claves más comunes.

Los nombres se repiten cada ocho notas (figura 2.5). Es ahí de donde nace el concepto de octava. Una octava es la distancia entre dos sonidos cuyas frecuencias fundamentales tienen una relación de dos a uno, es decir, la frecuencia fundamental de uno es el doble o la mitad del otro. Reciben el mismo nombre (normalmente acompañado de un número que referencia a su octava) pero les corresponden diferentes posiciones en el pentagrama. Las octavas cambian de número en el DO.

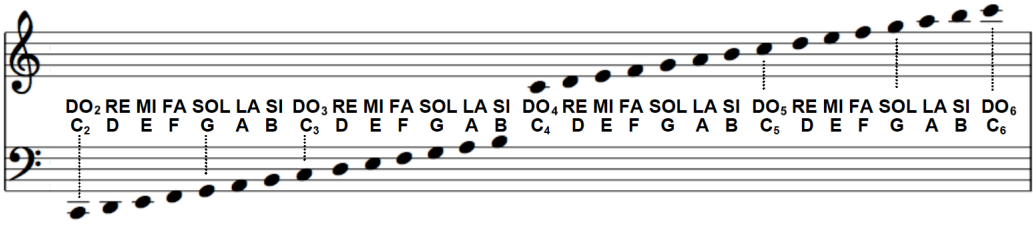


Figura 2.5: Alturas en las claves de sol y fa.

En la música occidental, cada octava se divide en doce notas equiespaciadas. La distancia que hay entre dos notas consecutivas de estas doce, se conoce como semitono. En la figura anterior, desde un DO al siguiente sólo hay seis notas más. Para representar las notas que faltan (hasta las 12) se utilizan una serie de símbolos llamados alteraciones:

- Sostenido (*sharp*): Aumenta la altura de la nota que acompaña en un semitono.
- Doble sostenido (*double sharp*): Aumenta la altura de la nota que acompaña en dos semitonos.
- Bemol (*flat*): Disminuye la altura de la nota que acompaña en un semitono.
- Doble bemol (*double flat*): Disminuye la altura de la nota que acompaña en dos semitonos.
- Becuadro (*natural*): Anula el efecto de cualquiera de las alteraciones anteriores.

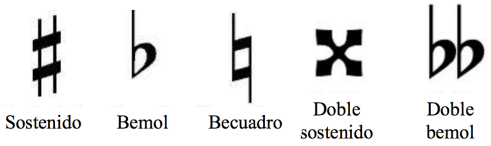


Figura 2.6: Símbolos correspondientes a las distintas alteraciones.

Estos signos se colocan sobre la línea o el espacio que se pretende modificar y alteran los sonidos situados a su derecha por el resto del compás. Cuando aparece al inicio de una obra (entre la clave y la métrica de compás) un conjunto de sostenidos o bemoles, se está indicando la tonalidad (*key*) de la partitura. Este conjunto de alteraciones es conocido como armadura e indica qué alturas serán alteradas durante toda la obra (o hasta un nuevo cambio).

2.1.3. Representación de la duración

El tiempo musical tiene una estructura jerárquica. La unidad básica de estructuración es el compás (*measure*). Los compases se representan en la partitura como líneas verticales (*bars*) que cruzan el pentagrama. El compás se divide en intervalos de igual duración llamados tiempos (pulsos o *beats*) y estos, a su vez, en partes (normalmente divisores enteros de la duración del pulso, como 1/2, 1/3, 1/4, etc.).

La notación numérica que hay al principio de un pentagrama recibe el nombre de métrica (*meter*) y denota el tipo de compás y cómo se rellena. Dicha notación viene indicada por medio de una fracción x/y[2], donde x, el numerador, indica el número de figuras que caben en el compás e y, el denominador, el tipo de figuras de las que se trata (expresado como división de la redonda, según la tabla 2.1).

Tabla 2.1: Denominador. Indica la duración de un pulso.

Tipo de nota	Redonda	Blanca	Negra	Corchea	Semicorchea	Fusa	Semifusa
Denominador	1	2	4	8	16	32	64

Por ejemplo, una métrica 4/4 (o 4 por 4, como se suele denominar) indica que cada compás se completa con 4 tiempos (por el numerador) de notas negras (por el denominador) o su duración equivalente con otras notas (cualquier combinación de notas que sume cuatro negras). En 7/8 cada compás se rellena con 7 tiempos de corcheas y 2/2, con 2 tiempos de blancas. El compás 4/4 estructura el tiempo de esta manera:

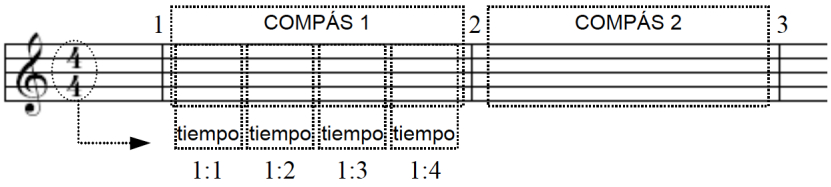


Figura 2.7: Estructuración del tiempo en un compás 4/4.

Si los compases de una obra musical tienen la misma duración y características, se dice que la obra es regular. Gracias a esta regularidad, basta con definir dichas características al comienzo de la obra.

Las duraciones proporcionales de las notas o de los silencios (*rests*) se representan mediante distintas figuras las cuales expresan fracciones o múltiplos de duración respecto al tiempo unidad (marcado por el denominador). La duración absoluta de una nota o de un silencio en segundos viene dada por el *tempo*: la "velocidad" a la que transcurre la música.

[2] Existe la posibilidad de representar con una C al compás 4/4 y con una ₵ al de 2/2.

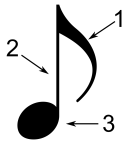


Figura 2.8: Partes de una nota: 1.- Corchete 2.- Plica 3.- Cabeza

Las notas y los silencios sólo aparecen en aquellos compases en los que "caben".

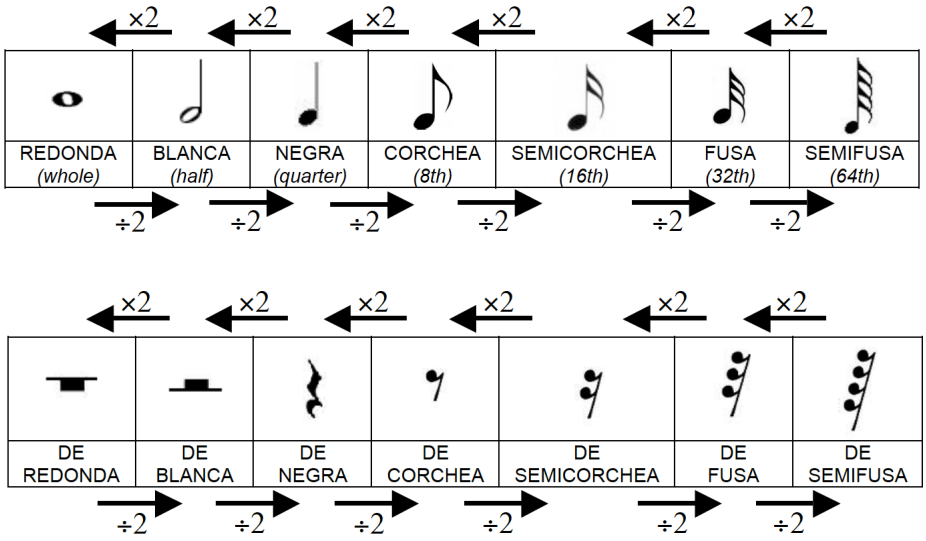


Figura 2.9: Figuras musicales para la representación de notas y silencios.

Podemos alargar la duración de una nota o un silencio en la mitad de su duración original añadiendo a su derecha un pequeño punto, conocido como puntillo (*dot*). Por ejemplo, si encontramos el puntillo al lado de una blanca (2 tiempos de duración respecto a una negra), pasaría a durar 3. Pueden añadirse sucesivos puntillos, que irían agregando las duraciones correspondientes a la cuarta parte, octava, etc.

Ligaduras

La ligadura (*tie*) es un signo de prolongación con forma de línea curva que sirve para unir dos figuras de nota de la misma altura (aunque no necesariamente del mismo valor) para que duren como una sola nota equivalente a la de la suma de las dos notas ligadas. Por ejemplo, una negra ligada a una corchea tiene el valor total de una negra con puntillo.

Si la ligadura es entre notas de distintas alturas sus duraciones no se suman. Se trata de una ligadura de expresión que indica al intérprete que debe dar continuidad a la interpretación de esa secuencia de notas, no interrumpiendo el sonido al cambiar de una a otra. Este tipo de ligadura se le conoce como legato (*slur*).

Grupos de valoración especial

Los grupos de valoración especial (*tuplet*) son grupos de figuras que toman una duración diferente de la que representan como grupo natural.

- Dosillo (*duplet*): grupo de dos notas que se articulan en el tiempo correspondiente a tres de ellas. Se les denota colocando una barra sobre ellas con el número 2.
- Tresillo (*triplet*): grupo de dos notas que se articulan en el tiempo correspondiente a dos de ellas. Se les denota colocando una barra sobre ellas con el número 3.

Existen otras combinaciones de duración, pero se usan mucho menos.

Acordes

Un acorde es un conjunto de tres o más notas que se hacen sonar simultáneamente. Dichas notas están alineadas verticalmente en el pentagrama, colocadas una sobre otra (excepto en las ocasiones donde se superponen).

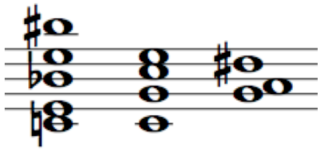


Figura 2.10: De izquierda a derecha, acordes de 5, 4 y 3 notas.

2.2. Humdrum

Humdrum[3] es un conjunto de herramientas de línea de comandos que facilita el análisis musical, así como una sintaxis generalizada para representar flujos secuenciales de datos.

[3] http://humdrum.org

Debido a que es un conjunto de herramientas de línea de comandos, es independiente del lenguaje de programación.

David Huron creó Humdrum en los años 80 con el propósito de facilitar el planteamiento y la respuesta de preguntas de investigación en el campo de la música. Humdrum permite a los investigadores encadenar, manipular y producir una gran variedad de representaciones musicalmente correctas.

2.2.1. El código Kern

La representación Humdrum llamada *kern*[4] permite la representación básica de la notación musical tradicional occidental: tono, duración y sonoridad. Esta representación será la usada para codificar el contenido de las partituras musicales que el generador creará.

La representación *kern* es compatible con los siguientes significantes de notación musical [Selfrigde-Field, 1997]:

TONO: afinación (notación occidental), alteraciones, claves, posición de clave, tonalidades, tono, armónicos, glissandi, arpegios;

DURACIÓN: duraciones musicales canónicas, silencios, puntillos, grupos de valoración especial, gruppetos, acciaccaturas, ligaduras, tempo (pulsaciones por minuto), unidades métricas, eventos indefinidos o sin duración;

ARTICULACIÓN Y ORNAMENTACIÓN: staccato, spiccato, tenuto, fermata, pizzicato, respiro, attacca, acento, sforzando, articulación genérica, trinos, mordente, mordente invertido, giro, giro invertido (Wagnerian), adornos genéricos;

TIMBRE: nombre del instrumento, clase de instrumento;

OTROS: marcas de frase, legatos, marcadores de elisión, barras de compás, dobles barras, barras de repetición, barras punteadas, barras invisibles, números de compás, arreglo de pentagrama, arco hacia arriba, arco hacia abajo, corchetes, dirección de la plica;

EDITORIAL: ossias, sic, marcadores de interpretación editorial, marcadores de intervención editorial, notas de pie de página, comentarios globales, comentarios locales, símbolos definidos por el usuario.

[4] Estrictamente, dentro del contexto de otras representaciones de Humdrum, el nombre es **kern.

La sintaxis de codificación *kern* se desarrolla verticalmente hacia abajo, es decir, todo el flujo de datos viene dado en una columna (figuras 2.11 y 2.12), donde cada línea indica la codificación de un símbolo musical. Cada columna de datos comienza con la palabra clave ****kern**, que indica que el material codificado se ajusta a la representación *kern*, y finaliza con la expresión ***-**.

A continuación, se tratarán con mayor detalle aquellos elementos de notación musical considerados de mayor importancia para el desarrollo del generador automático de partituras musicales.

Elementos generales de notación y organización

A. LÍNEAS DE PENTAGRAMA [*|.]

Las líneas de pentagrama pueden ser codificadas de manera específica. Algunos ejemplos son los siguientes:

Tabla 2.2: Ejemplos de codificación en *kern* de las líneas de pentagrama.

*|.||||	pentagrama de cuatro líneas
*|.|	pentagrama de una línea
*|.0	sin líneas de pentagrama
*|.||R|	pentagrama de cuatro líneas con la tercera (empezando por abajo) coloreada en rojo
*|.|X|	pentagrama de tres líneas con la línea del medio invisible
*|.:::::	pentagrama de cinco líneas puntuadas

Si el número de líneas no es especificado, se creará un pentagrama que tenga cinco líneas negras continuas. Además del color por defecto (negro), las líneas del pentagrama pueden ser rojas (ruber-R), verdes (viridis-V) y azules (caeruleus-C).

B. POSICIÓN DEL PENTAGRAMA [*staff]

El orden de los pentagramas dentro de un sistema puede codificarse explícitamente por posición absoluta o relativa. A continuación, se muestran algunos ejemplos:

Tabla 2.3: Ejemplos de codificación en *kern* de la posición del pentagrama.

*staff1	pentagrama superior del sistema
*staff12	doceavo pentagrama del sistema empezando por arriba
*staff$	pentagrama inferior del sistema
*staff$-1	pentagrama encima del pentagrama inferior del sistema

Cuando se usa un único pentagrama no es necesario especificar su posición.

C. CLAVES [*clef]

La información acerca de la clave usada se codifica explícitamente. Algunas de las posibles codificaciones son:

Tabla 2.4: Ejemplos de codificación en *kern* de las claves.

*clefG2	clave de sol en segunda
*clefF4	clave de fa en cuarta
*clefC3	clave de do en tercera
*clefC4	clave de do en cuarta
*clefG1	clave de sol en primera
*clefX	no hay clave
*clefGv2	clave de sol; 8va bassa

D. TONALIDADES [*k]

Kern distingue entre tono y tonalidades. La tonalidad de la partitura facilita la reconstrucción de la fuente musical mientras que el tono facilita el análisis del contenido musical. Una tonalidad es codificada mediante la expresión *k seguida de una lista de tonos alterados entre corchetes: *k[f#g#c#].

Los sostenidos, los bemoles y los becuadros son representados con el símbolo almohadilla, el signo menos y la letra n minúscula, respectivamente. Las dobles alteraciones se representan con la repetición de su símbolo de codificación. Conociendo esto, situaciones donde se mezclen tanto bemoles como sostenidos (*k[f#b-]) como becuadros (*k[bnen]) pueden codificarse fácilmente.

Las especificaciones de tono son codificadas con un asterisco y finalizadas con dos puntos. Las letras mayúsculas designan modos mayores, las letras minúsculas los menores. Normalmente, la codificación de tono se indica tras la de tonalidad pero puede aparecer por si sola en aquellos casos donde el tono cambia pero no lo hace la tonalidad.

Tabla 2.5: Ejemplos de codificación en *kern* de tonos.

*G:	SOL mayor
*d:	RE menor
*e-:	MI♭ menor
*?:	Tono desconocido
*X:	Pasaje atonal (no hay tono)

Representación de la altura

A. ALTURA [A..G]

Las alturas de las notas están representados por letras. *Kern* codifica alturas absolutas.

Las alturas son representadas mediante un esquema de letras mayúsculas y minúsculas, siguiendo el sistema de notación musical anglosajón. El DO central (DO_4) se representa con la letra minúscula "c". Las octavas sucesivas se designan por repetición de letras, por lo que DO_5 se representa con "cc", DO_6 con "ccc" y así sucesivamente. Cuanto mayor sea la octava, mayor la repetición de la letra.

Para alturas por debajo de DO_4, se usan letras mayúsculas: C designa DO_3, CC designa DO_2, y así sucesivamente. Los cambios de octava ocurren entre B y C. Por lo tanto, el SI debajo del DO central se representa como B; el SI debajo de DO_2 se representa como BBB, y así sucesivamente. Cuanto menor sea la octava, mayor será el número de letras repetidas.

Tabla 2.6: Ejemplos de codificación en *kern* de alturas.

c	DO (DO_4)
B	SI_3
C	DO_3
cc	DO_5
d#	$RE\sharp_4$
e-	$MI\flat_4$

Los silencios son representados con la letra minúscula r.

Representación de la duración y del acento

A. MÉTRICA [*M<*n*>/<*n*>]

Kern pueden representar tanto unidades métricas comunes como inusuales. Estas son introducidas por el identificador *M. Los tipos de compases más comunes son representados mediante un numerador y un denominador separados por una barra inclinada (por ejemplo, *M2/4). También se pueden especificar agrupaciones de tiempos más complejos (compases de amalgama) por números enteros separados por un signo más (+) en el numerador. Únicamente se permiten valores enteros. Se muestran algunos ejemplos de codificación en la tabla 2.7.

Tabla 2.7: Ejemplos de codificación en *kern* de métricas de compás.

COMPÁS	TIPO	SIGNIFICADO
*M2/4	Compás binario de subdivisión binaria	Dos negras por compás
*M3/4	Compás ternario de subdivisión binaria	Tres negras por compás
*M4/4	Compás cuaternario de subdivisión binaria	Cuatro negras por compás
*M5/4	Compás irregular de subdivisión binaria	Cinco negras por compás
*M6/8	Compás binario de subdivisión ternaria	Seis corcheas por compás
*M9/8	Compás ternario de subdivisión ternaria	Nueve corcheas por compás
*M12/8	Compás cuaternario de subdivisión ternaria	Doce corcheas por compás
*M3+2/4	Compás irregular (Compás de amalgama)	Cinco negras por compás
*M?	Compás desconocido	
*MX	Compás amétrico	Sin medida

B. DURACIÓN [2, 4, 8, ...]

Las duraciones de las notas se representan utilizando una notación numérica recíproca. A excepción del valor cero, las duraciones se indican mediante números recíprocos correspondiente a los nombres de duración estadounidenses: “1” para la redonda, “8” para la corchea, “32” para la fusa, etc. Estos números indican cuántas notas de esa duración son necesarias para obtenerar la duración de una nota redonda. El número cero (0) está reservado para la cuadrada (figura musical que posee una duración equivalente a dos redondas ligadas).

Los grupos de valoración especial como los tresillos, se codifican siguiendo el mismo patrón explicado en el párrafo anterior. Las corcheas de tresillo son representadas por el número “12” porque 12 de ellas equivalen a una redonda. De la misma manera, las semicorcheas de tresillo son representadas por el número “24” porque 24 de ellas equivalen a una redonda.

Para codificar unidades musicales simultáneas, como son los acordes, se codifica cada nota perteneciente al acorde dejando entre ellas un espacio en blanco. Deberán repetirse los ritmos y articulaciones para cada nota, pero no las ligaduras o los corchetes. De manera que un acorde de tres negras, DO_4, MI_4 y SOL_4, respectivamente, sería codificado como “4c 4e 4g”.

Para indicar que una nota lleva puntillo agregamos un punto (.) a la derecha del número que indica la duración de la nota, por lo tanto, “8.” indica que la nota es una corchea con puntillo y “2..” representa a una blanca con doble puntillo.

C. BARRAS DE COMPÁS [=]

Las barras de compás se identifican con el signo igual (=) seguido de un número que indica el número de compás. Si a ello, le sigue un signo menos (-), la barra de compás

no será visible en la representación gráfica de la partitura. Una doble barra de compás es representada por dos signos iguales sucesivos (==).

La doble barra final se codifica mediante un signo igual seguido de una barra vertical y un signo de cierre de exclamación (=|!).

D. LIGADURAS, LEGATOS y FRASEOS MUSICALES [[..], (..), {..}]

Kern hace distinción entre ligaduras, legatos (ligadura de expresión) y fraseos musicales.

El corchete de apertura [denota la primera nota de una ligadura y el de cierre], de manera consecuente, denota la última nota de una ligadura. Las ligaduras de expresión y los fraseos musicales se indican de la misma manera usando paréntesis (..) y llaves {..}, respectivamente.

Los legatos y los fraseos puede anidarse (por ejemplo, legatos dentro de legatos) y también se pueden elidir (por ejemplo, fraseos superpuestos) a una única profundidad. Las marcas de anidación significan que una ligadura de expresión o un fraseo está completamente subsumida bajo otra ligadura u otro fraseo. Por ejemplo: (()) significa que hay una ligadura incluida dentro de otra más larga.

Las elisiones son superposiciones, es decir, cuando un fraseo no ha terminado pero hay otro que comienza. En *kern*, el signo ampersand (&) se usa para marcar legatos o fraseos elididos. Por ejemplo: { &{ } &} significa que dos fraseos se superponen: el primer fraseo termina después de que el segundo comience.

Información visual

A. DIRECCIÓN DE LA PLICA [/, \]

Las direcciones de la plica se representan con la barra inclinada (/) para indicar que la plica va hacia arriba y la barra inclinada invertida (\) para indicar que va hacia abajo. Si la dirección de la plica no es indicada, *kern* decidirá cuál es la más adecuada.

B. CORCHETES [L, J, k, K]

Tanto corchetes "completos" como corchetes "parciales[5]" pueden ser representados en *kern*. Las letras se repiten por cada corchete presente.

[5] Los corchetes parciales son aquellos que se dan cuando se unen dos notas de distinta duración, quedando una de las líneas de unión a mitad para indicarlo.

Tabla 2.8: Ejemplos de codificación en *kern* de corchetes.

L	Abre un corchete
J	Cierra un corchete
LL	Abre dos corchetes
JJ	Cierra dos corchetes
k	Corchete "parcial" con dirección a la izquierda
K	Corchete "parcial" con dirección a la derecha
kk	Dos corchetes "parciales" con dirección a la izquierda
kK	Dos corchetes "parciales" con dirección a la izquierda y a la derecha, respectivamente

2.3. Verovio

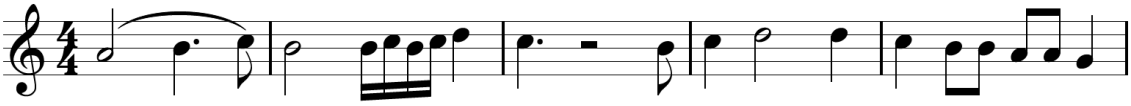
Verovio[6] es una librería musical, rápida y portátil, escrita en C++ que permite grabar partituras musicales codificadas a través del sistema de notación musical MEI (*Music Encoding Initiative*)[Pugin et al., 2014]. Las partituras musicales son grabadas en archivos SVG (*Scalable Vector Graphics*), que son, en pocas palabras, imágenes vectoriales. Además, del sistema de notación musical MEI, Verovio puede recibir otros lenguajes de codificación como MusicXML, Paline, Easie o Humdrum. Estos son convertidos a MEI para poder renderizar el archivo SVG de la partitura deseada.

Verovio se puede utilizar o bien como una herramienta de línea de comandos o bien se puede compilar en JavaScript utilizando el compilador LLVM-to-JavaScript.

En el presente proyecto, será usada la parte de desarrollo de Humdrum dentro de Verovio[7], que permitirá que las partituras codificadas en *kern* por el generador automático sean transcritas gráficamente. Su uso será tratado con más detalle en el capítulo siguiente.

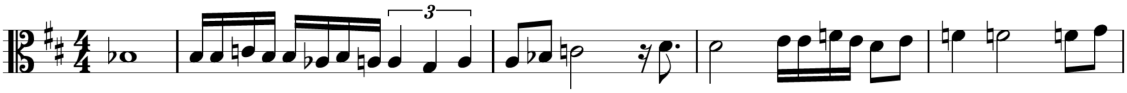
Antes de pasar al siguiente capítulo, se muestran dos ejemplos que engloban gráficamente la información tratada a lo largo de este capítulo. Los ejemplos son extractos de partituras musicales acompañados de su codificación en el lenguaje *kern*. La codificación en *kern* será creada por el generador automático y gracias a Verovio, será representada gráficamente.

[6] http://www.verovio.org/index.xhtml
[7] http://verovio.humdrum.org



Inicio de la representación en **kern**	**kern
Clave de **sol**, 2ª línea de pentagrama	*clefG2
Tonalidad (Sin alteraciones). DO Mayor/La Menor.	*k[]
Métrica, 4/4	*M4/4
Primera barra de compás (**invisible**)	=1-
Inicio de ligadura, blanca LA4	(2a
Negra con **puntillo** SI4	4.b
Corchea DO5, fin de ligadura	8cc)
Segunda barra de compás	=2
Blanca SI4	2b
Semicorchea SI4, **inicio corchete**	16bL
Semicorchea DO5	16cc
Semicorchea SI4	16b
Semicorchea DO5, fin corchete	16ccJ
Negra RE5	4dd
Tercera barra de compás	=3
Negra con **puntillo** DO5	4.cc
Silencio de blanca	2r
Corchea SI4	8b
Cuarta barra de compás	=4
Negra DO5	4cc
Blanca RE5	2dd
Negra RE5	4dd
Quinta barra de compás	=5
Negra DO5	4cc
Corchea SI4, **inicio corchete**	8bL
Corchea SI4, fin corchete	8bJ
Corchea LA4, **inicio corchete**	8aL
Corchea LA4, fin corchete	8aJ
Negra SOL4	4g
Sexta barra de compás	=6
Fin de la representación en **kern**	*-

Figura 2.11: Extracto musical nº 1 acompañado de su codificación en el lenguaje *kern*.



Inicio de la representación en **kern**	**kern
Clave de **do**, 3ª línea de pentagrama	*clefC3
Tonalidad (FA4♯ DO4♯)	*k[f#c#]
Métrica, 4/4	*M4/4
Primera barra de compás (**invisible**)	=1-
Blanca SI3♭	1B-
Segunda barra de compás	=2
Semicorchea SI3, **inicio corchete**	16BL
Semicorchea SI3	16B
Semicorchea DO4♮	16cn
Semicorchea SI3, fin corchete	16BJ
Semicorchea SI3, **inicio corchete**	16BL
Semicorchea LA3♭	16A-
Semicorchea SI3	16B
Semicorchea LA3♮, fin corchete	16AJ
Negra tresillo LA3	6A
Negra tresillo SOL3	6G
Negra tresillo LA3	6A
Tercera barra de compás	=3
Corchea LA3, **inicio corchete**	8AL
Corchea SI3♭, fin corchete	8B-J
Blanca DO4♮	2cn
Silencio de semicorchea	16r
Corchea con **puntillo** RE4	8.d
Cuarta barra de compás	=4
Blanca RE4	2d
Semicorchea MI4, **inicio corchete**	16eL
Semicorchea MI4	16e
Semicorchea FA4♮	16fn
Semicorchea MI4, fin corchete	16eJ
Corchea RE4, **inicio corchete**	8dL
Corchea MI4, fin corchete	8eJ
Quinta barra de compás	=5
Negra FA4♮	4fn
Blanca FA4♮	2fn
Corchea FA4♮, **inicio corchete**	8fnL
Corchea SOL4, fin corchete	8gJ
Sexta barra de compás	=6
Fin de la representación en **kern**	*-

Figura 2.12: Extracto musical nº 2 acompañado de su codificación en el lenguaje *kern*.

3 Diseño del sistema de generación automática de datos etiquetados

En este capítulo, se detallará el proceso de implementación de un sistema de generación automática de datos etiquetados para música monofónica. También, se abordará con detalle el lenguaje de transcripción elegido.

3.1. Lenguaje de programación

El lenguaje de programación elegido para el desarrollo del generador automático ha sido **C++**. Considerado un lenguaje de programación de nivel medio, C++ se caracteriza por su eficacia, rapidez y versatilidad, soportando tanto programación estructurada e imperativa como programación orientada a objetos. Esto último hace que se le denomine un lenguaje de programación multiparadigma, definido por su propio creador, Bjarne Stroustrup, como "lenguaje que permite crear programas usando más de un estilo de programación". Además, es un lenguaje multiplataforma. Todas estas características, convierten a C++ en uno de los lenguajes de programación más utilizados.

Figura 3.1: Logo de C++.

La versión de C++ utilizada en este trabajo ha sido la versión **C++17**, por ser la última versión estable de C++ al inicio de este proyecto.

Una de las características principales de C++ es su compatibilidad con el lenguaje C, ya que es considerado una extensión de este último. A lo largo del proyecto, se combinan ambos lenguajes de programación.

El entorno de desarrollo escogido ha sido Visual Studio Code.

Para el desarrollo del generador automático de partituras musicales, se ha hecho uso de una serie de librerías de las bibliotecas estándar de C y C++, necesarias para la declaración e implementación de numerosas tareas presentes en el generador. A continuación, se describen las librerías usadas.

Librerías

- **iostream**: Librería de la biblioteca estándar de C++ que aporta los elementos fundamentales para las operaciones de entrada y salida en C++.
- **string**: Librería de la biblioteca estándar de C++ que contiene las clases y plantillas estándares para trabajar con cadenas de caracteres.
- **stdlib.h**: Librería de la biblioteca estándar de C que almacena los prototipos de funciones para gestión de memoria dinámica y control de procesos, entre otros.
- **time.h**: Librería de la biblioteca estándar de C usada para el tratamiento de formatos de fecha y hora.
- **fstream**: Librería de la biblioteca estándar de C++ que provee los elementos fundamentales para las operaciones de entrada y salida de archivos.
- **random**: Librería numérica de la biblioteca estándar de C++ que facilita la generación de números pseudo-aleatorios.

3.2. Esquema general del sistema de generación automática de datos etiquetados

Tras documentarse acerca de la codificación musical en *kern*, sus capacidades y limitaciones, se ha pasado a desarrollar un sistema de generación automática de datos etiquetados para música monofónica. Este sistema, mediante una serie de parámetros de entrada, devuelve dos salidas: la partitura en formato PDF y la transcripción de esta en formato txt. Con ambas salidas, se obtiene el par necesario usado en los algoritmos de aprendizaje automático.

La estructura del sistema es la expuesta en la figura 3.2. Como se observa, el usuario introducirá cinco parámetros que permitirán determinar la complejidad gráfica de la

partitura musical a generar. Los parámetros a introducir son:

- **Nombre de la partitura**. Este nombre será el que recibirá tanto el archivo que contiene la transcripción como el archivo que contiene la representación gráfica.
- **Número de compases que tendrá la partitura**.
- **Método deseado para el cálculo de la altura de las notas**. El usuario podrá elegir entre tres métodos posibles: *distribución normal*, *random walk* o *ecuación logística*. En caso de que el usuario haya elegido el método de cálculo por la ecuación logística se le pedirá que introduzca un parámetro más, ligado a esta probabilidad. Este parámetro será un número entre 3,5 y 4.
- **Probabilidad (entre 0 y 100) de que una nota esté alterada**.
- **Probabilidad (entre 0 y 100) de que una figura musical sea un acorde o un tresillo**.

Una vez que el usuario ha introducido los parámetros anteriores, el generador automático se encarga de codificar una partitura acorde a ellos, generando la codificación de la partitura en el lenguaje *kern* y la transcripción esperada de esta.

La codificación de la partitura en *kern* queda guardada en un archivo Humdrum (.kern), que es pasado como parámetro de entrada al ejecutable de Verovio. Como se ha explicado en la sección 2.3, Verovio puede ser utilizado como una herramienta de línea de comandos. De manera que, ejecutando el siguiente comando:

$ verovio archivo.kern

Se creará un archivo llamado archivo.svg con la notación musical gráfica representada en el archivo Humdrum archivo.kern. Para transformar el archivo svg a un archivo de tipo PDF, se ha usado la instrucción de línea de comando **svg2pdf**[1]. Para realizar dicha conversión, únicamente habrá que ejecutar el siguiente comando:

$ svg2pdf archivo.svg archivo.pdf

La transcripción de la partitura queda guardada en un archivo .txt. Dicho archivo contiene la transcripción esperada de la partitura generada, usando la gramática propuesta en el proyecto **HispaMus** (*Handwritten Spanish Music Heritage Preservation by Automatic Transcription*). Dicha gramática será explicada con detalle en la sección 3.4.1.

[1] http://macappstore.org/svg2pdf/

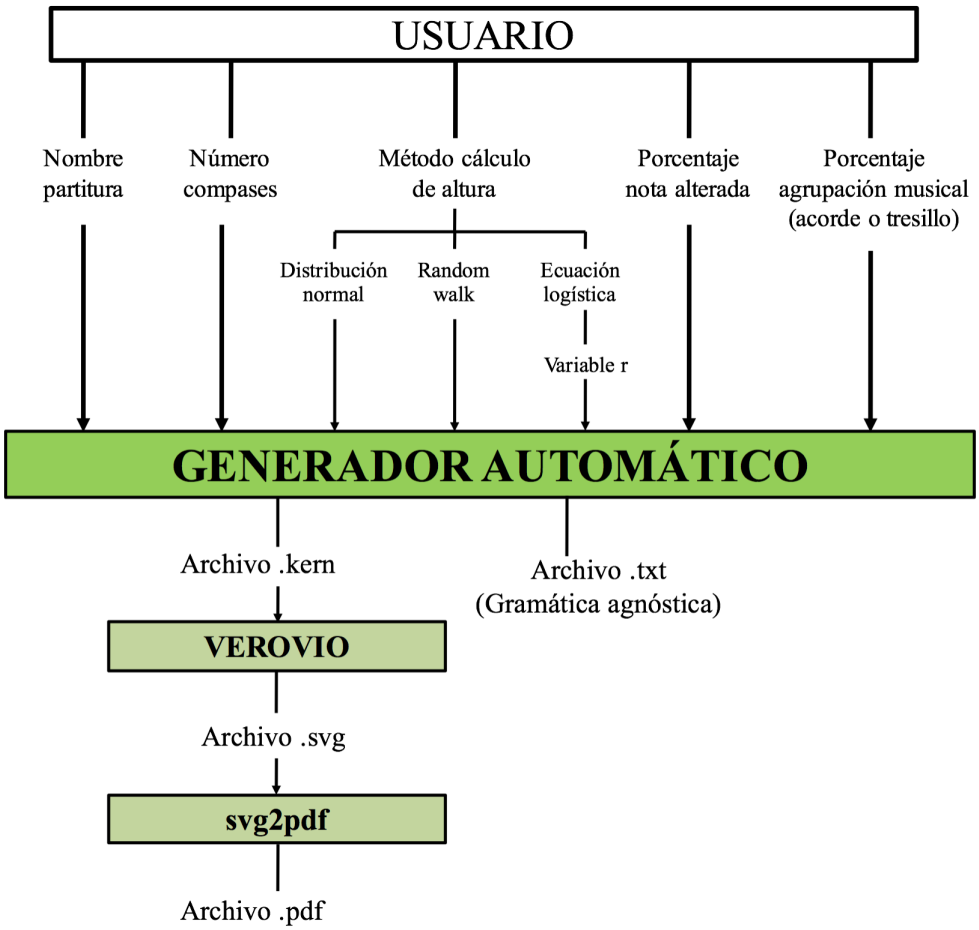


Figura 3.2: Esquema general del sistema propuesto.

3.3. Esquema general del generador automático

El archivo principal del sistema de generación automática de datos etiquetados es el archivo programado en C++ (.cpp). Este archivo cuenta con una función principal, la función **main**, dentro de la cual se inicializan los parámetros necesarios para las llamadas a las distintas funciones que permiten generar una partitura musical codificada en el lenguaje musical *kern* de Humdrum.

Por lo que, primero se inicializará el archivo Humdrum .kern, donde se guardará toda la codificación en *kern* de la partitura que se genere. Se procederá entonces a calcular los primeros contenidos musicales:

1. **Clave a usar**.

2. **Tonalidad de la partitura**.

3. **Métrica de compás**.

Una vez calculados los contenidos anteriores, los cuales serán fijos a lo largo de toda la partitura, se procederá a calcular los distintos símbolos musicales que rellenarán los compases de la partitura. Para ello, se ejecutará un bucle principal dentro del cual se localizarán las llamadas a las distintas funciones usadas para codificar los símbolos musicales que aparecerán en la partitura. Se puede observar la estructura de este bucle, el cual se ejecutará hasta que el número de compases sea igual al deseado por el usuario, en la figura 3.3.

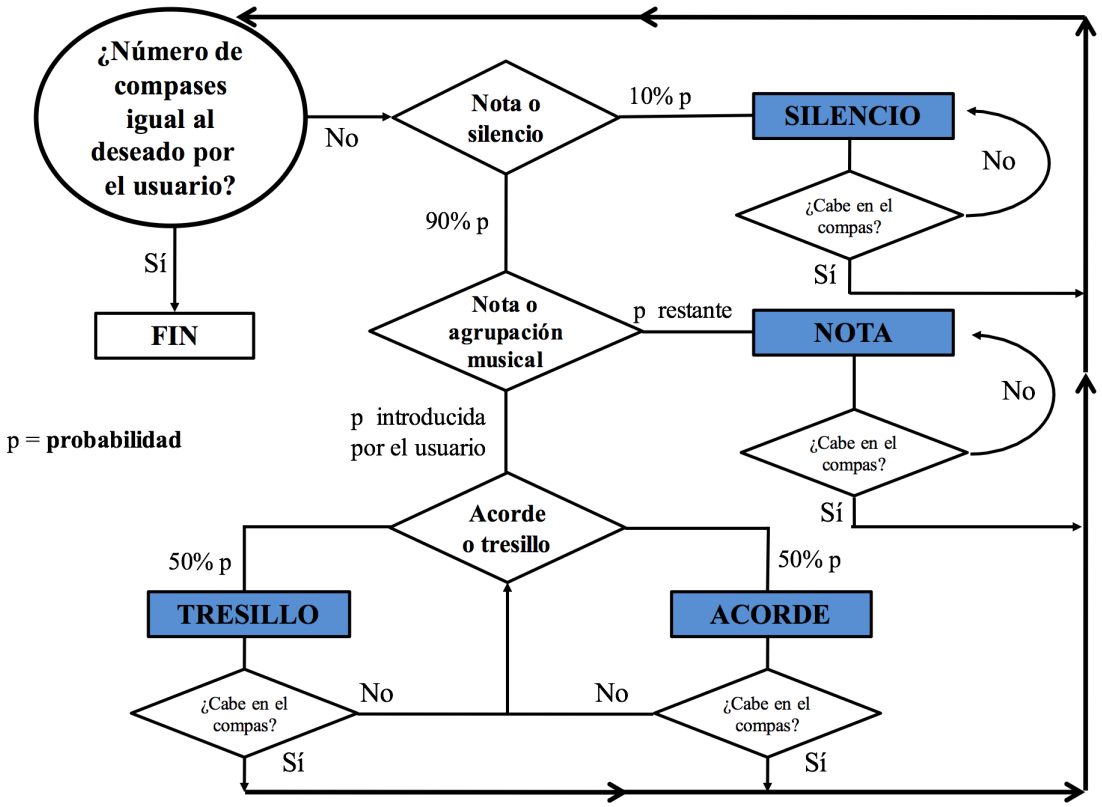


Figura 3.3: Esquema general del generador automático.

Cada una de las funciones usadas para determinar contenido musical, devolverá una secuencia de caracteres que contendrá la codificación en *kern* de dicho contenido musical, la cual se guardará en el archivo de Humdrum.

Una vez que se haya completado la partitura con los compases deseados por el usuario, se cerrará el archivo Humdrum .kern.

A continuación, se explican con más detalle cada una de las funciones involucradas en la determinación del contenido musical de la partitura.

3.3.1. Función para determinar la clave

Esta función recibe un número entero que ha sido elegido aleatoriamente. El número recibido puede ser 0, 1 o 2, estando cada uno de ellos asociado a una clave:

- Si el número recibido es **0**, la clave a codificar será la de **sol**.
- Si el número recibido es **1**, la clave a codificar será la de **fa en cuarta**.
- Si el número recibido es **2**, la clave a codificar será la de **do en tercera**.

Se ha decidido codificar únicamente estas tres claves por ser las más usadas.

Por tanto, mediante una estructura de control, se guardará la secuencia de caracteres correspondiente a codificar en *kern* según la clave seleccionada. Dicha secuencia de caracteres será lo que devuelva la función cuando sea llamada. Además, cuando se guarda la secuencia de caracteres a codificar, se imprime seguidamente por pantalla la estructura de la gramática agnóstica asociada a la clave elegida.

3.3.2. Función para determinar la tonalidad

La tonalidad de la partitura será o bien elegida aleatoriamente, pudiendo ser cualquiera de las tonalidades pertenecientes al círculo de quintas (figura 3.4) o bien DO Mayor/LA Menor si el usuario indica que no quiere alteraciones. Esta información vendrá guardada en un número entero que será uno de los dos parámetros de entrada de esta función. El otro parámetro de entrada será un vector entero de siete ceros (uno por cada nota natural) que será usado para identificar qué alturas son alteradas en cada tonalidad.

Dentro de la función, se implementará una estructura de control basada en el entero de entrada. Esta estructura contará con 32 casos posibles:

- 0-Do mayor (C); 1-La menor (a)
- 2-Sol mayor (G); 3-Mi menor (e)
- 4-Re mayor (D); 5-Si menor (b)

- 6-La mayor (A); 7-Fa sostenido menor (f♯)
- 8-Mi mayor (E); 9-Do sostenido menor (c♯)
- 10-Si mayor (B); 11-Sol sostenido menor (g♯)
- 12-Fa sostenido mayor (F♯); 13-Re sostenido menor (d♯)
- 14-Do sostenido mayor (C♯); 15-La sostenido menor (a♯)
- 16-Fa mayor (F); 17-Re menor (d)
- 18-Si bemol mayor (B♭); 19-Sol menor (g)
- 20-Mi bemol mayor (E♭); 21-Do menor (c)
- 22-La bemol mayor (A♭); 23-Fa menor (f)
- 24-Re bemol mayor (D♭); 25-Si bemol menor (b♭)
- 26-Sol bemol mayor (G♭); 27-Mi bemol menor (e♭)
- 28-Do bemol mayor (C♭); 29-La bemol menor (a♭)
- 30-Sin alteraciones; 31-Sin alteraciones

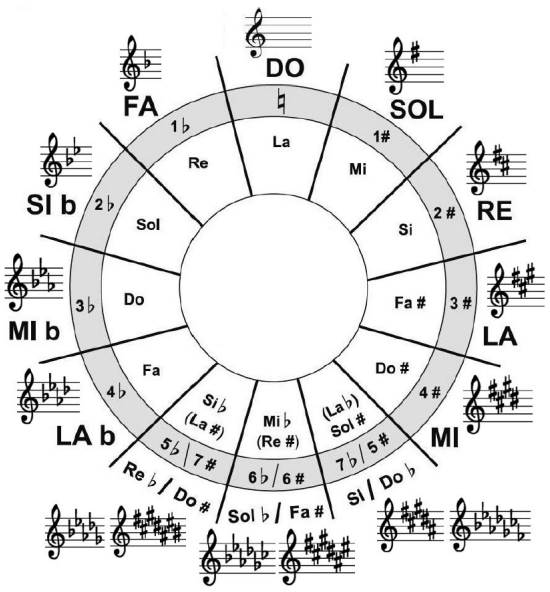


Figura 3.4: Círculo de quintas.

Los casos están agrupados en pares pues cada quinta mayor tiene su correspondiente menor. Por cada par de casos, se guardará la secuencia de caracteres correspondiente a codificar en *kern* según la tonalidad seleccionada (está secuencia será el parámetro de retorno de la función) y se guardará también en el vector de enteros las notas alteradas según su orden de aparición en la armadura de la tonalidad.

Para ello a cada nota natural le hemos asociado un número:

Tabla 3.1: Entero identificador para cada nota natural.

Nota	A (LA)	B (SI)	C (DO)	D (RE)	E (MI)	F (FA)	G (SOL)
Entero asociado	1	2	3	4	5	6	7

De manera que si la tonalidad elegida es RE Mayor/SI Menor, el vector de enteros pasará de ser {0, 0, 0, 0, 0, 0, 0} para ser {6, 3, 0, 0, 0, 0, 0}, pues en dichas tonalidades la primera altura modificada es un FA y la siguiente, un DO.

Además, conforme se guarde la información asociada a la tonalidad se imprimirá por pantalla la estructura de la gramática agnóstica de las alteraciones que formen parte de su armadura.

En el círculo de quintas se observa que hay 30 tonalidades posibles, pero la estructura de control presenta 32. Esto se debe a que los dos últimos casos, llamados *sin alteraciones*, hacen referencia a las tonalidades de DO Mayor/LA Menor, las cuales no presentan ninguna alteración en su armadura. Se ha decidido añadir estos dos casos para dotar de una mayor probabilidad a estas tonalidades, ya que son más comunes en las partituras.

3.3.3. Función para determinar la métrica de compás

Se podrán implementar los siguientes tipos de compás: **4/4**, **2/2**, **3/4**, **2/4**, **6/8**, **12/8**, **9/8** y **5/4**, por ser considerados los más comunes. A su vez, como no todos ellos son igual de comunes, contarán con distintas probabilidades de ser elegidos. Es por ello, que esta función recibe:

- Un flotante, en el que se guardará la duración del compás en relación a la duración de un tiempo de negra. Es decir, si el compás elegido es 3/4, dicho flotante tendrá el valor de 3; mientras que si el compás elegido es 9/8, el flotante valdrá 4,5.

- Un entero cuyo valor será elegido aleatoriamente entre 1 y 200 y sobre el cual se basará la estructura de probabilidad que decidirá el tipo de métrica.

- Un entero que servirá para identificar el tipo de compás que ha sido seleccionado (tabla 3.2).

Tabla 3.2: Entero identificador para cada tipo de compás.

Compás	4/4	2/2	3/4	2/4	6/8	12/8	9/8	5/4
Entero asociado	1	2	3	4	5	6	7	8

Por tanto, la función está basada en una estructura de probabilidad, en la que para cada probabilidad (asociada a un tipo de compás, tabla 3.3) se indica la duración, el entero identificador y la secuencia de caracteres (parámetro de retorno de la función) correspondiente a codificar en *kern* según el compás seleccionado. Seguidamente, se imprime por pantalla la estructura de la gramática agnóstica de la métrica correspondiente.

Tabla 3.3: Porcentaje de probabilidad asociado a cada compás.

Compás	4/4	3/4	2/2	2/4	6/8	12/8	9/8	5/4
Probabilidad	50 %	25 %	12 %	%5	2 %	2 %	2 %	2 %

Como se ha indicado con anterioridad, existe la posibilidad de representar con una C al compás 4/4 y con una 𝐂 al de 2/2. Es por ello, que se permite codificar el tipo de compás con ambas representaciones, eligiendo una de ellas aleatoriamente (ambas cuentan con la misma probabilidad).

3.3.4. Función para determinar el tipo de silencio

La función planteada para codificar los silencios recibe dos parámetros: un flotante que indica cuántos tiempos se han completado de la duración total del compás y un entero que identifica el tipo de compás que ha sido seleccionado.

Dentro de la función se crean tres parámetros:

- Una secuencia de caracteres en la cual se guardará la codificación en *kern* del silencio elegido. Esta secuencia es el parámetro de retorno de la función.

- Un entero cuyo valor será elegido aleatoriamente, entre 0 y 4, y sobre el cual se basará la estructura de control que decidirá el tipo de silencio. Cada uno de los posibles valores de este entero está asociado a un tipo de silencio, figura 3.4.

Tabla 3.4: Entero identificador para cada tipo de silencio.

Silencio	De redonda	De blanca	De negra	De corchea	De semicorchea
Entero asociado	0	1	2	3	4

- Un entero cuyo valor será elegido aleatoriamente entre 0 y 19 y sobre el cual se basará la estructura de probabilidad que decidirá si el silencio lleva puntillo o no.

Para cada caso (tipo de silencio) de la estructura de control, habrá dos posibilidades de codificación: silencio con puntillo o silencio sin puntillo. La primera de ellas solo tendrá un 5 % de probabilidad de ocurrir, pues solo se dará si el entero nombrado anteriormente toma un valor determinado (como puede tomar 20 valores distintos, cada uno de ellos tiene un 5 % de probabilidad). Se ha elegido este valor de probabilidad para un silencio con puntillo debido a que se ha considerado que no son muy comunes, pero en cualquier momento podría alterarse dicha probabilidad y cambiar su valor por uno más alto o más bajo, según sea decidido.

Tanto para el caso del silencio con puntillo como para el caso del silencio sin puntillo, se comprobará si dicho silencio "cabe" en el compás. Si así es se guardará la secuencia de caracteres correspondiente a codificar en *kern*, se imprimirá por pantalla la estructura de la gramática agnóstica asociada y se le sumará al flotante de entrada que indicaba los tiempos completados de la duración total del compás el valor en tiempos de negra del silencio elegido.

En el caso de que el silencio elegido sea de semicorchea, no existirá la posibilidad de que vaya acompañado de un puntillo porque podría darse el caso de que para completar un compás hicieran faltan silencios de fusa o fusas y ninguno de los símbolos musicales anteriores han decidido codificarse en este generador porque había que marcar un límite en la codificación y este ha sido el elegido.

3.3.5. Función para determinar la altura según la clave

Esta función actúa como un plantilla que indica las alturas que podrán codificarse según la clave seleccionada. Puesto que se ha tratado como una plantilla, habrá tres funciones/plantillas de este estilo, una para cada una de las tres claves que podemos codificar en el generador.

Se ha querido limitar las alturas codificables según la clave con el fin de lograr una partitura con la mayor coherencia musical posible, ya que al fin y al cabo, en las partituras, según la clave, hay un rango de alturas más común que otro. Se ha optado por un rango de 18 alturas diferentes para cada una de las claves.

Se puede observar el rango de alturas determinado para cada una de las claves en las siguientes figuras.

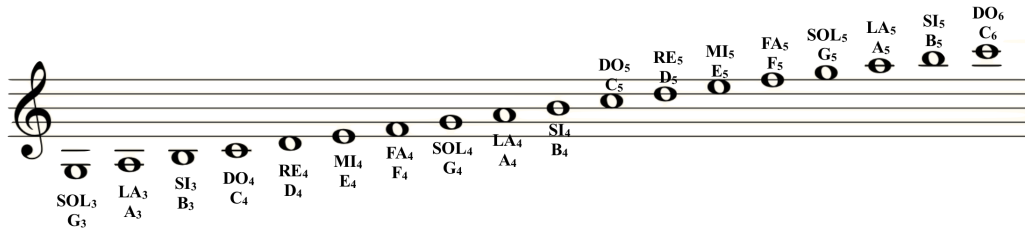


Figura 3.5: Rango de alturas codificables para la clave de sol.

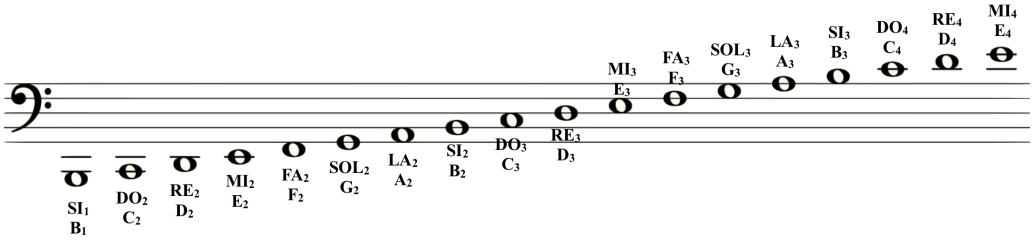


Figura 3.6: Rango de alturas codificables para la clave de fa en cuarta.

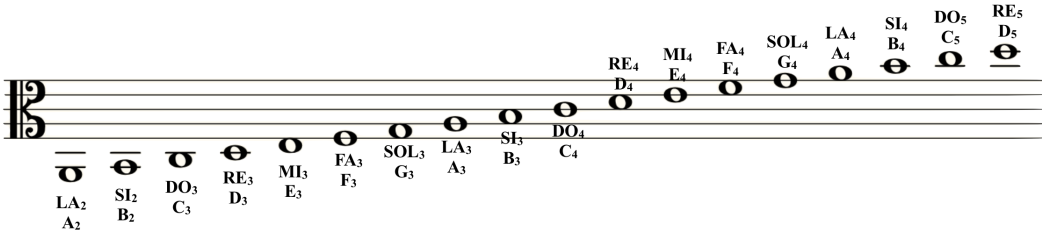


Figura 3.7: Rango de alturas codificables para la clave de do en tercera.

En definitiva, la función para determinar la altura según la clave recibe cuatro parámetros de entrada:

- Un entero sobre el cual se basará la estructura de control.
- Una secuencia de caracteres en la cual se guardará la estructura de la gramática agnóstica asociada a la altura elegida.
- Un entero donde se guardará el valor asociado a cada altura (tabla 3.1).
- Una secuencia de caracteres en la cual se guardará la estructura de la gramática agnóstica asociada a la posición del puntillo de la nota para aquellos casos que cuenten con dicha figura.

La estructura de control está basada en el entero nombrado antes y según el valor que este tenga, puede ser cualquier número entre 0 y 17 (18 alturas), se ejecutará un caso asociado a una de esas 18 alturas posibles. En dicho caso, se guardará en una secuencia

de caracteres la codificación en *kern* de la altura elegida. Esta secuencia es el parámetro de retorno de la función. Además, se dotarán las dos secuencias de caracteres de entrada de las estructuras correspondientes de la gramática agnóstica de la altura elegida y de la posición del puntillo, respectivamente, y se guardará en el entero el valor asociado a la altura elegida.

3.3.6. Función para determinar la altura de una nota

Para determinar la altura de una nota, se han propuesto tres métodos distintos con el fin de obtener composiciones con diferente aspecto musical. El usuario decide el método que se usará para determinar la altura de una nota, el cual será usado durante toda la generación de la partitura. Los posibles métodos son:

1. Generación aleatoria según la distribución normal La distribución normal, distribución de Gauss, distribución gaussiana o distribución de Laplace-Gauss, es una distribución de probabilidad de variable continua. Es de gran aplicación en los campos de ingeniería, física y ciencias sociales debido a que permite modelar numerosos fenómenos naturales, sociales y psicológicos.

La gráfica de su función de densidad tiene una forma de campana y es simétrica respecto a la media (figura 3.8). Esta curva se conoce como campana de Gauss y es el gráfico de una función gaussiana.

Se creará una distribución gaussiana con las 18 posibles alturas del rango determinado según la clave.

2. Random walk El paseo aleatorio, conocido como *random walk*, es una formalización matemática de la trayectoria que resulta de hacer sucesivos pasos aleatorios.

En este trabajo, el paseo aleatorio partirá siempre desde la altura central del rango de alturas determinado para cada clave. Se podrán dar tres posibles pasos aleatorios (todos igual de probables):

1. Un paso hacia adelante. Es decir, se sumará uno a la altura de la nota anterior, avanzando a la siguiente altura del rango, la cual será la altura de la nota actual.

2. Un paso hacia atrás. Es decir, se restará uno a la altura de la nota anterior, retrocediendo a la anterior altura del rango, la cual será la altura de la nota actual.

3. No se dará ningún paso. Es decir, la altura actual será igual a la altura de la nota anterior.

Aparecerán situaciones en las que se desee avanzar o retroceder más de lo permitido, es decir, la altura a decidir quede fuera del rango marcado. En estas situaciones, se establecerán dos soluciones:

- Límite reflexivo: como su propio nombre indica, funciona como un espejo haciendo que el paso a dar sea el reflejo del que inicialmente se quería dar. Si se quería avanzar, se retrocederá, y viceversa. Es decir, la altura de la nota actual será la segunda del rango empezando o bien por el límite superior o bien por el inferior, según corresponda.

- Límite absorbente: la altura de la nota actual será la correspondiente al límite superior o inferior, según corresponda, del rango de alturas.

La solución a tomar se elegirá al azar, siendo ambas igual de probables.

3. Sonificación de la ecuación logística La ecuación logística está definida por la ecuación 3.1.

$$x_{n+1} = r\ x_n(1 - x_n) \quad donde\ n = 0, 1, 2, 3, ... \tag{3.1}$$

Esta ecuación define una iteración, donde x_0 es igual a 0 y el parámetro r es un valor entre 0 y 4. El valor resultante siempre estará contenido en [0,1].

Si el usuario ha decidido este método, se le pedirá que introduzca un valor para el parámetro r entre 3,5 y 4, pues es el rango de valores para el cuál se generan las secuencias de notas más interesantes. Las secuencias para $3 \leq r \leq 3,5$ producen alternancias entre 2 o 4 alturas periódicamente, con poca variabilidad. Valores para $r < 3$ generan sencuencias de alturas constantes (unísonas) después de un corto periodo de transición.

El primer y último método generan composiciones musicales "estridentes", en las cuales es posible encontrar grandes saltos entre notas. Mientras que el segundo método, produce composiciones más armónicas, donde el salto de una nota a otra nunca va a ser mayor a cuatro semitonos.

A continuación, se describen cada una de las funciones implementadas para los distintos métodos.

Función para determinar la altura de una nota según la distribución normal

Esta función recibe los siguientes parámetros de control para la generación de notas:

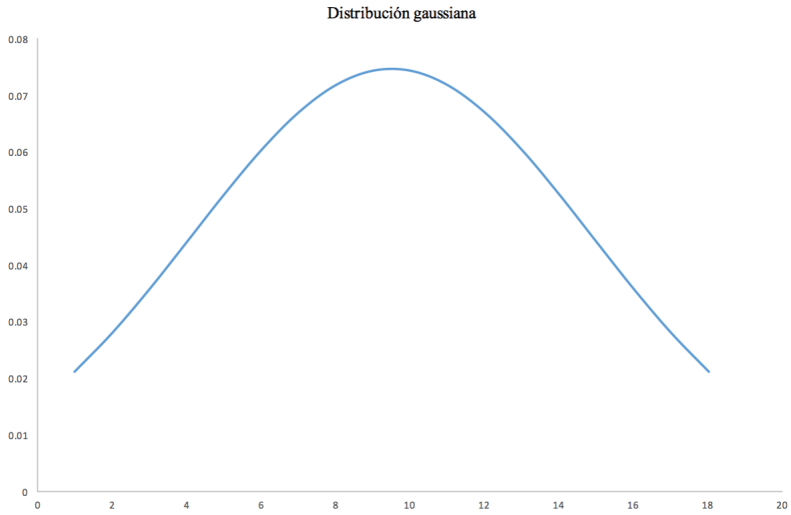
- Un entero que contiene el identificador de la clave elegida.
- Dos flotantes, uno indica el valor de la media y otro, el valor de la desviación típica de la distribución normal.
- Una secuencia de caracteres necesaria para llamar a la función 3.3.5 de determinar la altura según la clave, en la cual se guardará la estructura de la gramática agnóstica asociada a la altura elegida.
- Un entero necesario para llamar a la función 3.3.5 de determinar la altura según la clave, donde se guardará el valor asociado a cada altura (figura 3.1).
- Una secuencia de caracteres necesaria para llamar a la función 3.3.5 de determinar la altura según la clave, en la cual se guardará la estructura de la gramática agnóstica asociada a la posición del puntillo de la nota para aquellos casos que cuenten con dicha figura.
- Un entero cuyo valor será determinado mediante la distribución normal y que determinará el caso de la función 3.3.5 de determinar la altura según la clave que se dará.
- Dos enteros que contienen información acerca de las corcheas o semicorcheas ligadas.
- Dos enteros que contienen información acerca de los acordes.

C++ cuenta con una plantilla de clase para la distribución normal. Esta plantilla crea una distribución normal de números alrededor de la media de distribución ($\bar{x}$) con una desviación típica específica (σ). Como bien se ha especificado antes, el rango de enteros que identifica cada una de las alturas posibles según la clave es de 18 enteros (de 0 a 17). Por lo que el valor de la media, $\bar{x}$, según la ecuación 3.2, es 8,5; y el valor de la desviación típica, σ, según la ecuación 3.3, es 5,34.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.2}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.3}$$

Por tanto, se hace uso de la plantilla de clase de la distribución normal para crear una con los valores de media y desviación típica calculados. Una vez creada la distribución normal, se obtiene aleatoriamente un valor perteneciente a esta. El valor obtenido es un flotante, por lo que se le aproxima a su entero más cercano.



con $\bar{x} = 8,5$ y $\sigma = 5,34$

Figura 3.8: Distribución gaussiana para el rango de 18 alturas.

Entonces, mediante una estructura de control basada en el entero de entrada que contiene el identificador de la clave elegida, se llamará a la función 3.3.5 de determinar la altura según dicha clave. Esta función devolverá una secuencia de caracteres que contienen la codificación en ***kern* de la altura elegida. Esta secuencia es también el parámetro de retorno de la función.

Cuando las alturas a codificar pertenezcan a grupos musicales como corcheas o semicorcheas unidas, tresillos o acordes, estas serán limitadas. De manera que, para los casos de corcheas o semicorcheas unidas y tresillos las alturas de las notas involucradas distarán a lo sumo cuatro semitonos entre ellas. Este límite se ha establecido para evitar que dichos grupos musicales cuenten con alturas de más de una octava de diferencia entre ellas, pues no es lo común en notación musical. Los acordes a codificar serán acordes de triada mayor, menor, aumentada o disminuida. Por tanto, la limitación que se establece es que la primera altura permita codificar su correspondiente acorde, es decir, que el acorde a codificar esté dentro del rango de posibles alturas. Si el acorde a codificar queda fuera del rango, se codificará ese mismo acorde pero de una octava superior.

Función para determinar la altura de una nota según random walk

Esta función recibe los mismos parámetros de entrada que la función que utiliza el método de la distribución normal, a excepción de los flotantes que contienen los valores de la media y la desviación típica de la distribución normal y los dos enteros que contienen información acerca de las corcheas o semicorcheas ligadas. Estos dos enteros no serán necesarios en esta función pues, tal y como se ha explicado, el mayor salto será de una altura a la siguiente o a la anterior a ella en el rango de alturas.

Contará además con otro parámetro de entrada el cual será un entero que sirve para identificar si la altura a calcular es la primera de la partitura. Si esto es así, la primera altura será la central del rango de alturas.

La estructuración de esta función será igual que la de determinar la altura mediante la distribución normal, cambiando la parte de implementación de la distribución normal por la de *random walk*.

Función para determinar la altura de una nota según la ecuación logística

Esta función recibe los mismos parámetros que la función para determinar la altura mediante la distribución normal, a excepción de los flotantes que contienen los valores de la media y la desviación típica de la distribución normal. Además de esos parámetros cuenta con otros tres:

- Dos flotantes necesarios para la implementación de la ecuación logísitica.
- Un entero que identifica si la altura a calcular es la primera de la partitura. Si esto es así, la primera altura será la central del rango de alturas.

La estructuración de esta función es también la misma que la de determinar la altura mediante la distribución normal, cambiando la parte de implementación de la distribución normal por la de la ecuación logística.

3.3.7. Función para determinar la alteración de una nota

Se le ha pedido al usuario que introduzca un porcentaje entre 0 a 100 que indica la probabilidad de que una nota esté alterada. Si el porcentaje introducido es 0, la tonalidad será DO Mayor/LA Menor y no se alterará ninguna de las notas que aparezcan en la partitura. En cambio, si el porcentaje introducido difiere de 0, la tonalidad podrá ser

DO Mayor/LA Menor o cualquiera de las otras posibles del círculo de quintas, quedando guardadas las alturas modificadas, en caso de ello, en un vector tal y como se ha explicado en la sección 3.3.2. Por tanto, lo primero que se hará dentro de esta función será recorrer dicho vector para saber si la altura elegida está alterada o no por la armadura de tonalidad. En caso de que se encuentre alterada, se verá el porcentaje introducido por el usuario para decidir si se mantiene la alteración original (sostenido o bemol) dada por la tonalidad o se hace uso del becuadro para que la altura de la nota sea su altura natural.

Si el usuario ha introducido un porcentaje de alteración distinto a cero pero la tonalidad elegida es DO Mayor/LA Menor, en caso de darse el porcentaje de alteración para esa nota, la alteración será elegida aleatoriamente entre sostenido y bemol, ambas con la misma probabilidad de elección.

Cuando una nota sea alterada, su codificación en *kern* será añadida a la secuencia de caracteres que recogen toda la información de codificación de esa nota. Dicha secuencia de caracteres será unos de los parámetros de entrada junto con el porcentaje de alteración introducido por el usuario, el entero identificador de la tonalidad, el vector de enteros que identifica qué alturas han sido alteradas por la tonalidad, el entero identificador de la altura natural de la nota y la secuencia de caracteres con la estructura de la gramática agnóstica asociada a la altura. Además, cada vez que se altere una nota, se imprimirá por pantalla la gramática agnóstica asociada a la alteración, solo si esta se muestra explícitamente en la partitura.

3.3.8. Función para ligar notas

Kern diferencia entre ligaduras (*ties*) y ligaduras de expresión (*slurs*), pero como en la gramática agnóstica usada (sección 3.4.1) no se hace diferencia entre ambos, se ha optado por codificar las ligaduras (*ties*) como *slurs*.

Existe únicamente un 10 % de probabilidad de ligadura, por lo que esta función solo se usará cuando se de esa probabilidad.

Esta función es básicamente una estructura de decisión con tres posibles opciones: inicio de ligadura, continuación de ligadura o fin de ligadura. Si se da la opción de inicio de ligadura, la codificación en *kern* de inicio de *slur* es añadida a la secuencia de caracteres que recogen toda la información de codificación de esa nota. Dicha secuencia de caracteres será uno de los parámetros de entrada junto con un entero que indica que las notas siguientes van a ser ligadas, un entero que indica si se trata del inicio o del final de una ligadura y la secuencia de caracteres con la estructura de la gramática agnóstica asociada a la altura. Dentro de la opción de inicio de ligadura, habrá un 80 % de probabilidad de que esa ligadura una únicamente a dos notas (a la actual y su siguiente)

y un 20 %, de que una mínimo a tres. Cuando se da este último porcentaje se salta a la opción de continuación de ligadura.

La opción de continuación de ligadura se da en ese 20 %, que, como se ha indicado, es cuando la ligadura une mínimo tres notas. Dentro de esta opción, existe un 80 % de probabilidad de que no una más de tres notas y un 20 %, de que siga uniendo notas. Si se da la primera probabilidad, se saltará a la opción de fin de ligadura, sino se seguirá en la opción de continuación de ligadura hasta que se de la opción de fin de ligadura. Cuando se da esta última opción, la codificación en *kern* de fin de *slur* es añadida a la secuencia de caracteres que recogen toda la información de codificación de esa nota y se vuelve a restablecer la probabilidad de que una nota esté ligada a la siguiente o no.

Cada vez que se añade la codificación en *kern* de inicio o fin de ligadura, se imprime por pantalla la estructura de la gramática agnóstica de la ligadura.

3.3.9. Función para determinar una nota

La función planteada para codificar los distintos tipos de nota recibe los siguientes parámetros:

- Un flotante que indica cuántos tiempos se han completado de la duración total del compás.
- Un entero que identifica el tipo de compás que ha sido seleccionado.
- Un entero que identifica la clave que ha sido seleccionada.
- Un entero que contiene el porcentaje introducido por el usuario de que una nota esté alterada.
- Un entero que indica el método de cálculo de altura elegido.
- Dos flotantes con la media y la desviación típica de la distribución normal.
- Dos flotantes que contienen información para el cálculo de la altura mediante la ecuación logística.
- Un entero que contiene información de la altura anterior.
- Un entero que identifica la tonalidad elegida.

- Un vector de enteros que identifica qué alturas han sido alteradas por la tonalidad.
- Dos enteros que contienen información acerca de la ligadura de notas.
- Dos enteros que contienen información acerca de corcheas o semicorcheas unidas.
- Dos enteros que contienen información acerca de los acordes.
- Un entero que indica si la nota a calcular es la primera de la partitura.

Dentro de la función se crean los siguientes parámetros:

- Una secuencia de caracteres en la cual se guardará la codificación en ***kern* de la duración de la nota.
- Una secuencia de caracteres en la cual se guardará la codificación en ***kern* de la altura de la nota.
- Una secuencia de caracteres en la cual se guardará la codificación completa en ***kern* de la nota. Esta secuencia es el parámetro de retorno de la función.
- Una secuencia de caracteres en la cual se guardará la estructura de la gramática agnóstica acerca de la altura de la nota.
- Una secuencia de caracteres en la cual se guardará la estructura de la gramática agnóstica acerca de la posición del puntillo, en caso de que la nota lleve puntillo.
- Un entero cuyo valor será elegido aleatoriamente, entre 0 y 7, y sobre el cual se basará la estructura de control que decidirá el tipo de nota. Cada uno de los posibles valores de este entero está asociado a un tipo de nota, figura 3.5.

Tabla 3.5: Entero identificador para cada tipo de nota.

Nota	Redonda	Blanca	Negra	Corchea
Entero asociado	0	1	2	3

Nota	Semicorchea	Dos corcheas unidas	Dos semicorcheas unidas	Cuatro semicorcheas unidas
Entero asociado	4	5	6	7

- Un entero cuyo valor será elegido aleatoriamente entre 0 y 19 y sobre el cual se basará la estructura de probabilidad que decidirá si la nota lleva puntillo o no.
- Un entero en el que se guardará el valor elegido de la distribución normal.

Para cada caso (tipo de nota) de la estructura de control, habrá dos posibilidades de codificación: nota con puntillo o nota sin puntillo. La primera de ellas solo tendrá un 5 % de probabilidad de ocurrir. La razón que ha llevado a elegir este valor de probabilidad para una nota con puntillo es la misma que la del silencio con puntillo, ambas estructuras de probabilidad están creadas de la misma manera. Al igual que en el caso del silencio con puntillo, en cualquier momento se puede modificar la probabilidad de una nota con puntillo si así se desea.

Tanto para el caso de la nota con puntillo como para el caso de la nota sin puntillo, se comprobará si dicha nota "cabe" en el compás. Si así es:

1. Se guarda la codificación en *kern* de la duración de la nota elegida en su secuencia de caracteres correspondiente.

2. Se obtiene la altura de la nota según el método elegido.

3. Se guarda la codificación en *kern* de la altura de la nota junto a la de su duración en la secuencia de caracteres que contiene la información completa de la codificación de la nota.

4. Se llama a la función 3.3.7 que determina la alteración de la nota.

5. Se comprueba si en la nota actual termina una ligadura y si es así, se llama a la función 3.3.8 asociada.

6. Se imprime por pantalla la estructura de la gramática agnóstica asociada a la nota determinada.

7. Se comprueba si se ha cumplido la probabilidad de ligadura y si es así, se llama a la función 3.3.8 asociada.

8. Se le suma al flotante de entrada que indicaba los tiempos completados de la duración total del compás el valor en tiempos de negra de la nota elegida.

Cuando el tipo de nota a codificar es una corchea o semicorchea unida, se repetirán los pasos del 2 al 6, excluyendo el paso 5, tanta veces como número de corcheas o semicorcheas por unir queden. Para la primera y última corchea o semicorchea unida, se debe añadir la codificación en *kern* de inicio y fin de *beam*, respectivamente, a la secuencia de caracteres que contiene el resto de codificación de la nota.

En el caso de que el tipo de nota elegido sea de semicorchea, no existirá la posibilidad de que vaya acompañada de un puntillo porque podría darse el caso de que para completar un compás hicieran faltan silencios de fusa o fusas y ninguno de los símbolos musicales

anteriores han decidido codificarse en este generador porque había que marcar un límite en la codificación y este ha sido el elegido. Además, las corcheas o semicorcheas unidas tampoco contarán con la posibilidad de ir acompañadas de puntillo, esto se debe a que se ha determinado que las corcheas o semicorcheas solo podrán codificarse si además de caber en el compás se encuentran en el inicio de tiempo de este.

3.3.10. Función para determinar tresillos

Esta función cuenta con los mismos parámetros que la anterior, siendo la asociación de un entero identificador a un tipo de tresillo la siguiente:

Tabla 3.6: Entero identificador para cada tipo de tresillo.

Tresillo	De blanca	De negra	De corchea	De semicorchea
Entero asociado	0	1	2	3

No solo cuenta con los mismos parámetros que la función anterior sino que tiene la misma estructura y manera de proceder, siendo la única diferencia que para cada tipo de tresillo se calcularán tres alturas diferentes. Por tanto, los pasos a seguir son los mismos que los explicados en la función anterior, usando la codificación en *kern* y la estructura de la gramática agnóstica correspondientes para cada tipo de tresillo.

Recalcar que debido a su composición, los tresillos no van a llevar puntillos ni estarán ligados. Únicamente podrán formar parte de una ligadura de expresión (*slur*).

3.3.11. Función para determinar acordes

Se van a codificar únicamente acordes de tres notas, pues es el límite personal de codificación que se ha establecido.

Esta función cuenta con los mismos parámetros y estructura de control que la función 3.3.9 de determinar una nota. Funcionarán igual siendo la única diferencia entre ambas que para cada caso de la estructura de control, en la función de determinar acordes, se calculan tres alturas diferentes.

Tabla 3.7: Entero identificador para cada tipo de acorde.

Acordes	De redonda	De blanca	De negra	De corchea	De semicorchea
Entero asociado	0	1	2	3	4

3.4. Funcionalidad

La transcripción de la notación musical generada por el sistema planteado en el presente proyecto es lo que dota de funcionalidad y valor al sistema. Dicha transcripción junto a la imagen que representa gráficamente la información musical, forman el par necesario para el algoritmo de aprendizaje automático.

La transcripción permite almacenar el contenido musical de una partitura de manera gramatical mediante una serie de símbolos y reglas que permiten que en caso de perder la representación gráfica de la partitura se pueda volver a generar gracias a la transcripción de esta.

La gramática de transcripción usada en este trabajo es la creada para el proyecto **HispaMus**.[2] Este proyecto está dedicado a la conservación del patrimonio musical español mediante la transcripción de este, logrando de esa manera almacenar toda la información musical de una partitura en estructuras gramaticales. Se usan una serie de algoritmos de aprendizaje automático que reconocen los símbolos musicales presentes en las partituras y generan su transcripción correspondiente.

A la gramática de HispaMus se le denomina gramática agnóstica porque no hace suposiciones sobre el significado musical de lo que se representa en el documento que se analiza, es decir, los elementos se identifican en un catálogo de símbolos musicales por la forma que tienen y donde se sitúan en la partitura. Su significado musical queda para una fase posterior de análisis, en el que se tienen en cuenta las relaciones entre ellos y conocimiento a priori sobre el funcionamiento de la notación musical. Estos aspectos *musicológicos* de la representación no son objeto del presente proyecto.

Los principios que sigue este tipo de representación son los siguientes [Rizo et al., 2017]:

- Dos símbolos musicales que son gráficamente iguales se representan mediante el mismo símbolo de la gramática agnóstica.
- Los elementos identificados en una imagen deben ser representados de acuerdo a su posición relativa en esta, de manera que se pueda conocer dónde está posicionado un símbolo en relación al resto.
- La gramática debe ser determinista y no ambigua, permitiendo analizar un documento dado de una única manera.

[2] Proyecto Hispamus: *Handwritten Spanish Music Heritage Preservation by Automatic Transcription*, desarrollado por José Manuel Iñesta Quereda, Jorge Calvo Zaragoza, David Rizo Valero, María Luisa Micó Andrés, José Oncina Carratalá, Juan Ramón Rico Juan, Pedro José Ponce de León Amador, Antonio Jorge Pertusa Ibáñez y Carlos Pérez Sancho.

- No se adjuntará ningún atributo semántico a los elementos identificados. Es decir, la partitura podría ser perfectamente reescrita a partir de la transcripción por alguien que no sepa nada acerca del significado musical de los símbolos representados. Por ejemplo, no hay diferencia entre un *puntillo* y un *staccato*.

3.4.1. Gramática agnóstica

El esquema general que se sigue en esta gramática para representar un símbolo musical es el siguiente:

tipo_de_símbolo.especificación:posición_en_el_pentagrama

Donde:

1. **tipo_de_símbolo** puede ser:
   - **clef** para identificar una clave.
   - **digit** para identificar un número.
   - **accidental** para identificar una alteración.
   - **note** para identificar una nota.
   - **rest** para identificar un silencio.
   - **bracket** para identificar una barra.
   - **verticalLine** para identificar una barra de compás
   - **dot** para identificar un puntillo.
   - **slur** para identificar una ligadura.
2. **especificación** puede ser:
   - **G, C o F** para especificar el tipo de clave (sol, do o fa).
   - **1, 2, 3, 4...** para especificar de que número se trata, en el caso de *digit*.
   - **sharp, flat o natural** para especificar el tipo de alteración (sostenido, bemol o becuadro).

- **whole, half, quarter, eighth, sixteenth, beamedRight1, beamedBoth1, beamedLeft1, beamedRight2, beamedBoth2 o beamedLeft2** para especificar el tipo de nota (redonda, blanca, negra, corchea, semicorcheas, corcheas unidas y semicorcheas unidas).

- **whole, half, quarter, eighth o sixteenth** para especificar el tipo de silencio.

- **start o end** para especificar el inicio o el final de una barra o de una ligadura.

3. **posición_en_el_pentagrama** puede ser **S1, L2, S2**... en función de la línea o el espacio del pentagrama, de abajo a arriba, en el que se encuentre el símbolo. Véase figura 3.9.

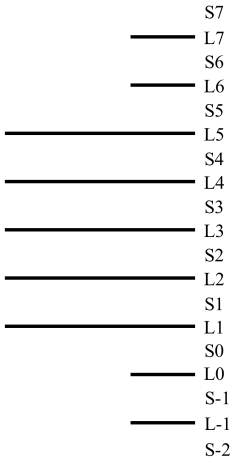


Figura 3.9: Criterio asignado a la posición del símbolo musical en el pentagrama.

Si el símbolo identificado es una barra de compás o un puntillo, la estructura gramatical no contará con la parte de especificación de manera que la transcripción será:

tipo_de_símbolo:posición_en_el_pentagrama

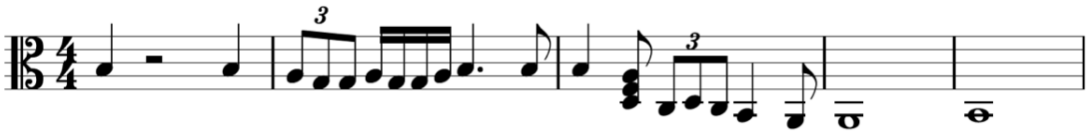
La estructura correspondiente a distintos símbolos musicales es separada por una coma seguida de un espacio en blanco, de manera que:

primer_símbolo, segundo_símbolo

Cuando los símbolos identificados son notas pertenecientes a un acorde, la estructura de cada una de las notas irá separada de la siguiente manera:

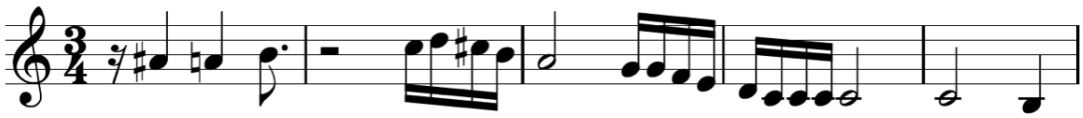
primera_nota/segunda_nota/tercera_nota

A continuación, se muestran una serie de ejemplos compuestos por una breve pieza musical y su transcripción, en los cuales se pueden observar todas las estructuras gramaticales descritas en esta sección. Los ejemplos han sido obtenidos gracias al sistema de generación automático de datos etiquetados, por lo que podemos afirmar que el sistema ha sido desarrollado correctamente.



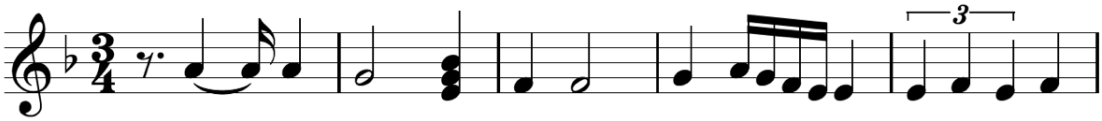
clef.C:L3, digit.4:L4/digit.4:L2, note.quarter:S2, rest.half:L3, note.quarter:S2, verticalLine:L1, note.eighth:L2, digit3:S6, note.eighth:S1, note.eighth:S1, note.beamedRight2:L2, note.beamedBoth2:S1, note.beamedBoth2:S1, note.beamedLeft2:L2, note.quarter:S2, dot:S2, note.eighth:S2, verticalLine:L1, note.quarter:S2, note.eighth:L2/note.eighth:L1/note.eighth:L0, note.eighth:S-1, digit3:S6, note.eighth:L0, note.eighth:S-1, note.quarter:L-1, note.eighth:S-2, verticalLine:L1, note.whole:S-2, verticalLine:L1, note.whole:L-1, verticalLine:L1

Figura 3.10: Ejemplo de transcripción número 1.



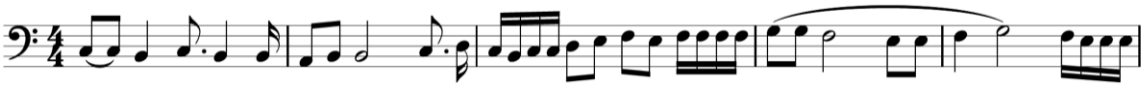
clef.G:L2, digit.3:L4/digit.4:L2, rest.sixteenth:L3, accidental.sharp:S2, note.quarter:S2, note.quarter:S2, note.eighth:L3, dot:S3, verticalLine:L1, rest.half:L3, note.beamedRight2:S3, note.beamedBoth2:L4, accidental.sharp:S3, note.beamedBoth2:S3, note.beamedLeft2:L3, verticalLine:L1, note.half:S2, note.beamedRight2:L2, note.beamedBoth2:L2, note.beamedBoth2:S1, note.beamedLeft2:L1, verticalLine:L1, note.beamedRight2:S0, note.beamedBoth2:L0, note.beamedBoth2:L0, note.beamedLeft2:L0, note.half:L0, verticalLine:L1, note.half:L0, note.quarter:S-1, verticalLine:L1

Figura 3.11: Ejemplo de transcripción número 2.



clef.G:L2, accidental.flat:L3, digit.3:L4/digit.4:L2, rest.eighth:L3, dot:S3, note.quarter:S2, slur.start:S2, note.sixteenth:S2, slur.end:S2, note.quarter:S2, verticalLine:L1, note.half:L2, note.quarter:L1/note.quarter:L2/note.quarter:L3, verticalLine:L1, note.quarter:S1, note.half:S1, verticalLine:L1, note.quarter:L2, note.beamedRight2:S2, note.beamedBoth2:L2, note.beamedBoth2:S1, note.beamedLeft2:L1, note.quarter:L1, verticalLine:L1, bracket.start-S6/note.quarter:L1, digit.3-S6/note.quarter:S1, bracket.end-S6/note.quarter:L1, note.quarter:S1, verticalLine:L1

Figura 3.12: Ejemplo de transcripción número 3.



clef.F:L4, digit.4:L4/digit.4:L2, note.beamedRight1:S2, note.beamedLeft1:S2, slur.start:S2, slur.end:S2, note.quarter:L2, note.eighth:S2, dot:S2, note.quarter:L2, note.sixteenth:L2, verticalLine:L1, note.beamedRight1:S1, note.beamedLeft1:L2, note.half:L2, note.eighth:S2, dot:S2, note.sixteenth:L3, verticalLine:L1, note.beamedRight2:S2, note.beamedBoth2:L2, note.beamedBoth2:S2, note.beamedLeft2:S2, note.beamedRight1:L3, note.beamedLeft1:S3, note.beamedRight1:L4, note.beamedLeft1:S3, note.beamedRight2:L4, note.beamedBoth2:L4, note.beamedBoth2:L4, note.beamedLeft2:L4, verticalLine:L1, slur.start:S4, note.beamedRight1:S4, note.beamedLeft1:S4, note.half:L4, note.beamedRight1:S3, note.beamedLeft1:S3, verticalLine:L1, note.quarter:L4, slur.end:S4, note.half:S4, note.beamedRight2:L4, note.beamedBoth2:S3, note.beamedBoth2:S3, note.beamedLeft2:S3, verticalLine:L1

Figura 3.13: Ejemplo de transcripción número 4.

4 Evaluación del sistema

El objetivo principal de este proyecto es contribuir a la investigación OMR mediante la creación de un corpus de referencia, el cual muestra una gran cantidad de partituras musicales junto con su correspondiente transcripción. Para la creación de dicho corpus se ha desarrollado el sistema de generación automática de datos etiquetados, con el fin de poder obtener un conjunto potencialmente ilimitado de ejemplos.

Para saber si se ha logrado alcanzar el objetivo principal de este trabajo, se ha llevado a cabo un experimento de reconocimiento automático.

4.1. Datos experimentales

Se utilizará el sistema desarrollado en el presente proyecto para generar un corpus de 100.000 partituras consistentes en una única línea de pentagrama cada una, con diferentes valores de métrica, tonalidad, clave y distintos algoritmos para la generación de alturas.

Este corpus se utilizará como conjunto de entrenamiento de un algoritmo de reconocimiento de símbolos musicales desarrollado previamente [Calvo-Zaragoza and Rizo, 2018], basado en una red neuronal convolucional. Este tipo de algoritmos necesitan numerosas muestras de entrenamiento. En este caso, cada muestra será un par compuesto por una imagen (el pentagrama generado) y su correspondiente representación con el formato impuesto por la gramática agnóstica. Estos pares son como los que se muestran en las figuras de la 3.10 a la 3.13.

La distribución de las longitudes (en número de símbolos por muestra) de las secuencias utilizadas es:

- Promedio: 23,5
- Desviación: 5,3
- Máximo: 40
- Mínimo: 12

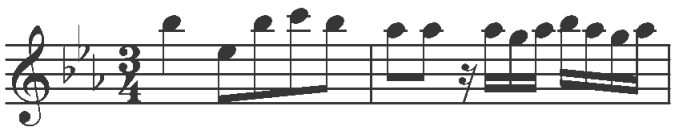


Figura 4.1: Ejemplo número 1 de muestra a reconocer en el experimento.

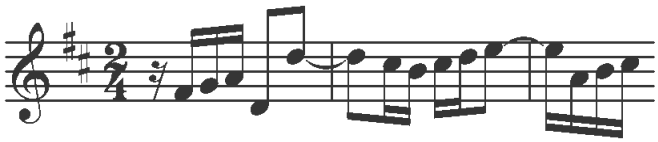


Figura 4.2: Ejemplo número 2 de muestra a reconocer en el experimento.

4.2. Experimento

Se trata de transcribir el contenido musical de 100 pentagramas, introducidos en el sistema como imágenes independientes (conjunto de test). El sistema nos devolverá en cada caso el contenido estructurado según la gramática agnóstica. Esta salida se comparará con la generada por el modelo del presente proyecto.

Para evaluar la bondad del sistema, se estudiará el promedio de errores de edición al comparar cada salida con la esperada (generada por nuestro modelo). Cuando se habla de error de edición, se refiere a que, para corregir la salida hay que añadir, quitar o sustituir uno o más símbolos hasta que coincida con la salida esperada. Se representará dicho promedio de dos formas, equivalentes entre sí:

- **sin normalizar**: correspondiente al número de operaciones necesarias para realizar la edición, y
- **normalizado**: correspondiente a dividir el número de operaciones entre la longitud de la partitura (lo que proporciona un valor en porcentaje entre 0 y 100).

Esta normalización permite afirmar que es lo mismo tener un error en una secuencia de 5 símbolos que dos errores en una secuencia de 10.

El promedio de errores de edición normalizado es equivalente al *Word Error Rate* (WER) en reconocimiento de texto o habla. Esta medida de evaluación calcula el número mínimo de correcciones (inserciones, borrados y sustituciones de una palabra por otra) necesarios para equiparar dos frases. Por tanto, estableciendo una referencia con el experimento realizado, se calcula el WER entre la partitura a reconocer y la partitura de referencia generada por el sistema desarrollado.

A continuación, se muestra la gráfica 4.3 en la cual han sido representados los resultados proporcionados por el experimento. En dicha gráfica, se compara el promedio errores de edición (sin normalizar y normalizado) que hay por partitura frente al número de muestras generadas para entrenar el modelo.

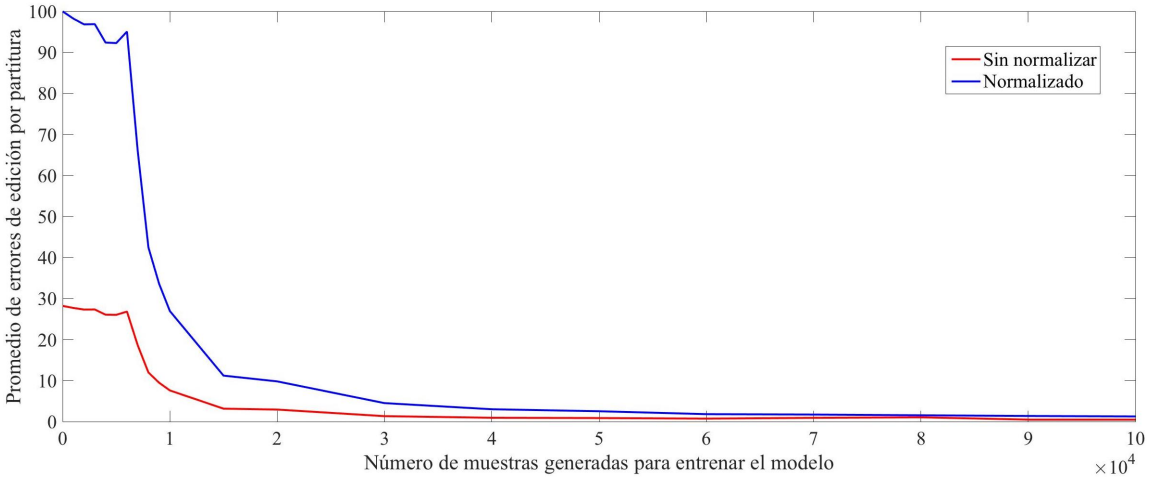


Figura 4.3: Gráfica que muestra los resultados obtenidos en el experimento. Las unidades del eje vertical se encuentran en porcentaje. Sin embargo, la curva roja mide el número absoluto de errores cometidos.

En ella se observa que cuando el modelo ha sido entrenado con una única muestra, el promedio de errores producidos en el conjunto de test es del 100 %. En una primera fase del entrenamiento, con un número limitado de muestras de aprendizaje (hasta 6.000 muestras), el sistema no es capaz de disminuir significativamente el error ($\geq 92\,\%$).

La primera gran disminución se encuentra entre las 6.000 y 7.000 muestras: cuando el modelo utiliza esta última cantidad de muestras de entrenamiento su promedio de error de edición baja bruscamente al 66 %. A partir de ahí, el promedio de errores de edición comienza un descenso gradual hasta estabilizarse en torno a un valor de un 1 %. Por tanto, se puede afirmar que, entorno a las 6.000 muestras, el sistema comienza a tener un modelo capaz de identificar las imágenes de los símbolos musicales contenidos en la partitura.

La estabilización del promedio de errores de edición en un valor tan bajo indica que el modelo ha logrado aprender correctamente, por lo que no resulta interesante seguir generando más muestras, ya que las mejoras esperadas serían muy pequeñas. De hecho, con la mitad de muestras de entrenamiento (50.000) el error relativo del sistema era sólo un 1,5 % superior al obtenido finalmente con las 100.000 muestras.

Normalmente cuando se entrenan estos modelos se suele establecer un número fijo de muestras de entrenamiento sobre las cuales se itera varias veces. En este caso, el

algoritmo de generación de muestras desarrollado genera nuevas muestras cada vez que se desee por lo que nunca se itera sobre la misma muestra sino que se van generando, pasando por el modelo para que aprenda, y luego se descartan. Se supone que, de esta forma, el modelo va a aprender mejor, aunque para demostrar que esta suposición es realmente correcta habría que realizar otro tipo de experimento.

5 Conclusiones

Como el lector habrá podido comprobar, se ha puesto un gran esfuerzo en desarrollar un sistema de generación automática de partituras de música monofónica eficaz, eficiente y realista. Hasta ahora no se había realizado ningún trabajo que generara de manera automática partituras musicales acompañadas de su transcripción, a pesar de su gran necesidad en el campo de investigación del Reconocimiento Óptico de Música. Gracias al sistema desarrollado, se ha podido observar su gran utilidad, al comprobar que permite generar una amplia librería, con infinidad de diferentes composiciones musicales, que sirve de base de datos de entrenamiento a los sistemas de redes neuronales de reconocimiento automático.

Por desgracia, el tiempo es limitado y en unos meses de trabajo es muy difícil llegar a obtener un generador automático que cree cualquier tipo partitura. Un posible trabajo futuro de este proyecto sería la implementación de música polifónica, así como el añadir otros símbolos musicales, como calderones, o incluso la posibilidad de poder agregar letras (*lyrics*) a la melodía. El sistema desarrollado cuenta con una gran carga de probabilidad, por lo que podría estudiarse si aquellos eventos dotados de probabilidad siguen una cierta regla musical e implementarlos según esta.

Son muchas las propuestas que dotarían de nuevas funcionalidades a este sistema. Sin embargo, el lector deberá comprender que el presente trabajo ha sido desarrollado estableciéndose desde un principio unos límites acordes al tiempo que se tenía para desarrollarlos. En un futuro, habría que rizar el rizo un poco más y proponer un método que fuera capaz de simular distintos tipos de escritura y niveles de degradación del papel a partir de un archivo de notación musical determinado. De esta manera, el archivo generado por el sistema actual podría convertirse en archivos de imagen de diferentes características. Para ello, se propone investigar el uso de Cycle Generative Adversarial Networks[1] con el fin de conseguir efectos que imiten el estilo manuscrito y así poder evaluar, de manera más exhaustiva, la utilidad de la construcción de un corpus de referencia para investigación en reconocimiento automático de partituras.

[1] https://github.com/junyanz/CycleGAN

Bibliografía

[Arbonés and Milrud, 2014] Arbonés, J. and Milrud, P. (2014). *La armonía es numérica*.

[Bainbridge, 1997] Bainbridge, D. (1997). *Extensible Optical Music Recognition*. PhD thesis, Department of Computer Science, University of Canterbury.

[Bainbridge and Bell, 2001] Bainbridge, D. and Bell, T. (2001). The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121.

[Bellini et al., 2001] Bellini, P., Bruno, I., and Nesi, P. (2001). Optical music sheet segmentation. pages 183 – 190.

[Bellini and Nesi, 2003] Bellini, P. and Nesi, P. (2003). *Modeling Music Notation in the Internet Multimedia Age*.

[Calvo-Zaragoza and Rizo, 2018] Calvo-Zaragoza, J. and Rizo, D. (2018). End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8(4):606–629.

[Cardoso and Rebelo, 2010] Cardoso, J. and Rebelo, A. (2010). Robust staffline thickness and distance estimation in binary and gray-level music scores. pages 1856–1859.

[Carter, 1992] Carter, N. P. (1992). Segmentation and preliminary recognition of madrigals notated in white mensural notation. *Mach. Vision Appl.*, 5(3):223–229.

[Coüasnon and Camillerapp, 1994] Coüasnon, B. and Camillerapp, J. (1994). Using grammars to segment and recognize music scores. *International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 15–27.

[Fujinaga, 1988] Fujinaga, I. (1988). Optical music recognition using projections.

[Fujinaga, 2004] Fujinaga, I. (2004). Staff detection and removal.

[Göcke, 2003] Göcke, R. (2003). Building a system for writer identification on handwritten music scores.

[Ng et al., 1999] Ng, K., Cooper, D., Stefani, E., Boyle, R., and Bailey, N. (1999). Embracing the composer: Optical recognition of handwritten manuscripts. pages 500–503.

[Niblack, 1985] Niblack, W. (1985). *An Introduction to Digital Image Processing*. Strandberg Publishing Company, Birkeroed, Denmark, Denmark.

[Novotný and Pokorný, 2015] Novotný, J. and Pokorný, J. (2015). Introduction to optical music recognition: Overview and practical challenges. 1343:65–76.

[Otsu, 1979] Otsu, N. (1979). A threshold selection method from gray-level histograms. 9:62–66.

[Pugin et al., 2014] Pugin, L., Zitellini, R., and Roland, P. (2014). Verovio: A library for engraving mei music notation into svg. *ISMIR*, page 107–112.

[Rebelo et al., 2010] Rebelo, A., Capela, G., and Cardoso, J. S. (2010). Optical recognition of music symbols. *International Journal on Document Analysis and Recognition (IJDAR)*, 13(1):19–31.

[Rebelo et al., 2012] Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marçal, A., Guedes, C., and Cardoso, J. (2012). Optical music recognition: State-of-the-art and open issues. 1:173–190.

[Rizo et al., 2017] Rizo, D., Calvo-Zaragoza, J., Iñesta, J. M., and Fujinaga, I. (2017). About agnostic representation of musical documents for optical music recognition. In *Music Encoding Conference, Tours, 2017*.

[Selfrigde-Field, 1997] Selfrigde-Field, E. (1997). *Beyond MIDI: The Handbook of Musical Codes*. The MIT Press.

[Sezgin and Sankur, 2004] Sezgin, M. and Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging 13(1)*, pages 146–168.