UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Estimación de la altura de la *x* en imágenes de páginas de texto manuscrito

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Pascual Andrés Carrasco Gómez

Tutor: Dr. Moisés Pastor i Gadea

Curso 2015 / 2016

RESUMEN

En la actualidad existen grandes volúmenes de imágenes que representan textos manuscritos digitalizados, estas imágenes han sido tomadas con dispositivos completamente diferentes a causa del avance tecnológico a lo largo de los años. Las resoluciones de dichas imágenes son diferentes provocando que los reconocedores de texto manuscrito presenten problemas para separar caracteres o palabras de la imagen. Conocer la altura de la *x* de cualquier texto manuscrito independientemente de la resolución que tenga la imagen nos permite corregir este problema. La altura de la *x* se define como la altura del carácter sin tener en cuenta los trazos ascendentes ni descendentes.

Inicialmente se realiza un estudio para obtener la altura de la *x* mediante la transformada de Fourier, en el estudio no encontramos una relación directa entre la resolución de la imagen y la altura de la *x* pero encontramos una solución para segmentar el texto en líneas, realizando una segmentación en líneas y aplicando una serie de algoritmos heurísticos obtenemos una estimación de la altura de la *x*. Medimos la calidad de esta estimación mediante el error que nos proporciona una experimentación sobre el corpus del *ICDAR 2013 handwriting segmentation contest.*

Este trabajo es pionero en este tema y es una base para mejoras y estudios posteriores que permitan mejorar la estimación obtenida incrementando las capacidades de los reconocedores de textos manuscritos.

Palabras clave: altura-X, DFT, segmentación, proyección, ICDAR, heurística.

ABSTRACT

Nowadays there exist large volumes of images that represent digital handwriten texts, these images have been taken with different devices that vary in specifications, this fact is due to technological development over the years. Those pictures have different resolutions, this causes manuscripted text recognizers to have issues in order to separate characters or words in the image. Knowing the height of the *X* in every manuscripted text independently of its image resolution allows us to correct this problem. The *X* height is defined as the character's height without having in account ascendent traces, neither descendent ones.

Initially, a study is performed in order to obtain the *X* height by using Fourier transform. In the study we don't find any direct relation between image resolution and the *X* height, but we find a solution to segmentate the text in lines; by performing a line by line segmentation and applying some series of heuristic algorithms we obtain one approximation of the *X* height. We measure the quality of this approximation by observing the error that comes provided in one experimentation about one corpus of the I*CDAR 2013 handwriting segmentation contest.*

This work is pioneer in this topic and sets up a basis to improvements and further studies that will allow the improvement of the estimation obtained and that will increment the capacities of manuscripted text recognizers.

Keywords: height-X, DFT, segmentation, projection, ICDAR, heuristics.

Índice

1. Introducción al problema

La digitalización de documentos permite que un mayor número de personas tengamos acceso a los mismos, que se reduzca su almacenamiento considerablemente, rapidez de consulta mediante buscadores indexados, una mejor organización y accesibilidad, tener replicaciones de los mismos, no dañar documentos antiguos al trabajar con ellos, evitar eventualidades como humedad y deterioro al conservarlos en papel, etc. Todas estas ventajas han propiciado que grandes volúmenes de documentos procedentes de todo el mundo sean digitalizados. Digitalizar un documento de texto consiste en obtener una imagen a partir del documento físico mediante un escáner, una cámara fotográfica, un móvil o cualquier dispositivo electrónico que disponga de una cámara o similar.

La tecnología avanza muy rápidamente reduciendo cada vez más los circuitos integrados e introduciendo tecnología muy avanzada a dispositivos cada vez más pequeños, si nos centramos en dispositivos con cámara fotográfica observamos que debido al crecimiento de mercado y la demanda, existen un gran número de sensores y ópticas que determinan que una imagen tomada desde dos dispositivos diferentes tengan diferentes características, estas características vienen dadas por componentes del dispositivo como son el rango focal, el zoom óptico, la calidad de la lente, la resolución de la imagen, etc.

Todos estos aspectos de la imagen causan problemas en los reconocedores ópticos de textos. Los reconocedores ópticos necesitan parámetros para su correcto funcionamiento, uno de estos parámetros es la dimensión de una ventana de análisis de la imagen para tratarla posteriormente, la dimensión de dicha ventana viene definida por lo que se conoce como la altura de la *x*, que corresponde a la altura del carácter de caja baja o minúsculas sin contar los trazos ascendentes y descendentes.

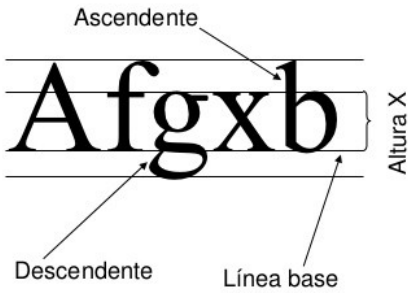


Figura 1.1: Definición gráfica de la altura de la x.

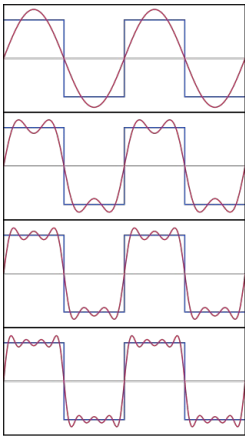
El objetivo de este trabajo es crear una herramienta informática que tome como entrada una imagen de un texto y devuelva como salida la altura de la *x*.
Cuando se trabaja con un corpus nuevo se requiere de un preproceso de configuración de los parámetros del reconocedor de texto para su correcto funcionamiento, la idea principal que buscamos es automatizar el parámetro de la ventana de análisis. Podemos automatizar el valor de dos formas:

- Si tenemos un corpus de imágenes de texto y somos capaces de obtener sus correspondientes alturas de la *x* podremos escalar todas las imágenes para que todas tengan la misma altura de la *x* configurando el valor de la ventana de análisis a dicho valor.
- Otra utilidad que podemos obtener al conocer la altura de la *x* de cualquier imagen de un texto es introducir la herramienta dentro del reconocedor para que independientemente de la resolución de las imágenes de entrada sea capaz de configurar automáticamente la dimensión de la ventana de análisis.

Este problema es un cuello de botella en los reconocedores ópticos provocando que los resultados obtenidos en el reconocimiento sean erróneos a causa de las variaciones de relación entre la altura de la *x* y las dimensiones de la imagen.

2. Solución propuesta

Nuestra idea principal consiste en obtener la altura de la *x* del texto mediante una herramienta matemática bien conocida como es la transformada de Fourier. La transformada de Fourier es una función matemática que transforma una señal en el dominio del tiempo al dominio de la frecuencia. En nuestro caso al trabajar con imágenes, la transformación es de un dominio espacial a un dominio frecuencial.
Estudiar como se calcula la transformada a nivel matemático no es el propósito de este trabajo y además existen muchas implementaciones que podemos obtener a través de librerías, aunque se hace necesario conocer la utilidad que nos proporciona y que coste computacional tiene.
De manera informal la transformada de Fourier se define como un sumatorio hasta el infinito de senos y cosenos, el número de senos y cosenos que sumemos harán que nos acerquemos más a la señal de entrada, siendo la señal de entrada la suma hasta el infinito de senos y cosenos.



A modo de ejemplo, en la figura 2.1 se puede ver que la señal de entrada corresponde a una señal cuadrada, representadas en cada cuadrante tenemos las primeras cuatro aproximaciones mediante la suma de senos y cosenos. Si nos fijamos en el último cuadrante, vemos que al tener más componentes en el sumatorio la señal sinusoidal se aproxima más a la señal de entrada.

Figura 2.1: Cuatro primeras aproximaciones de la transformada de Fourier de una señal cuadrada.

La figura 2.2 muestra de forma gráfica una proyección horizontal de una imagen que representa un texto.

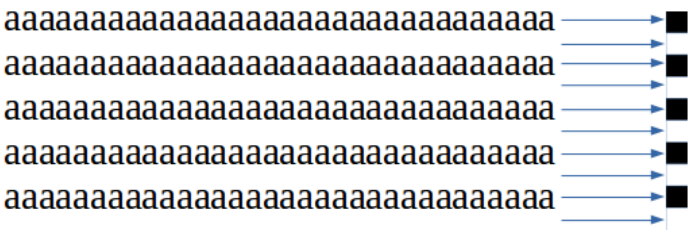


Figura 2.2: Ejemplo de proyección horizontal.

La transformación de la imagen en una señal unidimensional consiste en sumar para cada fila de la imagen todas sus columnas obteniendo un vector de proyección horizontal, dicho vector cabe destacar que no seria un vector negro y blanco como hemos indicado en la figura 2.2 ya que las casillas en negro serian variaciones en escala de grises, pero lo hemos planteado de esta forma para representar un caso ideal con la finalidad de entender mejor el resultado al aplicarle la transformada de Fourier a dicho vector.

El vector de proyección lo podemos visualizar con una gráfica, esta visualización nos facilita la comprensión sobre los elementos que forman el texto de la imagen de entrada, los elementos que buscamos al realizar esta proyección son las alturas de los caracteres y las alturas de separación entre las líneas de texto. En el ejemplo que hemos mostrado en la figura 2.2, la señal que representa la proyección del vector es una señal cuadrada, esta señal es un caso ideal, pero nos permite fácilmente acercar el concepto a su aplicación en nuestro trabajo.

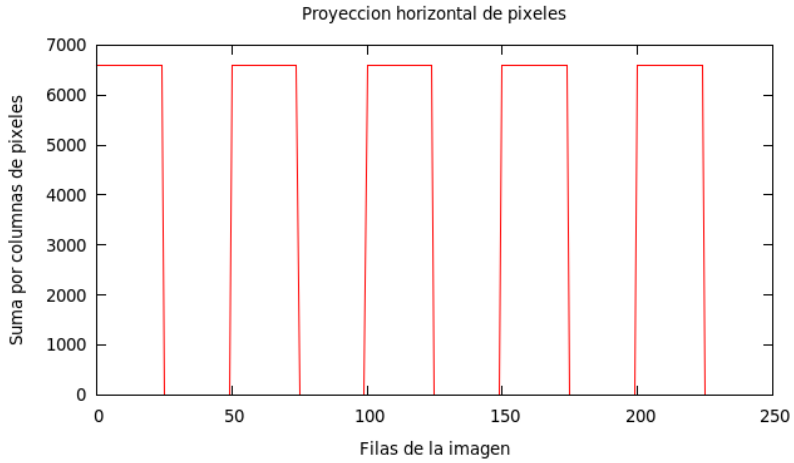


Figura 2.3: Proyección horizontal del vector negro-blanco.

La señal cuadrada que observamos en la figura 2.3 es la señal de entrada a la transformada de Fourier, en este punto tenemos que explicar que devuelve dicha transformada, como nosotros trabajamos con un dominio espacial la transformada no devuelve un conjunto de frecuencias en Hercios (Hz), que como ya sabemos representan ciclos/segundos, en nuestro caso tenemos frecuencias que representan ciclos/píxeles. Cada frecuencia tiene asociada una energía que representa la importancia de esa frecuencia en la representación de la señal, a esta gráfica se le denomina espectro de frecuencias. El espectro de frecuencias se define como la distribución de amplitudes para cada frecuencia que sea superposición de ondas de varias frecuencias, es decir, cada frecuencia que vaya apareciendo en el espectro de frecuencias es una superposición de ondas sobre una frecuencia base que representa la frecuencia fundamental de la señal, esta superposición de ondas corresponde al sumatorio de senos y cosenos que definen la transformada de Fourier.

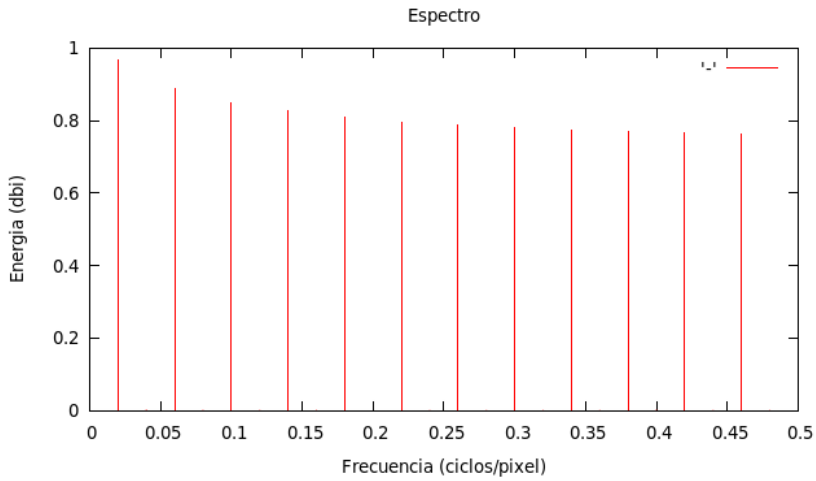


Figura 2.4: Espectro de frecuencias de vector negro-blanco.

La frecuencia fundamental corresponde a la frecuencia con mayor energía, en la figura 2.4 tiene un valor de 0.02 ciclos/píxel, este valor nos indica que cada 0.02 ciclos/píxel la señal de entrada se repite, a nosotros nos interesa el número de píxeles en vez de la frecuencia, por ello calculamos el periodo de la frecuencia que se define como su inversa. El periodo es 1/0.02 que es 50 píxeles.
A continuación mostramos como se visualiza el periodo de la frecuencia fundamental.

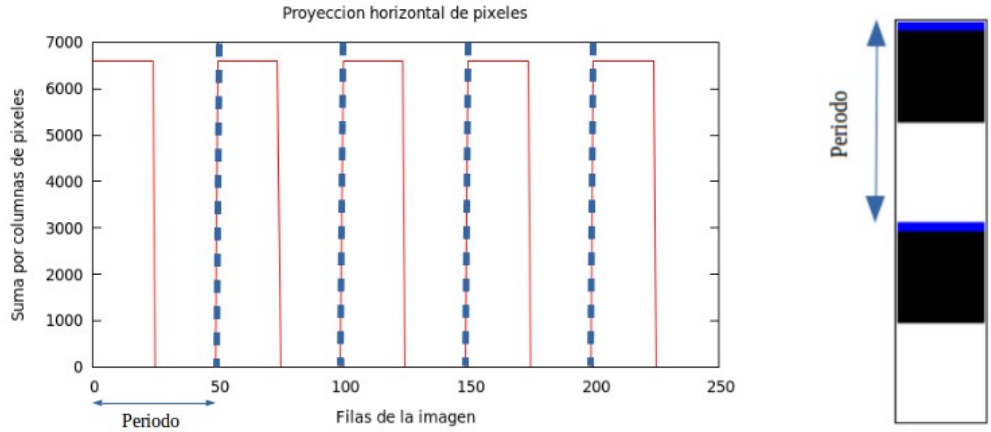


Figura 2.5: Representación del periodo de la frecuencia fundamental en vector negro-blanco.

Observamos que la frecuencia fundamental corresponde a la distancia entre dos líneas de texto consecutivas, esta distancia la vemos representada como un componente negro-blanco en el vector de proyección. La altura de la *x* en este ejemplo viene dada por la altura de un cuadrado negro, en este ejemplo ideal la altura de la *x* es exactamente la mitad del periodo obtenido. Para un caso ideal resulta trivial encontrar la altura de la *x*, cuando trabajamos con texto manuscrito las cosas se complican considerablemente por diversos motivos como el ruido que aparece en las

imágenes. El ruido afecta al vector de proyección y como consecuencia la salida de la transformada también resulta afectada y su análisis y estudio es más complejo. El vector de proyección tiene una talla discreta que corresponde con el número de filas que tiene la imagen, por lo tanto estamos trabajando con una transformada discreta de Fourier (DFT).

Los textos tienen un conjunto de características que se repiten siempre, es decir, en un texto siempre encontramos una separación entre líneas, márgenes al principio y al final del texto, letras en mayúscula y letras en minúscula, separaciones entre párrafos…
Cuando trabajamos con textos densos en líneas, de todas estas características hay una que es la que más se repite respecto a las demás, esta característica corresponde a la separación entre líneas de un texto.

Observando la solución teórica que hemos mostrado en este trabajo, nos encontramos con la primera restricción en nuestro sistema y es que la imagen del texto introducido como entrada en nuestro sistema debe tener un conjunto de líneas considerable, lo comprobamos a continuación:

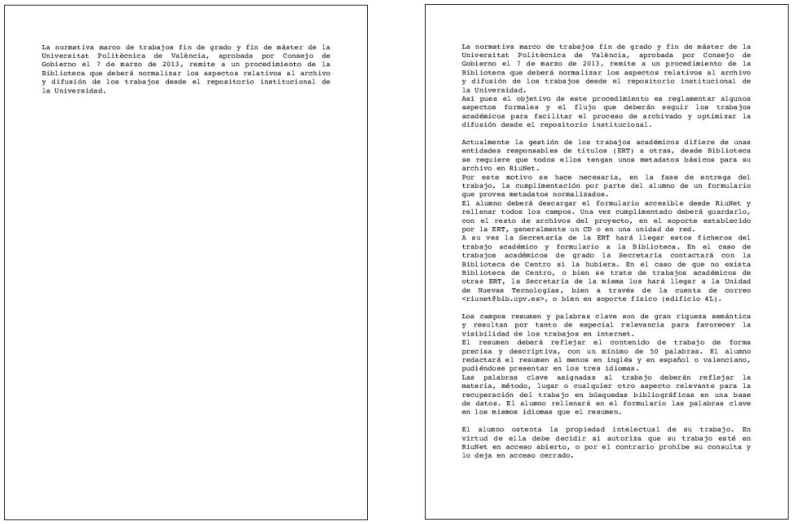


Figura 2.6: Imágenes con diferente número de líneas.

La dos imágenes corresponden al mismo texto, la diferencia se encuentra en que la imagen de la izquierda contiene el primer párrafo del texto y la imagen de la derecha contiene el texto completo. Si a estas dos imágenes les realizamos una proyección horizontal y obtenemos sus correspondientes vectores de proyección, al representarlos gráficamente observamos lo siguiente:

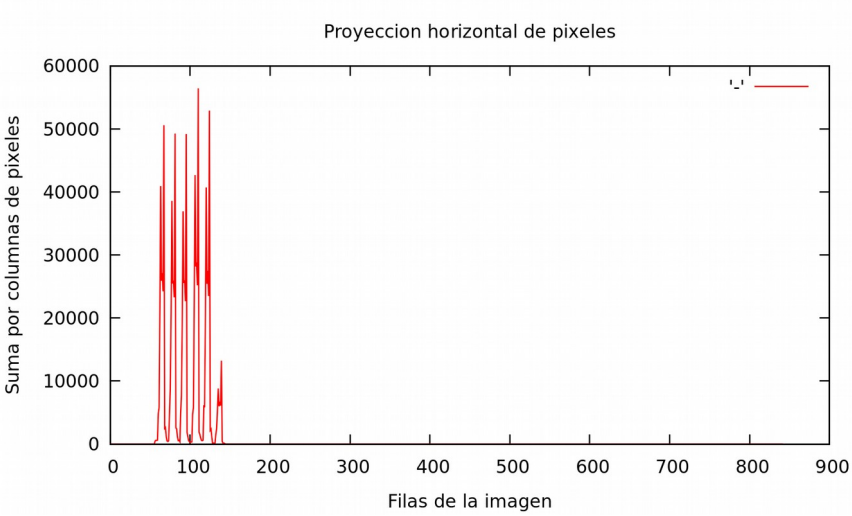


Figura 2.7: Proyección horizontal de texto con pocas líneas.

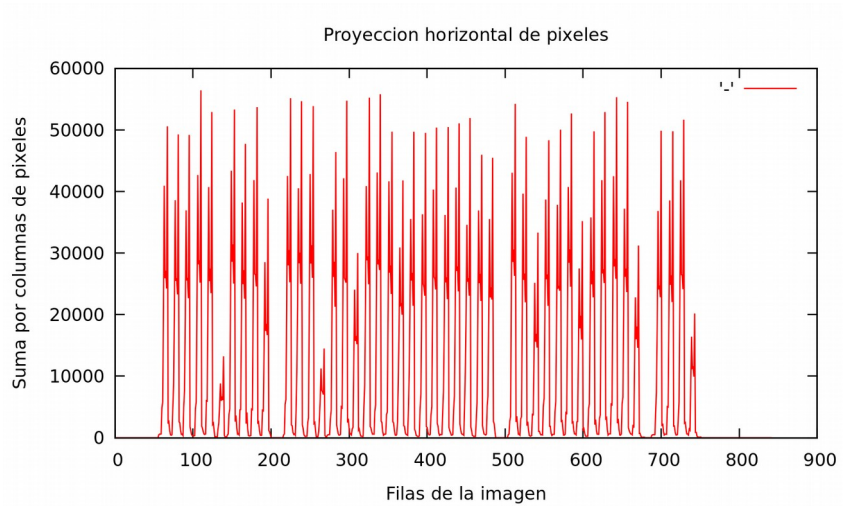


Figura 2.8: Proyección horizontal de texto con muchas líneas.

Observamos que en el texto con pocas líneas, al realizar la proyección obtenemos una señal con poca información acerca del texto y mucha información acerca del fondo, en este caso al ser un fondo completamente blanco la proyección es cero. En el texto con gran cantidad de líneas vemos que la proyección tiene mucha más información acerca del texto y muy poca acerca del fondo ya que solo corresponde al fondo los márgenes y los cambios de párrafo. Si le aplicamos la DFT a ambas proyecciones obtendremos dos espectros diferentes, provocando que la frecuencia fundamental no sea la misma, lo vemos gráficamente a continuación:

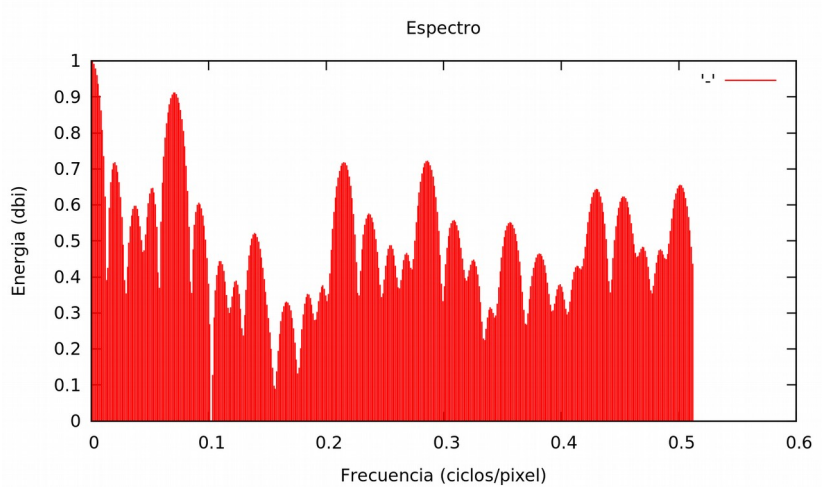


Figura 2.9: Espectro de frecuencias de texto con pocas líneas.

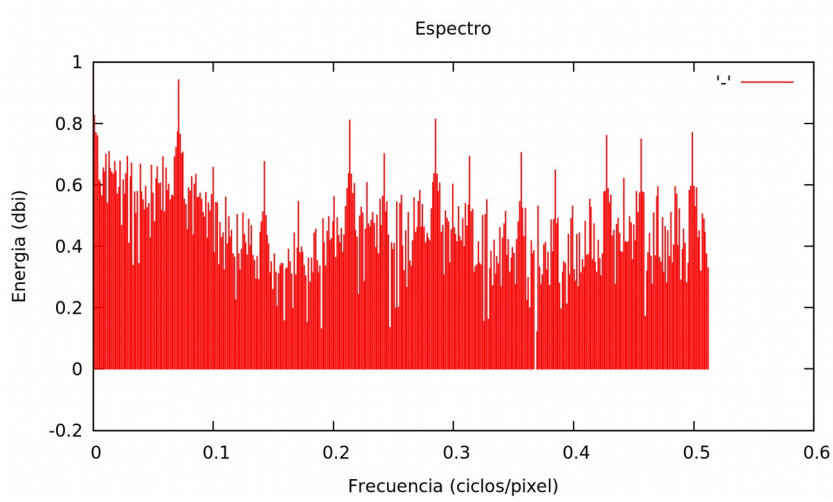


Figura 2.10: Espectro de frecuencias de texto con muchas líneas.

Observamos que para poder trabajar con la frecuencia fundamental en el espectro de frecuencias, el vector proyección debe contener suficiente información, esta información se mide con el número de líneas de texto que contenga la imagen, como es lógico cuantas más líneas de texto tenga más precisión nos devuelve la DFT por lo tanto es de gran importancia trabajar con imágenes de textos densos en líneas.

La frecuencia fundamental nos devuelve una distancia entre la separación de líneas consecutivas, como ya hemos expuesto en la introducción nuestro objetivo no es encontrar esta distancia, el objetivo consiste en encontrar la altura de la *x* del texto, esta altura también se repite considerablemente en la proyección horizontal y lo que buscamos es encontrar un patrón en el espectro de frecuencias que nos permita localizar dicha frecuencia y de este modo poder automatizar el sistema.

Es importante tener en cuenta que los textos con los que trabajamos pueden contener ruido, este ruido lo hemos estudiado en este trabajo y hemos sacado unas conclusiones que nos permiten establecer limitaciones dentro de nuestro sistema. El ruido afecta directamente a la proyección horizontal y hace que no sea trivial analizar los patrones que se repiten con mayor frecuencia dentro de la imagen llegando incluso a distorsionar tanto la imagen que nuestro sistema no responde correctamente.

Otra parte a tener en cuenta cuando trabajamos con la DFT es el tema de coste computacional, el algoritmo que se emplea para calcular la transformada recibe el nombre de transformada rápida de Fourier (FFT), las librerías actuales utilizan este algoritmo y tiene un coste de n * log(n) siendo n en nuestro caso el tamaño del vector de proyección que corresponde al número de filas de la imagen de entrada.

3. Implementación

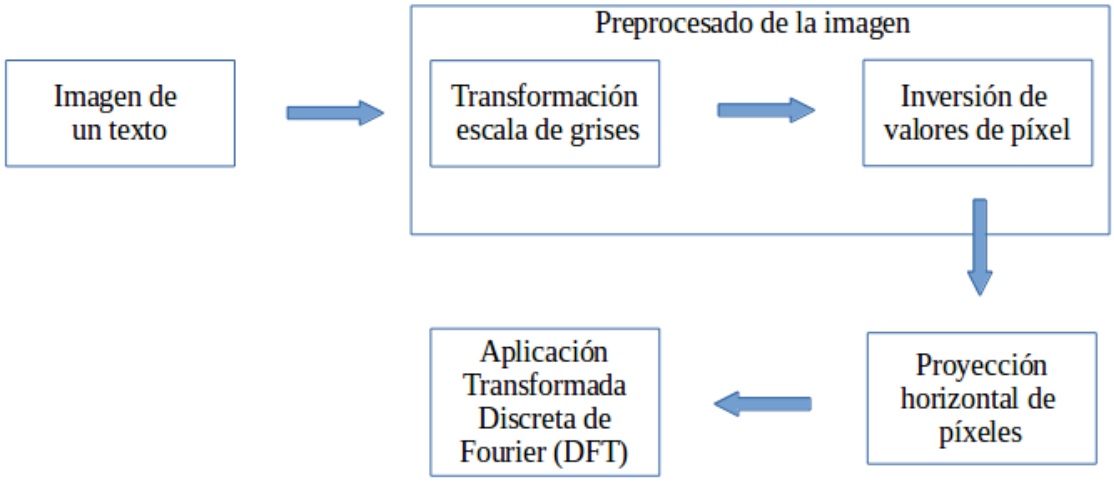


Figura 3.1: Esquema base de nuestro sistema.

Una imagen se representa normalmente con un sistema denominado RGB (rojo, verde y azul) que representa tres canales, cada canal es una matriz de la misma dimensión que la imagen que representa el valor de ese color en ese píxel, por lo tanto una imagen esta compuesta de tres matrices, una para los valores de color rojo, otra para los valores de color verde y una última para los valores de color azul, la combinación de los tres colores en cada píxel forman la imagen.

Nosotros únicamente queremos información acerca de las líneas que tiene un texto, esto nos permite realizar una transformación de RGB a escala de grises ya que el color de la imagen en nuestro sistema no nos proporciona información, la información que necesitamos es la intensidad de cada píxel para diferenciar que son líneas de texto y que es fondo del texto. Realizar una transformación a escala de grises consiste en pasar de tener una imagen de tres canales (tres matrices) a tener una imagen de un canal (una única matriz) donde cada píxel pasa a tomar un valor entre 0 y 255 siendo 0 el valor negro y 255 el valor blanco, la finalidad consiste en trabajar únicamente con una matriz.

Invertimos los valores de la matriz para representar el valor negro con el 255 y el valor blanco con el 0, de este modo en la gráficas que mostraremos posteriormente los picos de la gráfica representan las líneas de píxeles con mayor intensidad de negro.

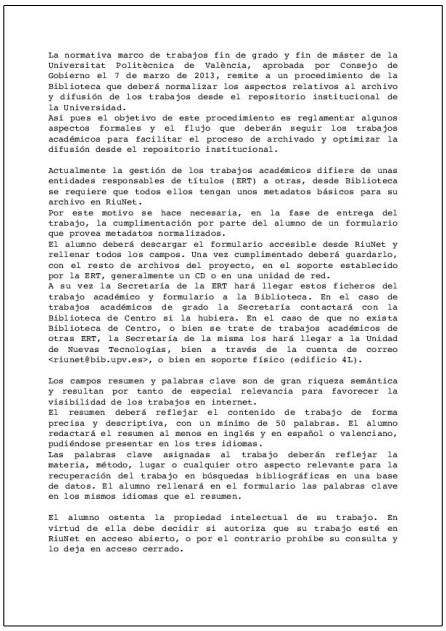
La proyección horizontal de píxeles consiste en sumar para cada fila de la imagen todas sus columnas, con esto obtenemos como resultado un vector de tamaño el número de filas que nos proporciona información acerca del peso que tiene cada fila de la imagen. El vector de proyección resultante es la entrada a la DFT que nos devuelve el espectro de frecuencias que componen la señal que representa el vector.

Este esquema es la base de nuestro sistema que como es de esperar va a generar un conjunto de problemas que vamos a tener que resolver, cada problema lo trataremos concretamente cuando aparezca en la memoria explicando la solución aportada.

Como primera aproximación utilizamos imágenes con texto impreso, esto nos permite tener un texto con características ideales para exponer nuestro trabajo de forma sencilla y sin tener que resolver todos los problemas al mismo tiempo. Una vez hemos obtenido un sistema que funcione bien con textos impresos, empezamos a trabajar con textos manuscritos identificando y resolviendo los problemas que nos encontramos.

3.1 Texto impreso

Los textos manuscritos como ya hemos comentado suelen presentar ruido, que afecta a la proyección horizontal de píxeles, por este motivo para empezar a tomar contacto con el problema es recomendable trabajar con un texto que nos proporcione una proyección limpia y poder sacar un conjunto de conclusiones iniciales. Por este motivo el primer texto con el que trabajamos es un texto impreso generado por un procesador de texto, con una letra muy regular y con un fondo muy diferenciado con la misma tonalidad de color blanco.



La normativa marco de trabajos fin de grado y fin de máster de la Universitat Politècnica de València, aprobada por Consejo de Gobierno el 7 de marzo de 2013, remite a un procedimiento de la Biblioteca que deberá normalizar los aspectos relativos al archivo y difusión de los trabajos desde el repositorio institucional de la Universidad.
Así pues el objetivo de este procedimiento es reglamentar algunos aspectos formales y el flujo que deberán seguir los trabajos académicos para facilitar el proceso de archivado y optimizar la difusión desde el repositorio institucional.

Actualmente la gestión de los trabajos académicos difiere de unas entidades responsables de títulos (ERT) a otras, desde Biblioteca se requiere que todos ellos tengan unos metadatos básicos para su archivo en RiuNet.
Por este motivo se hace necesaria, en la fase de entrega del trabajo, la cumplimentación por parte del alumno de un formulario que provea metadatos normalizados.
El alumno deberá descargar el formulario accesible desde RiuNet y rellenar todos los campos. Una vez cumplimentado deberá guardarlo, con el resto de archivos del proyecto, en el soporte establecido por la ERT, generalmente un CD o en una unidad de red.
A su vez la Secretaría de la ERT hará llegar estos ficheros del trabajo académico y formulario a la Biblioteca. En el caso de trabajos académicos de grado la Secretaría contactará con la Biblioteca de Centro si la hubiera. En el caso de que no exista Biblioteca de Centro, o bien se trate de trabajos académicos de otras ERT, la Secretaría de la misma los hará llegar a la Unidad de Nuevas Tecnologías, bien a través de la cuenta de correo <riunet@bib.upv.es>, o bien en soporte físico (edificio 4L).

Los campos resumen y palabras clave son de gran riqueza semántica y resultan por tanto de especial relevancia para favorecer la visibilidad de los trabajos en internet.
El resumen deberá reflejar el contenido de trabajo de forma precisa y descriptiva, con un mínimo de 50 palabras. El alumno redactará el resumen al menos en inglés y en español o valenciano, pudiéndose presentar en los tres idiomas.
Las palabras clave asignadas al trabajo deberán reflejar la materia, método, lugar o cualquier otro aspecto relevante para la recuperación del trabajo en búsquedas bibliográficas en una base de datos. El alumno rellenará en el formulario las palabras clave en los mismos idiomas que el resumen.

El alumno ostenta la propiedad intelectual de su trabajo. En virtud de ella debe decidir si autoriza que su trabajo esté en RiuNet en acceso abierto, o por el contrario prohíbe su consulta y lo deja en acceso cerrado.

Figura 3.1.1: Ejemplo de texto impreso "procedimiento_riunet.pdf".

Estas imágenes que contienen texto son la entrada a nuestro sistema, la transformación a escala de grises en este ejemplo no afecta, ya que trabajamos directamente con color negro para la letra y color blanco para el fondo. La inversión de valores de píxel ya hemos comentado que funcionalidad tiene y en todas las ejecuciones es la misma, por lo tanto, no la mencionaremos en lo que sigue de trabajo pero es importante saber que se realiza por si en un futuro se requiere una modificación del sistema.

La proyección horizontal es el primer punto que tenemos que analizar, esta proyección nos describe como es el texto y analizarla nos permite comprender que vamos a obtener al aplicarle la DFT. Hemos visto en el apartado teórico de la solución propuesta un ejemplo con texto impreso, en el cual hemos utilizado un vector de proyección ideal (blanco-negro), en este ejemplo tenemos un texto impreso pero en cambio vamos a trabajar con el vector de proyección real, esto como observamos en la gráfica de proyección no nos devuelve una señal cuadrada como hemos visto anteriormente, nos devuelve una señal ruidosa pero que tiene una regularidad.

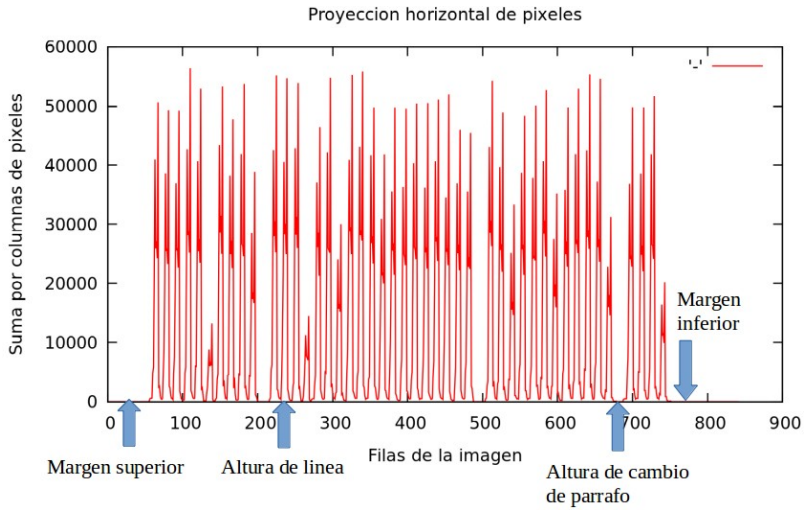


Figura 3.1.2: Proyección horizontal de píxeles del texto "procedimiento_riunet.pdf".

Con una proyección horizontal de píxeles podemos apreciar claramente en la gráfica que corresponde a una línea y que corresponde a otras características que nos podemos encontrar en un texto como puede ser en este caso un espacio en blanco que separa dos párrafos, el margen superior y el margen inferior.

Antes de aplicarle la DFT a dicha señal ya podemos tener una idea de lo que nos va a devolver, vemos en la gráfica anterior que hay una repetición predominante que representa las líneas del texto, esta repetición predominante corresponde a la frecuencia fundamental. Al aplicarle a esta señal la transformada comprobamos que el resultado obtenido es el que hemos observado y analizado con la proyección.

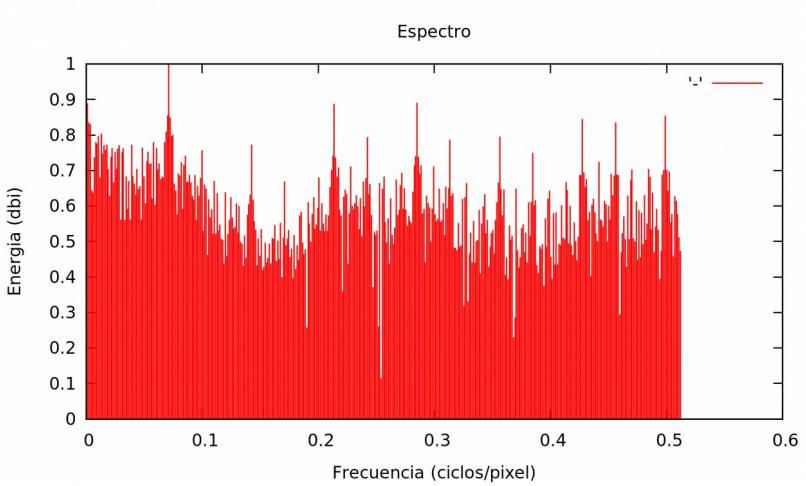


Figura 3.1.3: Espectro de frecuencias de la proyección horizontal de la figura 3.1.2.

La frecuencia fundamental es 0.071259 ciclos/píxel. El periodo es 14.03333 píxeles, el período representa la distancia entre las partes superiores de caracteres de dos líneas adyacentes que hemos comentado anteriormente, es importante remarcar que aunque estemos trabajando con un texto impreso, la altura obtenida es una aproximación ya que el texto tiene letras de diferente tamaño, es decir, una “a” tiene menos altura que una “t”, también las letras en mayúscula tienen mayor altura que las letras en minúscula, por lo tanto ya observamos ruidos en la proyección horizontal, en un ejemplo que es el mejor texto que nos podemos encontrar para nuestro sistema.

En la siguiente imagen vemos el primer párrafo del texto de la ilustración 3.1.1 segmentada en líneas de una altura correspondiente al periodo obtenido, es decir, 14.03333 píxeles.

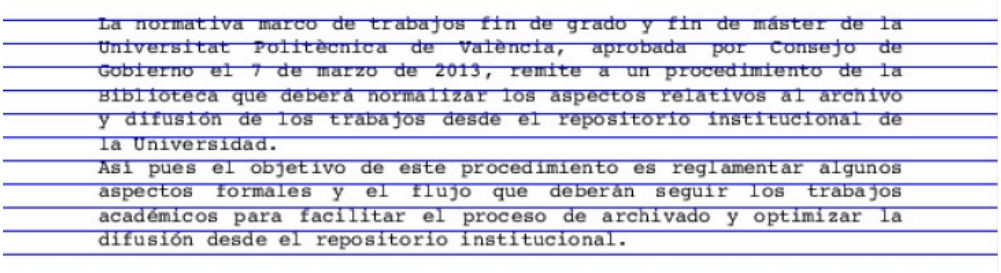


Figura 3.1.4: Primer párrafo del texto de la ilustración 3.1.1 segmentado con el periodo obtenido.

Observamos que al ser una aproximación de la altura existe un desfase, lo podemos observar comparando la primera línea con la última línea de la imagen de texto segmentado. El desfase se verá posteriormente en el apartado de segmentación del texto en líneas.

Probando diferentes alturas de forma empírica sobre el texto hemos obtenido la altura de la *x* que buscamos, hemos sacado su correspondiente frecuencia y la hemos buscado en la gráfica del espectro de frecuencias con la finalidad de estudiar si ese patrón de frecuencias se repite y poder automatizar el sistema. Observamos que la frecuencia que buscamos corresponde a una frecuencia que esta situada entre la frecuencia fundamental y la siguiente frecuencia con intensidad de energía más alta.

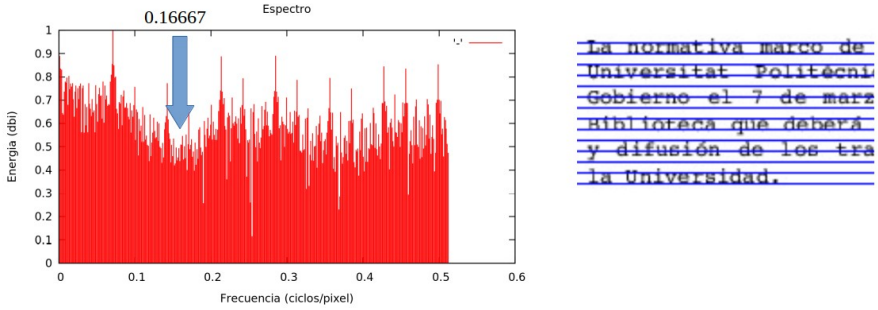


Figura 3.1.5: Frecuencia que nos proporciona la altura de la x.

Esta frecuencia que corresponde a la altura de la *x* no sigue un patrón concreto para todos los textos, por lo tanto no podemos utilizar este método para solucionar el problema, para demostrarlo simplemente utilizamos dos textos manuscritos que presentan una separación entre líneas de diferente tamaño:

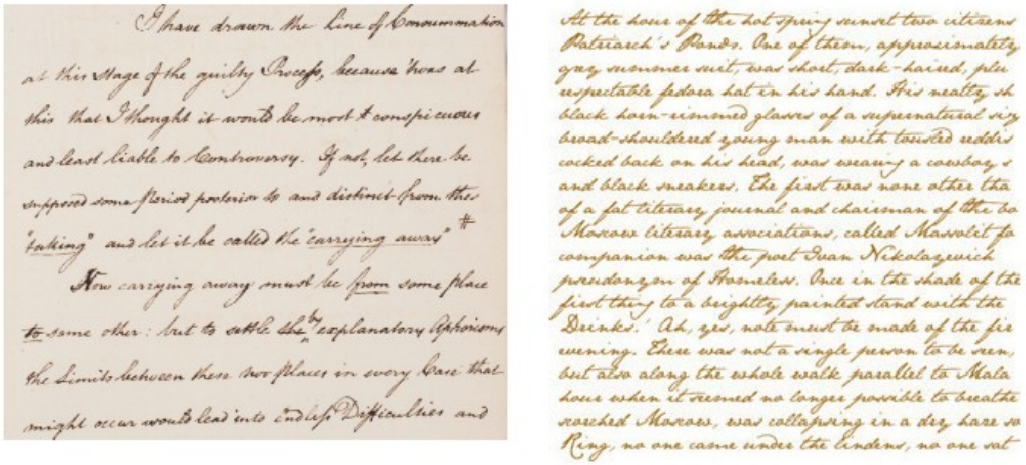


Figura 3.1.6: Imágenes con diferente distancia de separación de líneas.

Si le aplicamos a ambos textos la proyección horizontal y obtenemos posteriormente su espectro de frecuencias observamos lo siguiente:

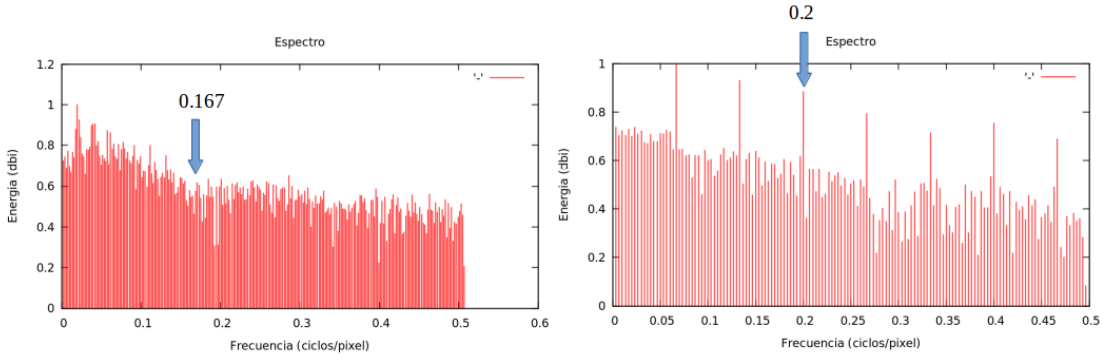


Figura 3.1.7: Espectro y frecuencia de la altura de la x de las imágenes de la figura 3.1.6.

Observamos que la distribución de frecuencias en el espectro es totalmente diferente, en la primera imagen la frecuencia que buscamos se encuentra muy lejos de la tercera frecuencia con mayor intensidad, sin embargo en la segunda imagen es exactamente la tercera frecuencia con mayor intensidad. Al no encontrar un patrón que se repita siempre y que nos proporcione la altura de la *x* del texto, no podemos obtener un algoritmo que funcione sobre el espectro de frecuencias y que nos proporcione una solución al problema, sin embargo, la frecuencia fundamental si que nos devuelve siempre la misma información acerca de la distancia de líneas consecutivas.

Por lo tanto únicamente con la DFT no podemos abordar el problema y obtener una solución útil, esto nos lleva a afrontar el problema de forma distinta.
Cuando tenemos problemas con complejidad elevada una solución racional es dividir el problema en problemas más pequeños que sean más fáciles de resolver, mediante el ejemplo anterior hemos obtenido una aproximación de la distancia entre líneas consecutivas del texto, para resolver el problema que buscamos vamos a trabajar con cada línea en vez de con el texto completo.

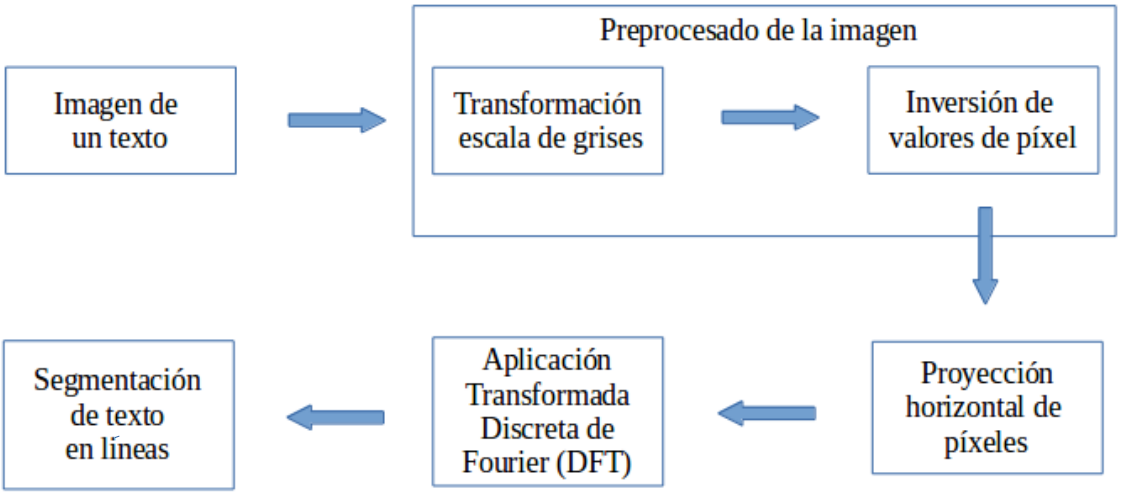
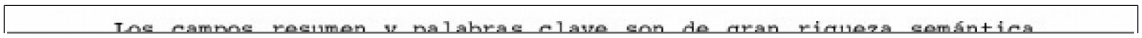


Figura 3.1.8: Esquema con módulo de segmentación del texto en líneas.

El módulo de segmentación de texto en líneas se basa en trabajar principalmente con la distancia entre líneas que nos proporciona la frecuencia fundamental de la DFT, al trazar líneas horizontales con una separación igual a dicha distancia observamos que las líneas de texto no quedan en muchos casos bien segmentadas, esto se debe a que la distancia obtenida es una aproximación debido a que los espacios que separan las líneas de los textos (generalmente en textos manuscritos) no suelen tener el mismo tamaño, además como hemos visto los textos suelen tener márgenes, letras en mayúsculas y separaciones entre párrafos, todas estas características hacen que aparezca un desfase.

El desfase lo hemos corregido trabajando con una ventana sobre cada línea, esta ventana se trata de una región de interés (del inglés ROI), que corresponde a cada segmentación de línea obtenida por la distancia entre líneas que nos devuelve la DFT. El algoritmo que hemos implementado consiste en reposicionar la ROI con el fin de colocar la línea dentro de la ventana y de este modo poder trabajar correctamente sobre ella. A modo de ilustración el algoritmo lo explicamos basándonos en la segmentación incorrecta de una línea sobre el texto impreso con el que estamos trabajando, presentando los pasos que realiza el algoritmo para que el texto se posicione correctamente en el interior de la ROI.



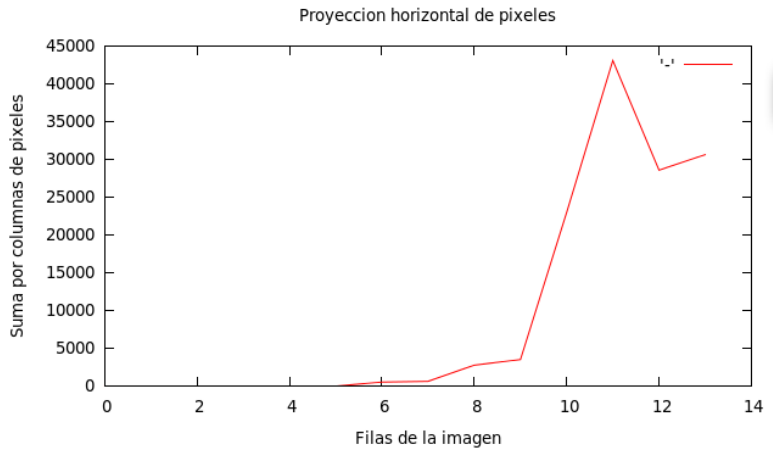


Figura 3.1.9: ROI de una línea mal posicionada.

Observamos que la segmentación de la línea mediante la ROI no incorpora en su interior la línea completa, también lo podemos ver en la gráfica de su proyección horizontal, donde vemos que perdemos información por la parte derecha de la gráfica, para corregirlo aplicamos un algoritmo sobre la proyección horizontal de la ROI que explicamos a continuación.

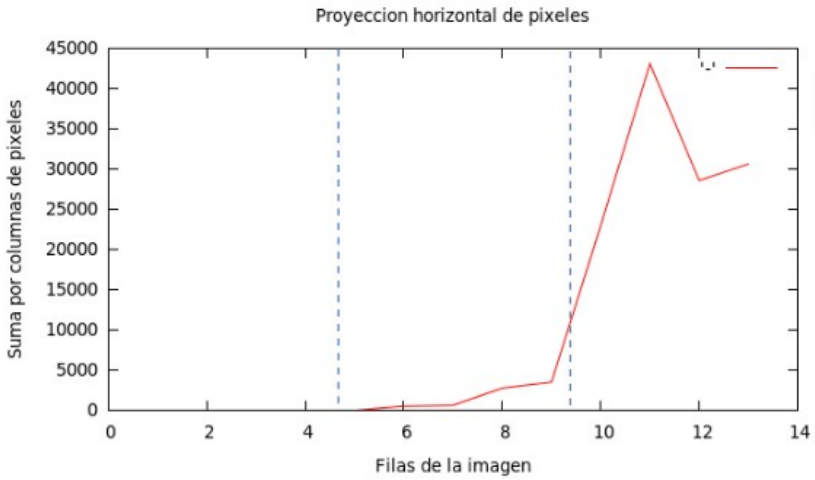


Figura 3.1.10: División de la ventana (ROI) en tres partes.

Ahora la ROI la tenemos dividida en tres ventanas de igual proporción, el algoritmo se basa en encontrar el pico más alto y determinar en que ventana de las tres esta situado:

- Pico más alto situado en la ventana izquierda: Esto implica que movamos la ROI hacia la izquierda.
- Pico más alto situado en la ventana central: El sistema deja la ROI como esta.
- Pico más alto situado en la ventana derecha: Esto implica que movamos la ROI hacia la derecha.

La cantidad de píxeles que movemos la ROI a la derecha o a la izquierda viene definido por la diferencia en el eje X (“Filas de la imagen”) entre el pico más alto y la mitad de la ventana ROI, que como ya sabemos, corresponde a la mitad de la distancia que devuelve la DFT.

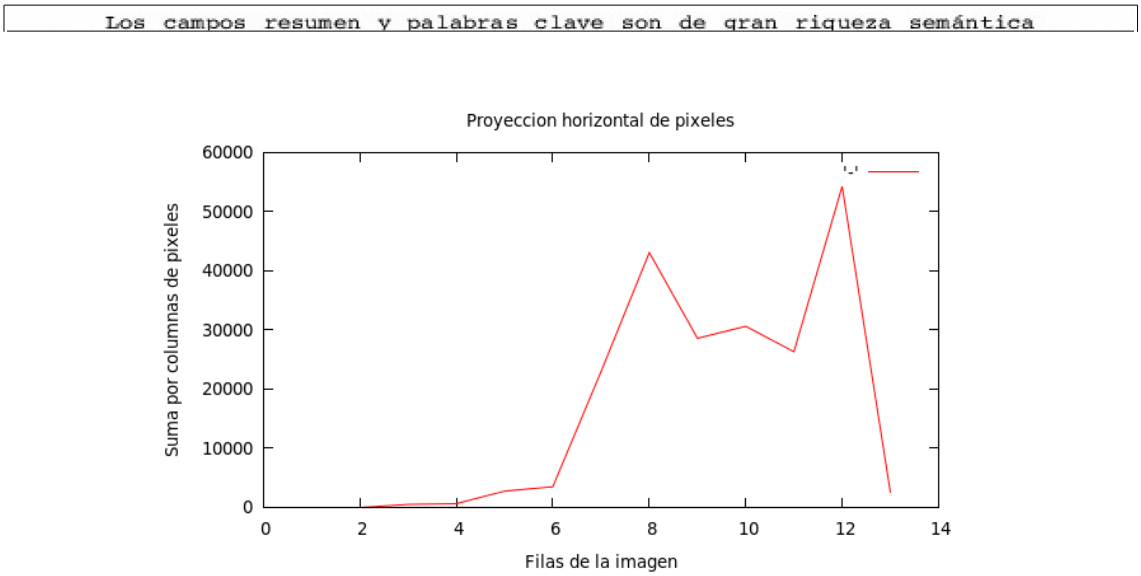


Figura 3.1.11: Corrección de la ROI de la figura 3.1.9.

El algoritmo que hemos visto es un heurístico, es fácil prever que en algunos casos no posicionará la línea correctamente, estos casos principalmente se dan cuando el pico que corta la ROI es una parte de un pico real de la proyección, esto implica que cuando movamos la ROI la proyección no quede completamente dentro de la ventana. Al tener una aproximación de la distancia que hay entre líneas, en el momento que el algoritmo sitúa una línea en el interior de la ROI las líneas consecutivas las sitúa correctamente.

Posteriormente veremos que las líneas que no contiene la línea completa en el interior de la ventana (ROI) las descartamos mediante una ponderación de calidad de la línea.

A continuación mostramos la segmentación en líneas del texto impreso con el que estamos trabajando en esta parte del trabajo, observamos que aísla correctamente las líneas de texto en sus ROI correspondientes.

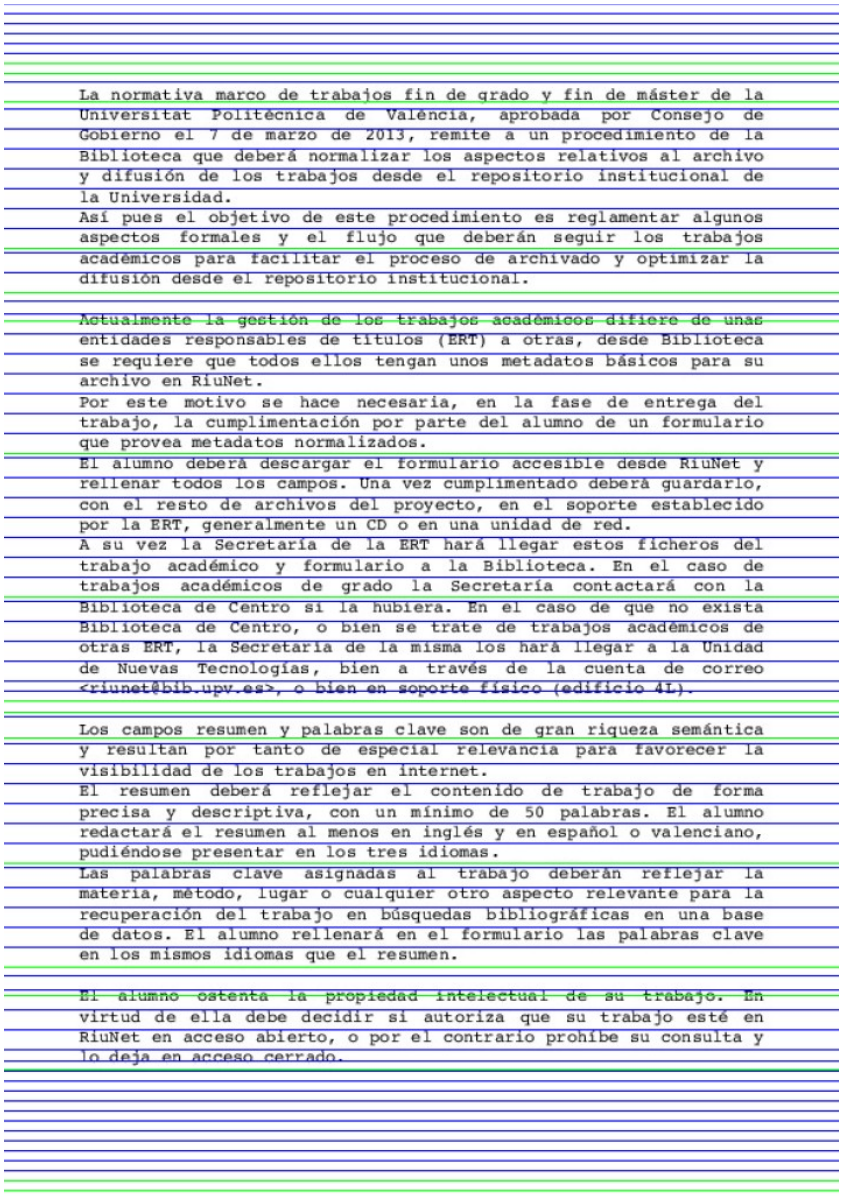


Figura 3.1.12: Texto de la figura 3.1.1 segmentado en líneas.

Para explicar como representamos la segmentación gráficamente sobre el texto vemos la siguiente ilustración.

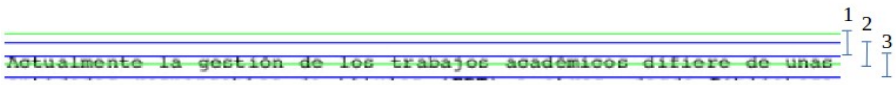


Figura 3.1.13: Algoritmo de segmentación en líneas de la ilustración 3.1.12.

ROI 1: La ventana de proyección horizontal de la ROI tiene el pico en la parte derecha de la ventana, se reposiciona la ROI hacia la derecha (hacia abajo en la imagen).
ROI 2: Esta ROI el algoritmo la interpreta como una línea de texto, es un error en el algoritmo heurístico, este error se elimina con la ponderación de líneas.

ROI 3: El algoritmo heurístico corrige la línea de la ROI 2, siendo la nueva línea que interpreta el sistema la línea en el interior de la ROI 3.
La altura de la ROI siempre tiene el mismo tamaño que como hemos explicado anteriormente es el valor obtenido de la frecuencia fundamental del espectro de la DFT.

Como podemos observar el algoritmo separa las líneas correctamente, hemos de tener en cuenta que no buscamos que las líneas estén perfectamente dentro de la ROI, simplemente que se sitúen en el interior ya que si están dentro podemos trabajar con la proyección horizontal para obtener la altura de la *x*.

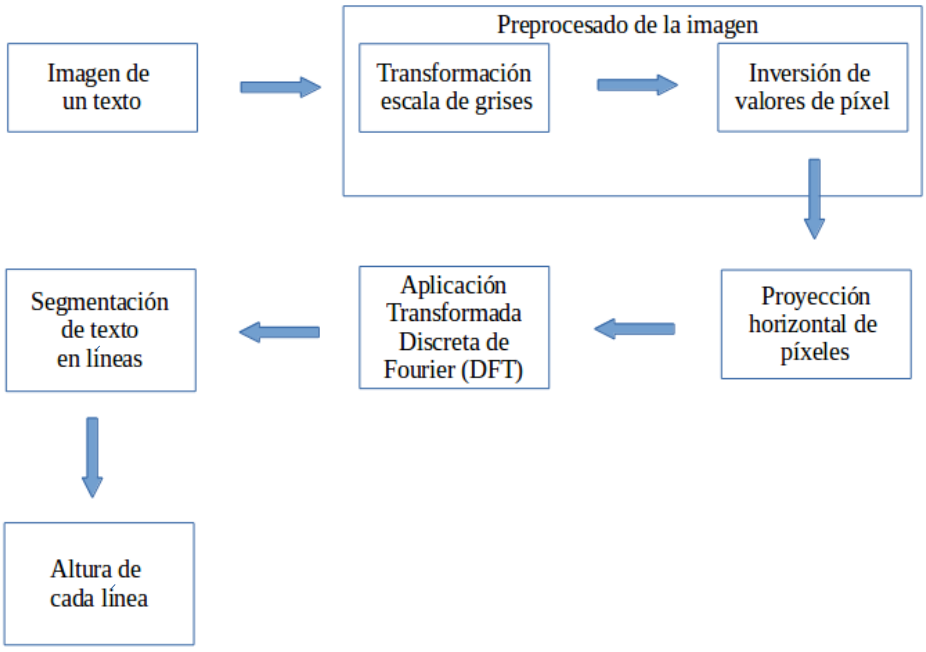


Figura 3.1.14: Esquema con módulo de altura de cada línea.

Una vez tenemos el texto segmentado en líneas, el siguiente paso es trabajar con cada línea para obtener la altura de la *x*, esta altura nos la proporciona la proyección horizontal de la ROI. Es importante entender que significado tiene la gráfica que nos proporciona la proyección horizontal sobre la ROI.

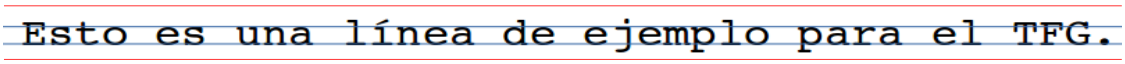


Figura 3.1.15: Elementos en la proyección de una ROI.

Las dos líneas exteriores (rojas) representan la ROI que segmenta esta línea en el texto, las dos líneas interiores (azules) representan la altura de la *x* que buscamos en nuestro trabajo, vamos a mostrar a continuación su proyección horizontal.

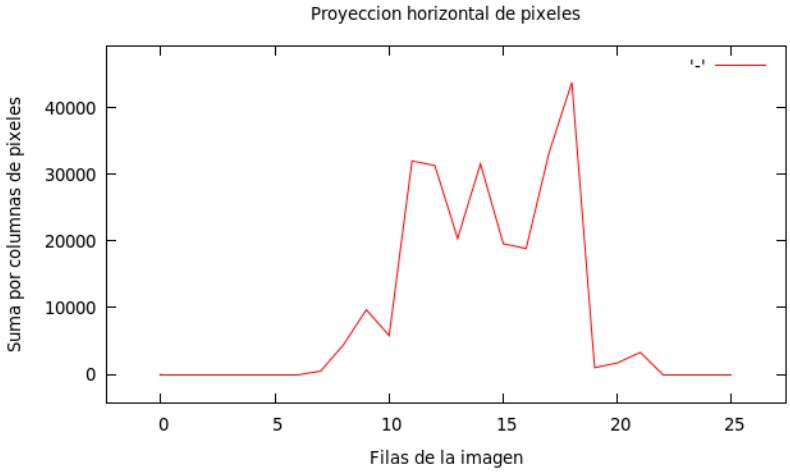


Figura 3.1.16: Proyección horizontal de la figura 3.1.15.

A simple vista podemos observar que los dos picos extremos son la altura de la *x* que buscamos. El pico que se produce en medio se debe a la proyección horizontal del tronco de los caracteres, en este ejemplo podemos observar que ese pico lo producen las letras como la “s”, “e”, “a” y “m”.

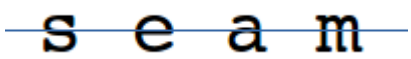


Figura 3.1.17: Proyección horizontal que provoca el pico central de la figura 3.1.16.

En texto impreso mediante un algoritmo trivial podríamos obtener los dos picos extremos que buscamos fácilmente ya que son los que más altura tienen, el problema viene cuando trabajamos con texto manuscrito donde la proyección horizontal de la ROI puede que no tenga los picos más altos en los extremos por motivos como manchas de tinta a causa del borrado de palabras, desviación horizontal de las líneas al escribir, tonos más oscuros de fondo en partes del documento, etc.
Todos estos aspectos ruidosos hacen que el algoritmo trivial del que hablábamos se complique bastante, nosotros hemos optado por relajar el problema y buscar una aproximación, el algoritmo consiste en obtener lo que denominamos un valor de discretización:

valor_discretizar = valorY_min + ((valorY_max – valorY_min)/3)

Este valor nos permite transformar la señal de la proyección horizontal en una señal cuadrada, todos los valores que se encuentren por encima del *valor_discretizar* forman la parte alta de la señal cuadrada, los valores que estén por debajo del *valor_discretizar* forman la parte baja. El algoritmo para encontrar los dos puntos extremos más altos de la señal vuelve a ser trivial ya que trabajamos con una señal cuadrada y por lo tanto únicamente tenemos dos valores, el *valorY_min* que es la parte baja de la señal y el *valorY_max* que es la parte alta.

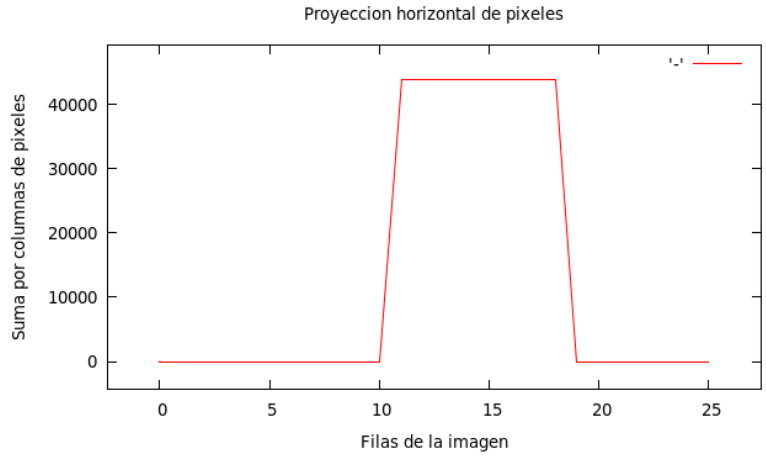


Figura 3.1.18: Discretización de la proyección horizontal de la figura 3.1.16.

Hemos de comentar que pueden existir casos en los que hayan más de una señal cuadrada, esto se puede producir por el ruido que aparece en la proyección horizontal en los textos manuscritos, vemos un ejemplo a continuación:

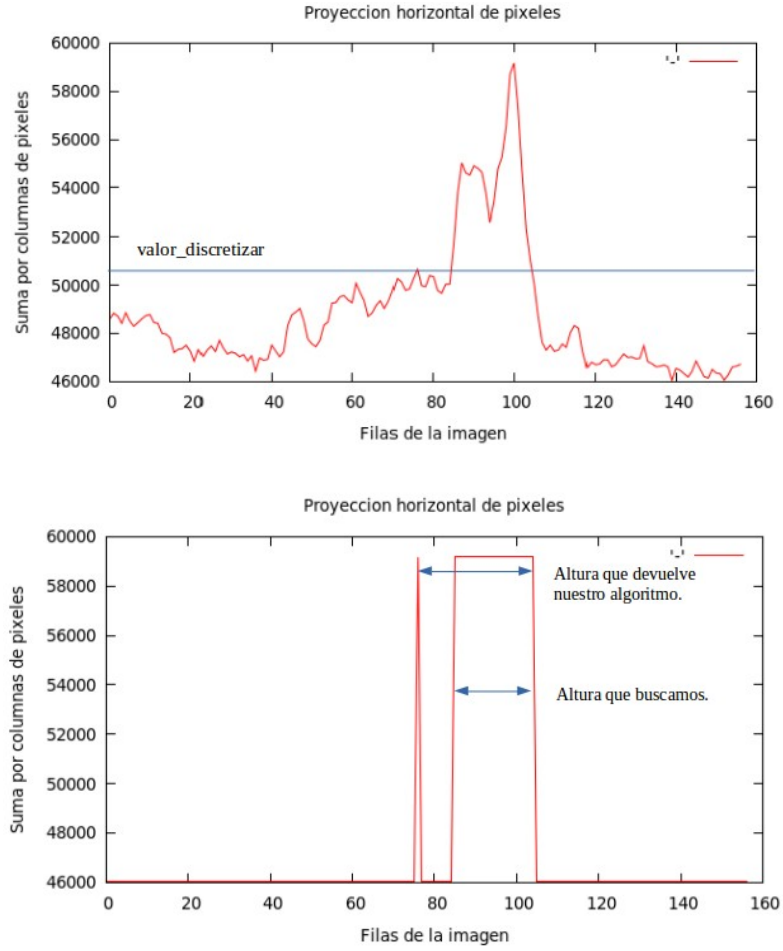


Figura 3.1.19: Obtención errónea de la altura de la x a causa de ruido en la proyección.

En este ejemplo, la altura que nos proporcionaría la señal cuadrada no es la que buscamos, esto es un error que aparece por el ruido que hemos comentado anteriormente. En los casos en los que no segmentemos correctamente la línea, o busquemos la altura de la *x* de una línea que no contiene texto, la obtención de la altura también nos devolverá una altura errónea, es decir, una altura que no nos proporcione información relevante. Estos problemas los afrontaremos en el apartado de ponderación de líneas.

Nuestro sistema proporciona como salida una altura de la *x* para cada una de las líneas de segmentación, es decir, para cada una de las ROI. Es importante ver el resultado global que proporciona como salida nuestro sistema, ya que en la explicación de los algoritmos que hemos implementado hemos utilizado una segmentación de líneas preparada intencionadamente para comprender el funcionamiento de los mismos.

A continuación realizamos la ejecución de nuestro sistema utilizando como imagen de entrada el texto impreso de la figura 3.1.1. Las alturas que devuelve el sistema las mostramos en la siguiente tabla, observamos que el sistema devuelve diferentes valores, es interesante relacionar las alturas con la segmentación de la imagen para comprender que salida nos esta proporcionando el sistema.

Alturas de la *x* obtenidas		
ROI	**Altura X (Píxeles)**	**Motivo**
1-6	13	Ruido: Algoritmo de obtención de altura con líneas vacías.
7-16	6	Altura correcta.
17	3	Ruido: línea no segmentada correctamente.
18-36	6	Altura correcta.
37	1	Ruido: línea no segmentada correctamente.
38-49	6	Altura correcta.
50	2	Ruido: línea no segmentada correctamente.
51-54	6	Altura correcta.
55	0	Ruido: línea no segmentada correctamente.
56	1	Ruido: línea no segmentada correctamente.
57-66	13	Ruido: Algoritmo de obtención de altura con líneas vacías.

Tabla 3.1.1: Salida de nuestro sistema para texto de la figura 3.1.1.

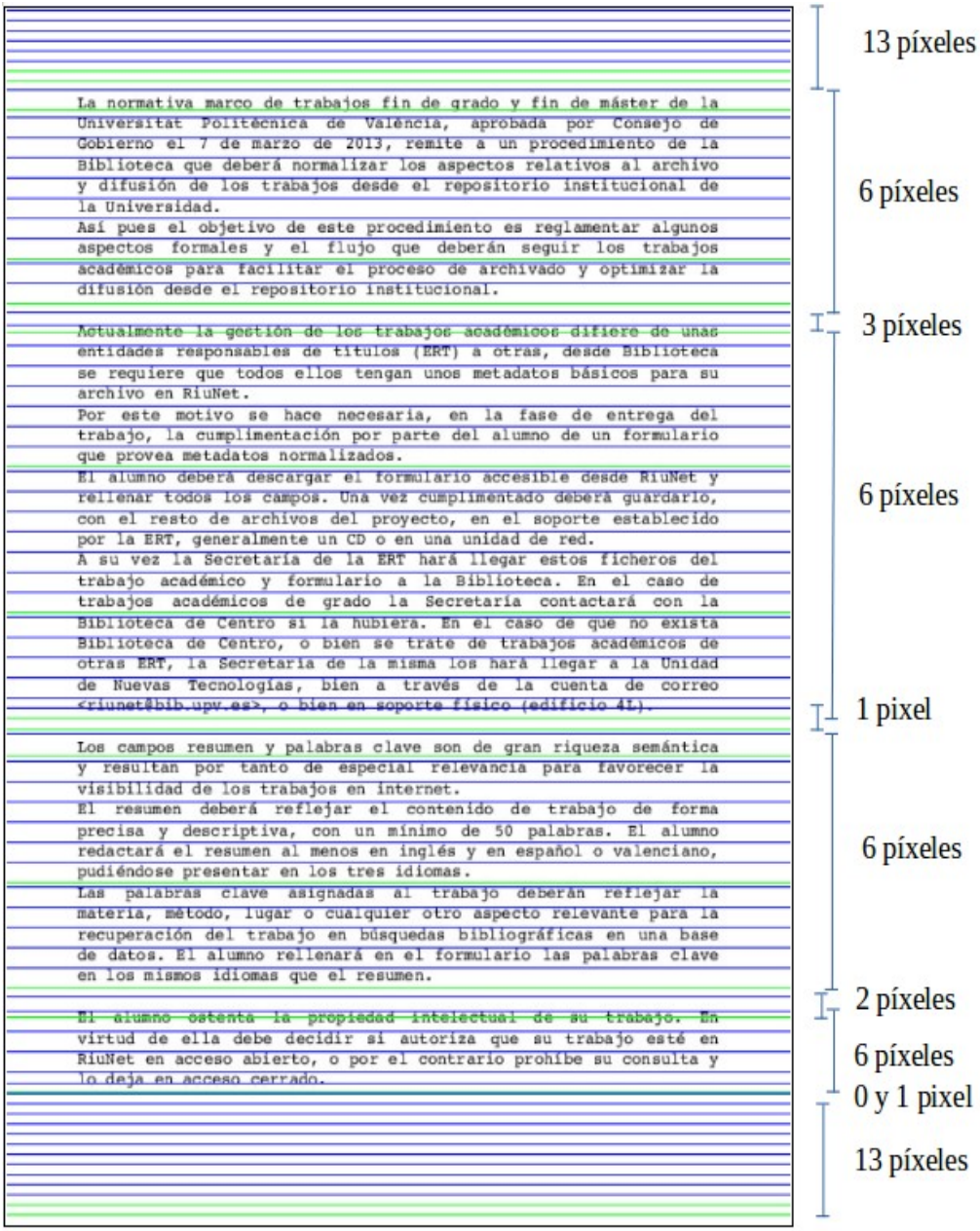


Figura 3.1.20: Resultado de la ejecución de nuestro sistema sobre el texto de la figura 3.1.1.

Observamos que el valor que más se repite es la altura de la *x* seis píxeles que corresponde a los párrafos que forman el texto, si segmentamos el texto horizontalmente con seis píxeles observamos que la altura de la *x* obtenida es correcta y que el sistema funciona correctamente.

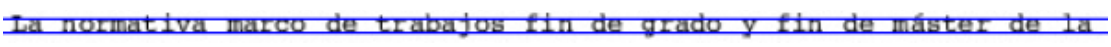


Figura 3.1.21: Altura de la x estimada para el texto de la figura 3.1.1.

3.2 Texto manuscrito

Los textos manuscritos suelen tener ruido y esto implica que tengamos que tener en cuenta un conjunto de factores para que nuestro sistema funcione correctamente.
El ruido que afecta directamente a nuestro sistema es el siguiente:

- Diferentes tonalidades de color en la imagen: El fondo de los textos manuscritos puede tener diferentes tonalidades, sobretodo en textos antiguos. Estos cambios de tonalidad producen una proyección horizontal ruidosa y esto provoca que la señal obtenida como entrada a la DFT no sea tan regular como nos gustaría.
- Desviación en la alineación horizontal de las líneas: Las personas cuando escribimos tendemos a trazar líneas que no están completamente alineadas horizontalmente y tampoco son paralelas, esto provoca que al realizar la proyección horizontal no podamos trazar una línea horizontal que separe dos líneas correctamente. Este problema también afecta a la proyección horizontal y provoca que al aplicar la DFT no nos devuelva la distancia entre líneas que buscamos para segmentar correctamente el texto.

Es importante estudiar el efecto que provoca las diferentes tonalidades de color de fondo que presentan algunos textos manuscritos, con este estudio podemos sacar limitaciones en la entrada debido a que es necesario que se diferencie en la imagen la parte que corresponde al fondo con la parte que corresponde a texto. Las diferentes variaciones de color en el texto afectan a la proyección horizontal, la señal que representa el vector de proyección horizontal como ya hemos estudiado debe seguir una periodicidad que sea dominante respecto al conjunto de señales que existen en la proyección, este tipo de ruido perjudica a la periodicidad de la señal como veremos a continuación.

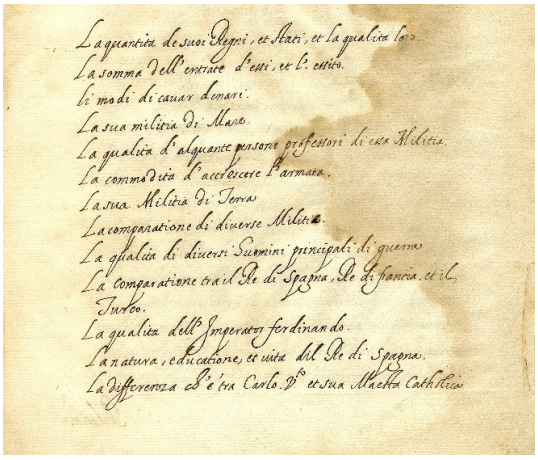


Figura 3.2.1: Texto manuscrito deteriorado.

Si realizamos la proyección horizontal y visualizamos la gráfica:

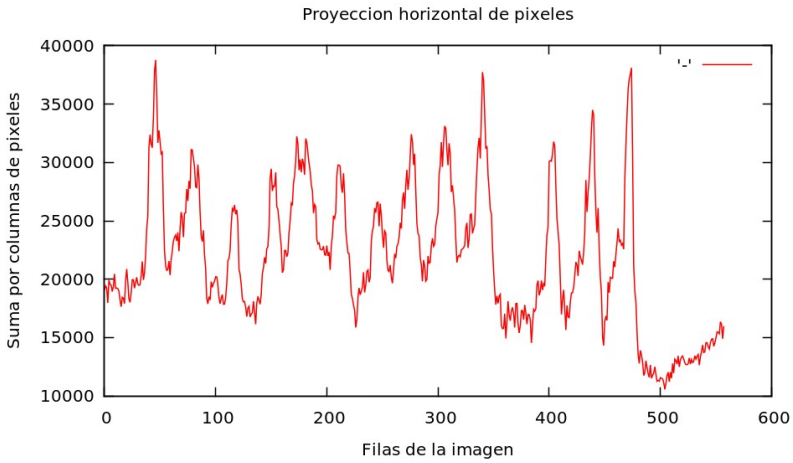


Figura 3.2.2: Proyección horizontal del texto de la figura 3.2.1.

Observamos que la señal de la proyección es muy irregular, esto afecta considerablemente a la DFT obteniendo un espectro de frecuencias con ruido, el cual no podemos analizar correctamente porque no podemos saber que energías son correctas para cada una de las frecuencias.
Al aplicar la DFT en este ejemplo observamos que la frecuencia fundamental no es la esperada ya que la señal es muy irregular a causa del ruido de la imagen:

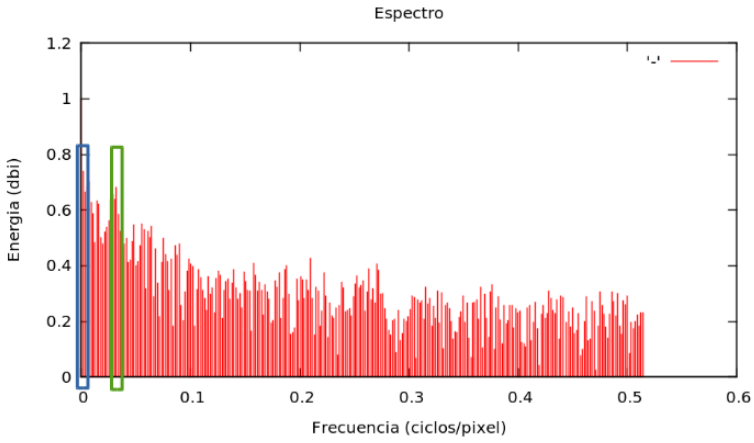


Figura 3.2.3: Espectro de la proyección horizontal de la figura 3.2.2.

La frecuencia fundamental que devuelve en este ejemplo el sistema es la que se encuentra situada dentro del rectángulo izquierdo (azul), esta frecuencia es errónea ya que la que buscamos es la que se encuentra situada en el rectángulo derecho (verde). Observamos segmentando la imagen con el periodo de la frecuencia fundamental que corresponde al rectángulo izquierdo (azul) que es errónea.

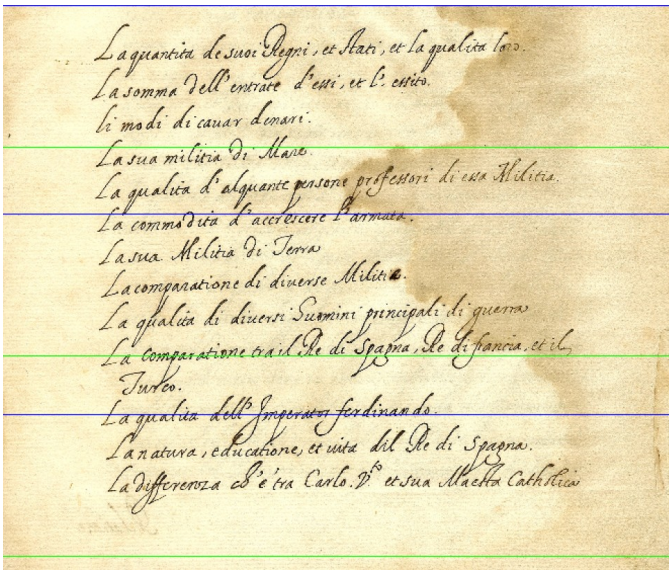


Figura 3.2.4: Texto de la figura 3.2.1 mal segmentado en líneas a causa del ruido.

Una mejora para el problema de diferentes tonalidades de fondo de la imagen es realizar un tratamiento previo de los datos de la proyección horizontal de la imagen, el tratamiento consiste en centrar sobre el origen la señal respecto al eje “Suma por columnas de píxeles” , esto se consigue sumando todos los valores de proyección horizontal de la imagen, dividirlos entre el número de filas y restar la media obtenida a la proyección horizontal de la imagen.

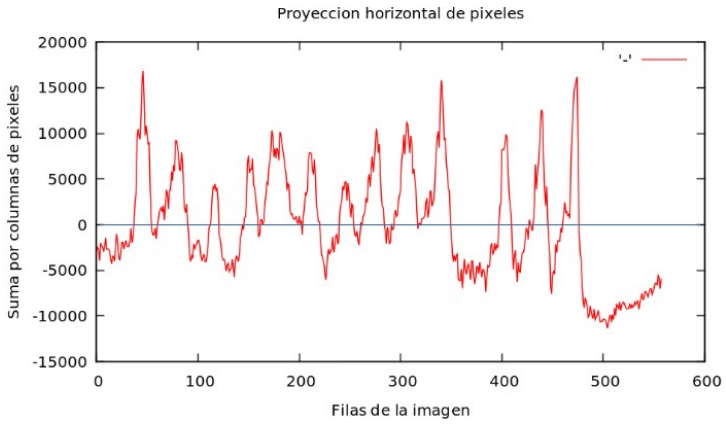


Figura 3.2.5: Proyección horizontal centrada de la figura 3.2.2.

Esta proyección separa la señal en dos señales, una señal con valores positivos y una señal con valores negativos, al aplicarle la DFT la señal con valores positivos tendrá más peso respecto a la señal con valores negativos y esto reduce el ruido en el espectro de frecuencias resultante.
Lo comprobamos visualizando la gráfica de la DFT resultante.

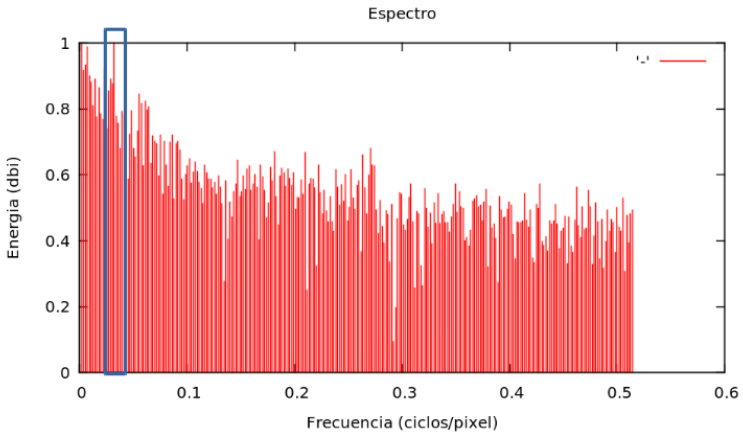


Figura 3.2.6: Espectro de la proyección horizontal de la figura 3.2.5.

Comprobamos que ahora nuestro sistema si que segmenta correctamente las líneas del texto:

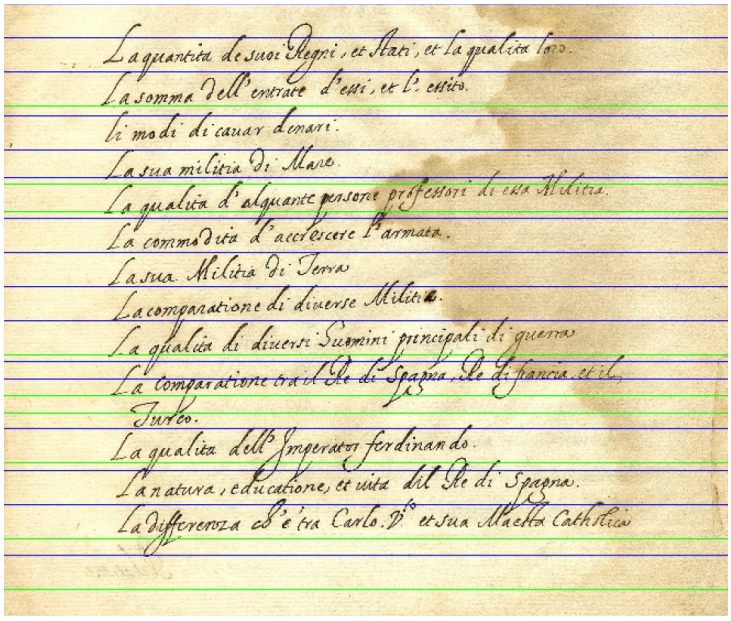


Figura 3.2.7: Texto de la figura 3.2.1 bien segmentado en líneas.

La señal de la proyección horizontal, como hemos visto, suele tener mucho ruido, este ruido provoca que la señal sea bastante irregular y esto puede provocar que al reposicionar la señal no se quede gran parte de la señal como parte positiva, es decir, que perdemos información cuando le pasamos la señal a la DFT, vemos un ejemplo a continuación.

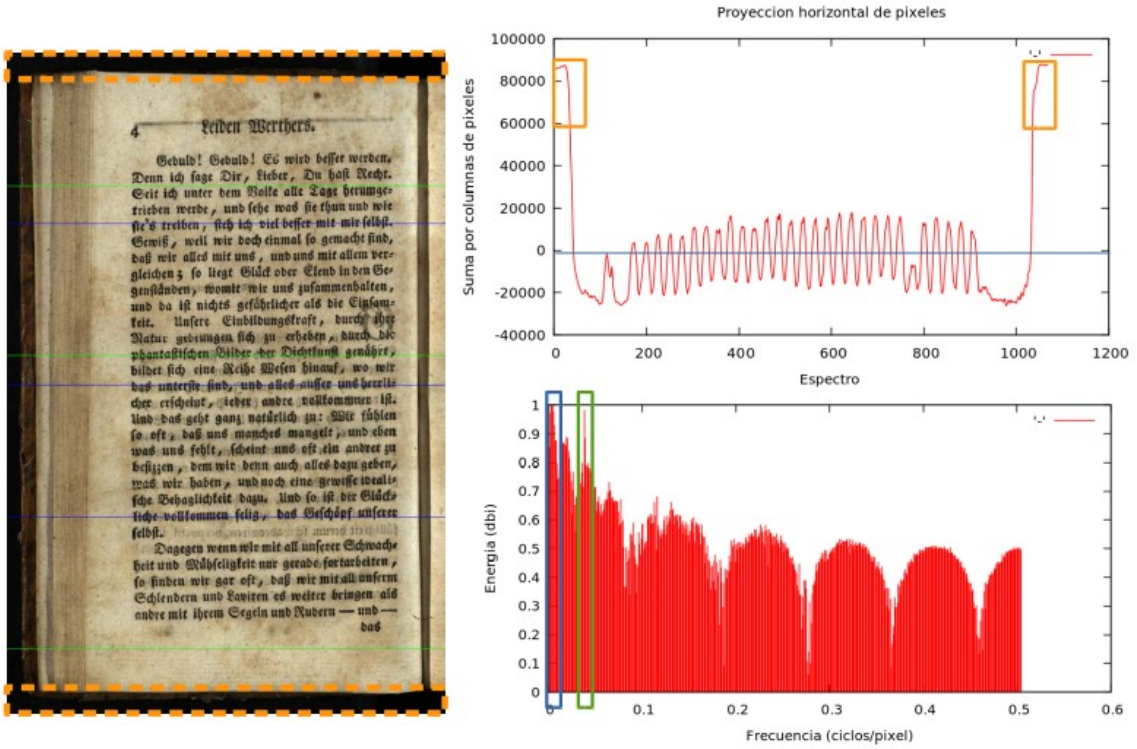


Figura 3.2.8: Texto manuscrito con ruido en los márgenes superior e inferior.

Los rectángulos señalados en la proyección horizontal hacen referencia al ruido generado por los márgenes, este ruido afecta principalmente a la media con la que hemos trabajado anteriormente ya que son valores extremos (anómalos) que provocan que la media de la señal se desplace. Si recortamos la imagen y dejamos el texto manuscrito sin esos márgenes observamos que nuestro sistema funciona correctamente.

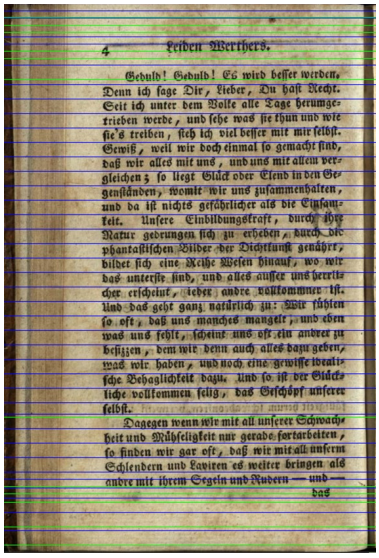


Figura 3.2.9: Texto manuscrito sin ruido en los márgenes superior e inferior.

Si las tonalidades de color de fondo afectan a gran parte del texto y lo hacen de tal forma que al realizar las proyecciones horizontales aparece tanta variación de valores, el sistema no puede separar el texto en líneas y no funciona correctamente, este problema es un problema límite dentro de nuestro sistema que lo debemos de tener en cuenta.

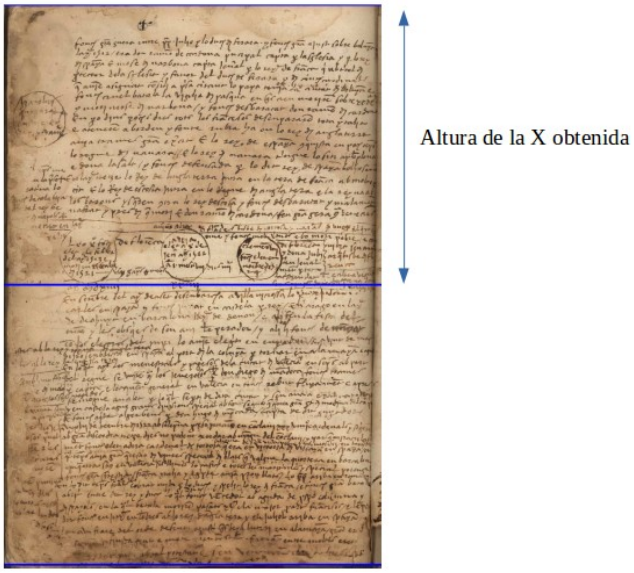


Figura 3.2.10: Limitación ante ruido de las diferentes tonalidades de fondo del texto.

Las conclusiones que podemos obtener son que las variaciones extremas de color en las tonalidades de la imagen influyen negativamente sobre nuestro sistema, esto provoca que el espectro de frecuencias contenga ruido y la frecuencia fundamental no sea la esperada. Los cambios de tonalidades de color suelen aparecer por zonas del texto, trabajar con partes de la imagen en vez de con la imagen completa nos permite reducir el ruido y poder obtener mejor información en el espectro de frecuencias.

La solución que proporcionamos para el problema de la desviación en la alineación horizontal de las líneas consiste en partir verticalmente la imagen de entrada en partes, observaremos que este procedimiento mejora considerablemente el problema que hemos visto respecto al ruido que aparece en la proyección horizontal por las tonalidades de color de la imagen.

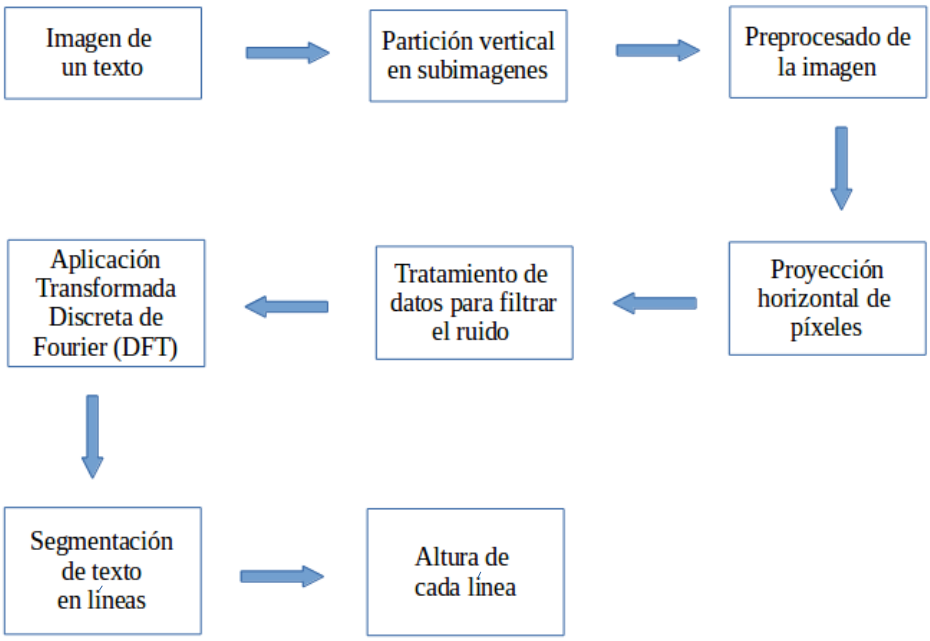


Figura 3.2.11: Esquema con módulo de partición vertical en subimagenes.

Cuando el texto contiene líneas que no están alineadas horizontalmente ni son paralelas, al realizar las proyecciones horizontales no podemos distinguir la separación de líneas en el texto. La solución a este problema consiste en segmentar la imagen verticalmente obteniendo un conjunto de imágenes que forman la imagen original, cada imagen de este conjunto es una entrada a nuestro sistema, el cual hace el mismo proceso que hemos visto hasta ahora, por lo tanto nuestro sistema devuelve ahora un conjunto de alturas de cada línea para cada imagen particionada.
Para comprender que es lo que realmente esta haciendo vamos a ver el resultado de la segmentación del texto en imágenes y la segmentación en líneas de cada imagen resultante.

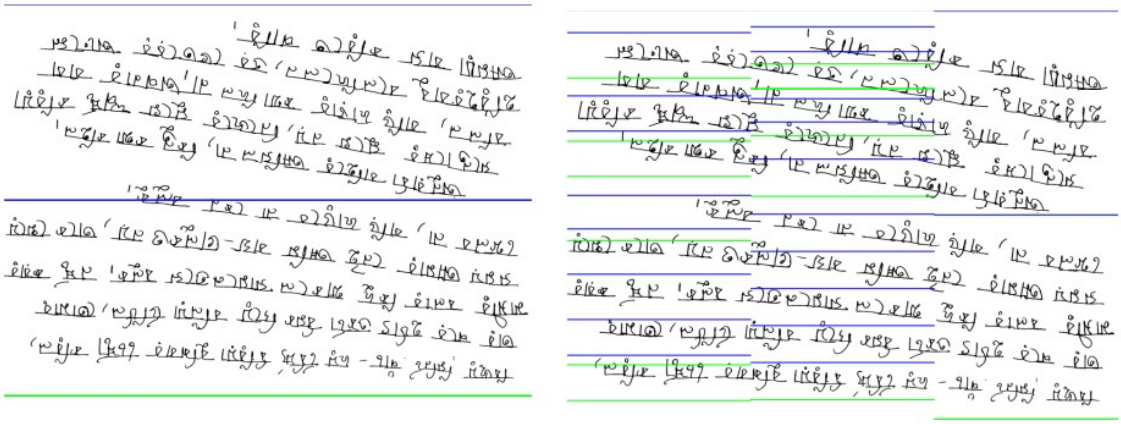


Figura 3.2.12: Izquierda. Partición dos imágenes. Derecha. Partición en tres imágenes.

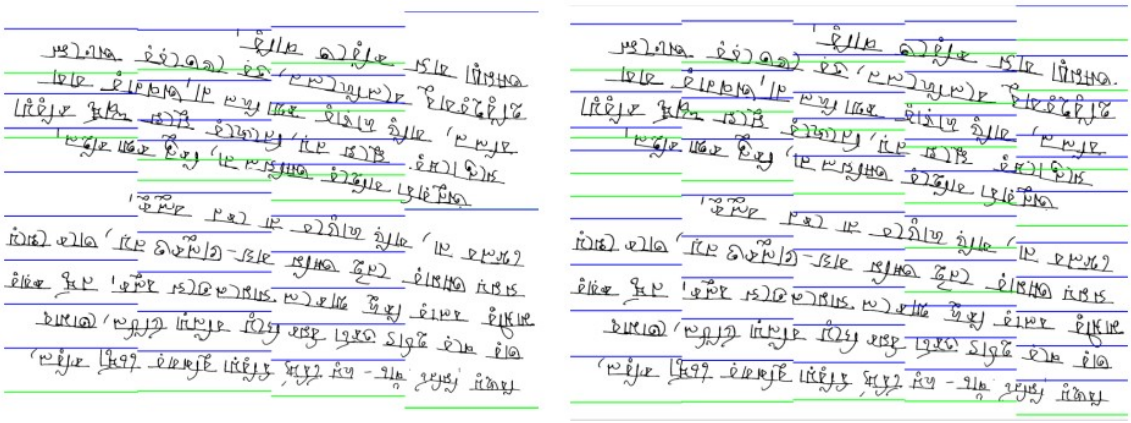


Figura 3.2.13: Izquierda. Partición en cuatro imágenes. Derecha. Partición en cinco imágenes.

La pregunta que nos hacemos es hasta que nivel de segmentación vertical debemos aplicar a la imagen para obtener una aproximación de la altura que consideremos buena. La solución que hemos planteado para este problema ha sido obtener un conjunto de particiones amplio para tener un conjunto de datos relevantes que hemos de tratar posteriormente. Nuestro sistema divide la imagen en 20, 40, 60, …, 100 partes iguales, cada parte en cada división es una entrada a nuestro sistema. Es importante remarcar porque particionamos la imagen mediante un barrido desde 20 hasta 100 partes, el inicio y el final del barrido lo hemos elegido de forma empírica realizando ejecuciones y observando que a partir de 20 particiones el sistema empieza a dar datos relevantes, trabajamos hasta 100 partes porque los datos resultantes son ya bastante buenos y no cargamos en exceso el sistema.

Particionar la imagen de forma excesiva provoca que nuestro sistema no funcione correctamente, en la imagen que ponemos como ejemplo observamos que los caracteres tienen un tamaño muy grande respecto a la resolución de la imagen.

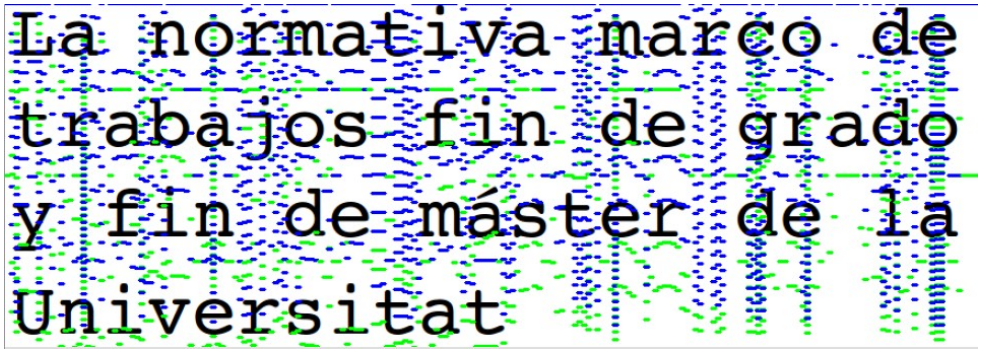


Figura 3.2.14: Partición excesiva de la imagen de un texto.

Observamos que la partición vertical hace que las imágenes particionadas sean muy pequeñas y que se particionen los caracteres en partes, esto provoca que el sistema trabaje con partes del carácter en vez de con el carácter o caracteres completos provocando que la salida del sistema sea errónea. Hemos decidido buscar una solución a este problema parametrizando el sistema, esto provoca que el usuario para este tipo de imágenes tenga que introducir un parámetro en la ejecución que modifique el barrido a un número de particiones más bajo, en este ejemplo el barrido sería como mucho hasta 20 partes.

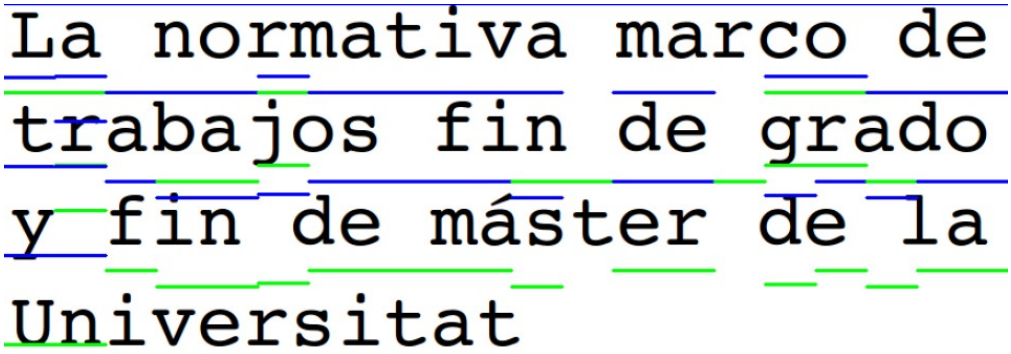


Figura 3.2.15: Partición correcta de la imagen de la figura 3.2.14.

Como hemos visto nuestro sistema devuelve un conjunto de alturas de la *x* que hacen referencia a las líneas de cada parte de la imagen que hemos dividido. Tener este conjunto de alturas como salida del sistema no es una buena solución al problema por el motivo de que obligamos al usuario final que trate los datos para llegar a la conclusión de que altura es la que busca. Esto nos obliga a automatizar nuestro sistema y hacer un tratamiento de datos para que el sistema devuelva una única altura que sea la que más se aproxime a la altura de la *x* del texto de entrada.

Nuestro sistema trabaja con dos vectores globales que almacenan, para cada parte de la imagen dividida, las alturas de la *x* de cada línea obtenidas con su correspondiente frecuencia. Estos dos vectores al finalizar la ejecución son los que tienen toda la información resultante con todas las alturas de la *x* que han aparecido en las divisiones en partes de la imagen y su correspondiente frecuencia de aparición.

Un tratamiento previo de las alturas obtenidas por el sistema antes de ir rellenando los dos vectores globales nos ayuda a descartar alturas que sabemos que no nos interesan y que hacen que la talla de los vectores globales disminuya considerablemente, esto lo conseguimos realizando una ponderación de las líneas en cada entrada de nuestro sistema. Para una mejor comprensión vamos a explicar esta ponderación aplicada sobre todo el texto, pero nuestro sistema realmente la aplica sobre cada parte de cada división de la imagen de entrada.

Cuando fragmentamos el texto en líneas no todas las líneas nos aportan la información que deseamos, nos encontramos con líneas que están vacías y que no

nos aportan nada de información como son los márgenes del texto, la separación entre párrafos, etc. Por otro lado también tenemos líneas más importantes que otras que son aquellas que tienen más letras, es decir, una línea que tenga diez palabras nos va a aportar más información que una línea que tenga dos. La idea principal es quedarnos con las líneas que más aportación nos proporcionen y eso lo podemos conseguir ponderando las proyecciones horizontales de cada línea, esta ponderación nos proporciona un peso que nos indica la confianza de la línea.

La normalización consiste en dividir cada proyección horizontal entre la suma de todas las proyecciones horizontales de la imagen de entrada. Una vez tenemos todas las proyecciones con un peso que nos indica la confianza que tiene esa fila de píxeles, aplicamos dicho peso a la ROI que corresponde a una línea del texto. Lo que buscamos es obtener un valor que nos diga la confianza de cada línea ya que cada línea nos proporciona una altura de la *x*, para ello lo que hacemos es sumar todas las normalizaciones que corresponden a cada fila de la ROI. Este valor que hemos obtenido nos permite conocer la confianza que tiene una línea, los valores más altos nos indican las líneas con mayor confianza.

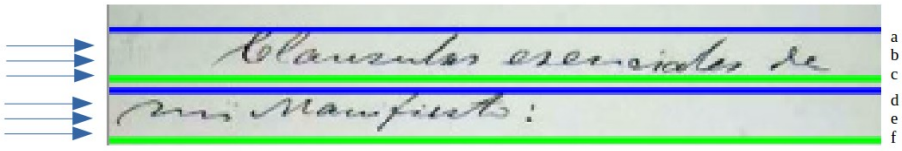


Figura 3.2.16: Texto con dos líneas para ilustrar la ponderación de línea en cada ROI.

Cada letra corresponde a la proyección horizontal de cada fila de la imagen, la ponderación obtenida en la figura 3.2.16 sería:

suma_p = a+b+c+d+e+f
Ponderación_ROI_1 = (a/suma_p+b/suma_p+c/suma_p)
Ponderación_ROI_2 = (d/suma_p+e/suma_p+f/suma_p)

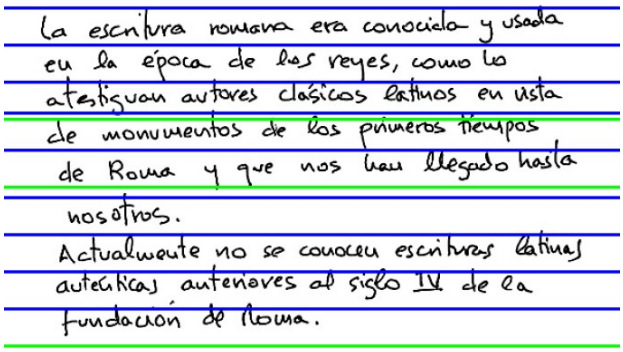


Figura 3.2.17: Texto que contiene dos líneas con una ponderación más baja al resto de líneas.

En la figura 3.2.17 tenemos un texto en el que la ROI que contiene la palabra “nosotros” y la ROI que contiene la frase “fundación de Roma” tienen menos caracteres que las demás ROI por lo tanto la ponderación resultante debe ser menor. El resultado de las ponderaciones de cada ROI lo vemos en la siguiente tabla.

ROI del texto	**Altura de la *x* obtenida**	**Ponderación**
1	12 píxeles	0.131756
2	11 píxeles	0.108513
3	12 píxeles	0.140415
4	15 píxeles	0.128038
5	13 píxeles	0.121482
6	12 píxeles	0.033407
7	13 píxeles	0.145452
8	13 píxeles	0.129544
9	10 píxeles	0.064853

Tabla 3.2.1: Resultado de la ejecución de la imagen de la figura 3.2.17 con ponderaciones de línea.

El criterio que hemos escogido para quedarnos con las mejores líneas del texto ha sido ordenar las líneas del texto de mayor a menor ponderación y coger la mitad del total de líneas que tiene el texto, en este ejemplo las alturas que escogemos serían las siguientes:

ROI del texto	**Altura de la *x* obtenida**	**Ponderación**
7	13 píxeles	0.145452
3	12 píxeles	0.140415
1	12 píxeles	0.131756
8	13 píxeles	0.129544

Tabla 3.2.2: Alturas que nos interesan basándonos en la ponderación de líneas de la tabla 3.2.1.

Los vectores globales se actualizarían en este ejemplo añadiendo a la altura 12 y 13 píxeles dos repeticiones más. De esta forma no tenemos valores de alturas repetidos en los vectores globales y tenemos todas las alturas buenas a tener en cuenta con sus respectivas frecuencias de aparición.

Es interesante observar que la solución que podemos pensar como más racional como salida de nuestro sistema es la altura de la *x* que más se ha repetido en todas las divisiones que hemos realizado, esa altura es una aproximación y puede ser la salida de nuestro sistema pero podemos aproximar aún mejor nuestra salida, para ello no tenemos en cuenta únicamente la altura de la *x* más frecuente, es decir, hemos de tener en cuenta para cada altura sus alturas más próximas (alturas vecinas) con sus respectivas frecuencias.

Utilizamos una moda con contexto, como ya sabemos la moda en estadística se define como el valor con mayor frecuencia en una distribución de datos, una moda con contexto tiene la misma definición pero además tiene en cuenta la frecuencia de sus datos vecinos, esto nos permite tener en cuenta un conjunto de alturas próximas en vez de la altura más frecuente, veamos lo con un ejemplo:

Altura de la *x*	1	2	3	4	5	6
Frecuencia	0	8	0	4	5	5

Tabla 3.2.3: Ejemplo para explicar la moda con contexto.

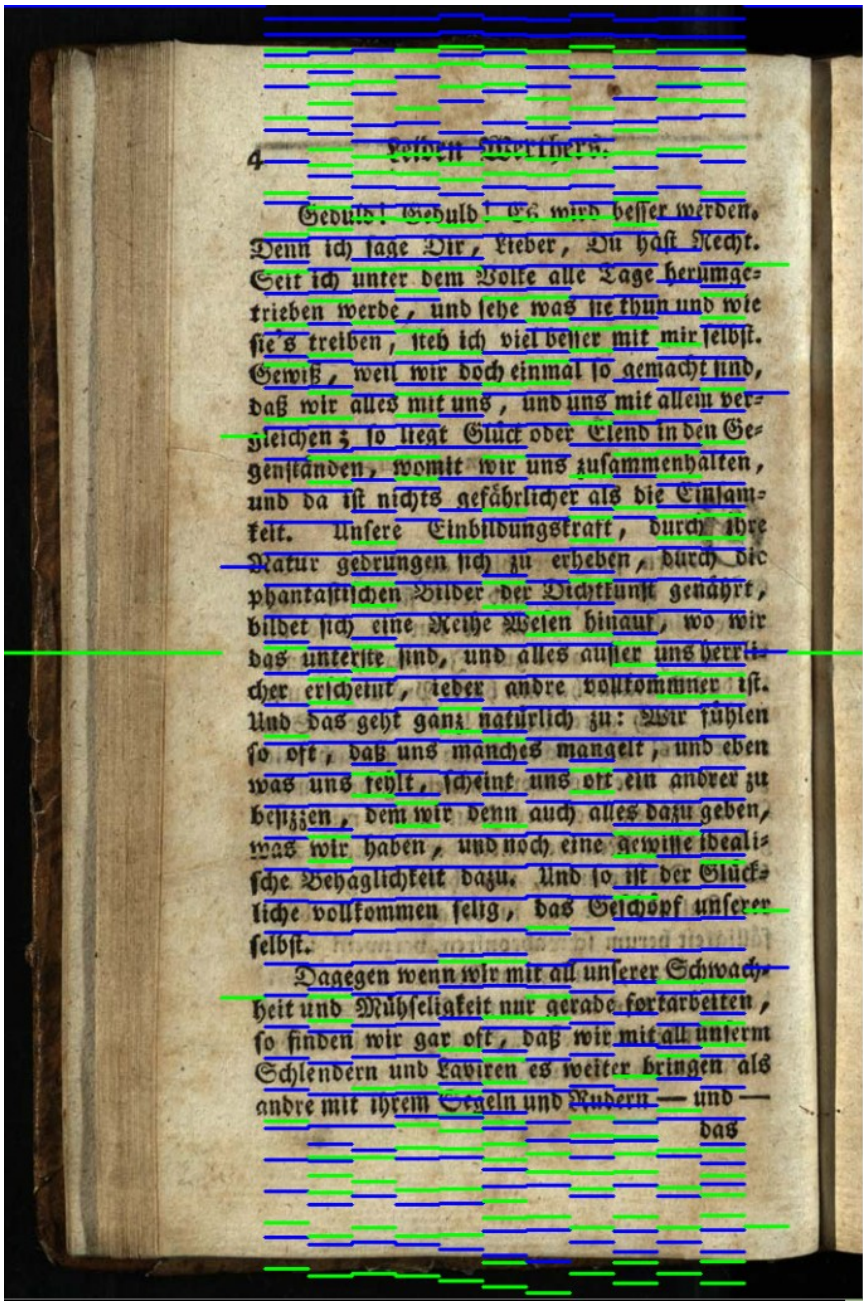
En la tabla 3.2.3 la altura de la *x* más frecuente es 2 píxeles que se ha repetido en 8 líneas del texto, pero en cambio la altura con valor 2 píxeles esta aislada ya que sus vecinos no se repiten, vemos que en las alturas 4, 5 y 6 píxeles tenemos más información aunque ninguno sea la moda. La moda con contexto que utilizamos se basa en una ventana de 3 datos, el cálculo para el ejemplo anterior se realiza de la siguiente forma:

Altura de la *x*	1	2	3	4	5	6
Frecuencia	0	8	0	4	5	5
Frecuencia contexto	(0+0+8)/3 = 2.67	(0+8+0)/3 = 2.67	(8+0+4)/3 = 4	(0+4+5)/3 = 3	(4+5+5)/3 = 4.67	(5+5+0)/3 = 3.33

Tabla 3.2.4: Ejemplo de la tabla 3.2.3 con moda con contexto calculada.

El valor que buscamos es la altura cuya frecuencia con contexto sea máxima, en este ejemplo obtendríamos una altura de la *x* de 5 píxeles con una frecuencia con contexto de 4.67 repeticiones.

Particionar verticalmente la imagen también mejora el ruido que aparece por las diferentes tonalidades de color de fondo de una imagen, en la figura 3.2.8 hemos visto que el ruido no nos permitía segmentar el texto en líneas, a continuación vamos a ver las particiones verticales de la imagen que hace nuestro sistema sobre la figura 3.2.8. Observamos que al particionar la imagen verticalmente nuestro sistema si que segmenta el texto en líneas correctamente.



Geduld! Geduld! es wird besser werden. Denn ich sage Dir, Lieber, Du hast Recht. Seit ich unter dem Volke alle Tage herumgetrieben werde, und sehe was sie thun und wie sie's treiben, steh ich viel besser mit mir selbst. Gewiß, weil wir doch einmal so gemacht sind, daß wir alles mit uns, und uns mit allem vergleichen; so liegt Glück oder Elend in den Gegenständen, womit wir uns zusammenhalten, und da ist nichts gefährlicher als die Einsamkeit. Unsere Einbildungskraft, durch ihre Natur gedrungen sich zu erheben, durch die phantastischen Bilder der Dichtkunst genährt, bildet sich eine Reihe Wesen hinaus, wo wir das unterste sind, und alles ausser uns herrlicher erscheint, jeder andre vollkommner ist. Und das geht ganz natürlich zu: Wir fühlen so oft, daß uns manches mangelt, und eben was uns fehlt, scheint uns oft ein andrer zu besitzen, dem wir denn auch alles dazu geben, was wir haben, und noch eine gewisse idealische Behaglichkeit dazu. Und so ist der Glückliche vollkommen selig, das Geschöpf unserer selbst.

Dagegen wenn wir mit all unserer Schwachheit und Mühseligkeit nur gerade fortarbeiten, so finden wir gar oft, daß wir mit all unserm Schlendern und Laviren es weiter bringen als andre mit ihrem Segeln und Rudern — und — das

Figura 3.2.18: Texto de la figura 3.2.8 con partición en imágenes de imagen.columnas/20.

La figura 3.2.18 nos permite observar que hace nuestro sistema en la primera partición que realizamos, la primera partición como hemos explicado anteriormente consiste en dividir la imagen en 20 partes, observamos que la DFT con la posterior aplicación del algoritmo de segmentación en líneas nos permite dividir en bloques la imagen, en este ejemplo tenemos tres bloques, esto nos permite separar el texto trabajando únicamente con el bloque central, este punto es importante pero no es un tema que tratemos en este trabajo.

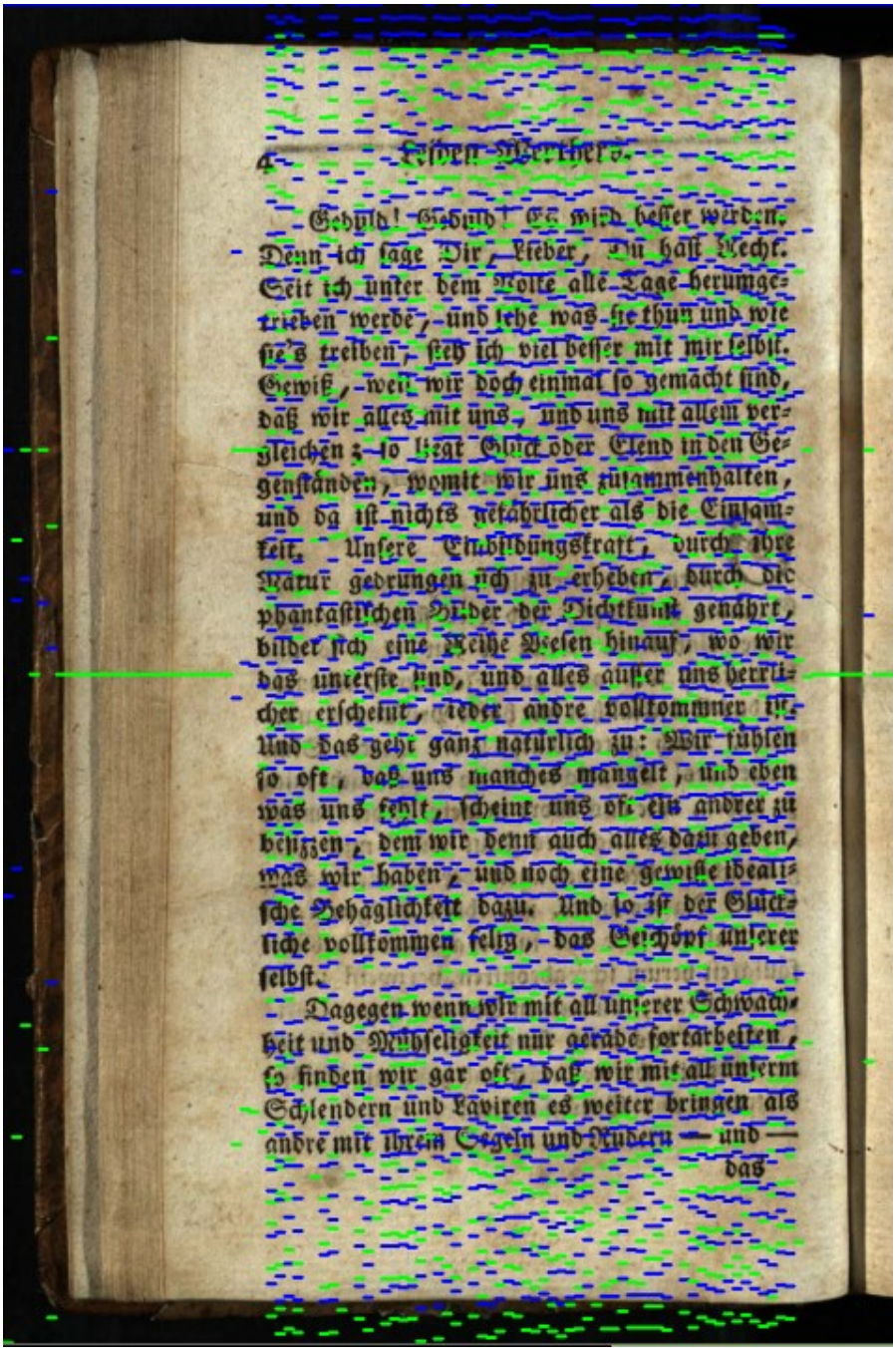


Figura 3.2.19: Texto de la ilustración 3.2.8 con partición en imágenes de imagen.columnas/100.

La figura 3.2.19 representa la última partición que realiza nuestro sistema que consiste en particionar la imagen en 100 partes, vemos que empieza a generar alturas erróneas en los márgenes de la imagen, estas alturas no nos aportan información coherente.

El problema de estas alturas erróneas de los márgenes lo hemos resuelto aplicando una proyección vertical a la imagen, una proyección vertical consiste en sumar todas las filas de cada columna obteniendo un vector de proyección vertical de talla el número de columnas, dicho vector lo hemos normalizado y hemos obtenido una ponderación del mismo modo que en el apartado de la ponderación de líneas. Cada vez que particionamos la imagen verticalmente hemos de conocer como es de buena esa ROI vertical que corresponde a la parte de la imagen.

El criterio que hemos escogido para determinar que ROI verticales nos proporcionan buena información y cuales no ha sido descartar el 30 % de ROI con puntuación más baja, esto nos elimina las ROI correspondientes a los márgenes y ROI con menos líneas de texto.

Figura 3.2.20: Texto de ilustración para la ponderación de la partición vertical de la imagen.

Parte	Ponderación
1	0.036792
2	0.255810
3	0.227612
4	0.242246
5	0.213779
6	0.023760

Tabla 3.2.5: Puntuaciones obtenidas para cada parte de la segmentación vertical de la figura 3.2.20.

El 30% de 6 partes es 1.8 partes, al redondear nuestro sistema elimina 2 partes de la segmentación vertical de la imagen, las dos partes que elimina como ya hemos explicado son aquellas con menor puntuación.

Parte	Ponderación
2	0.255810
4	0.242246
3	0.227612
5	0.213779

Tabla 3.2.6: Partes de la segmentación vertical con las que trabaja nuestro sistema.

Las dos partes que no tiene en cuenta nuestro sistema en la figura 3.2.20 son la parte 1 y la parte 6 que corresponden a los márgenes laterales que no contienen texto.

Es interesante mostrar resultados de ejecuciones de nuestro sistema sobre textos manuscritos. A continuación vamos a mostrar los resultados para un texto manuscrito con variación de tonalidades de fondo y para un texto con desviación respecto a la alineación horizontal de sus líneas.

Figura 3.2.21: Texto manuscrito con variaciones de tonalidades de fondo.

Altura X (Píxeles)	Frecuencia	Frecuencia con contexto
1	2	3.00
2	7	4.33
3	4	12.33
...	...	...
10	67	67.33
11	79	157.00
12	325	509.33
13	1124	701.67
14	656	654.33
15	183	346.66
16	201	180.67
...	...	...
527	1	0.67
528	1	0.67
530	1	0.33

Tabla 3.2.7: Resultado de la ejecución de nuestro sistema para el texto de la figura 3.2.21.

Los puntos suspensivos indican que existen alturas entre los valores que mostramos en la tabla, observamos que nuestro sistema para esta ejecución nos devolverá como salida una altura de la *x* de 13 píxeles.

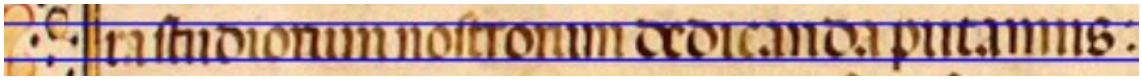


Figura 3.2.22: Segmentación de texto de la figura 3.2.21 con una altura de 13 píxeles.

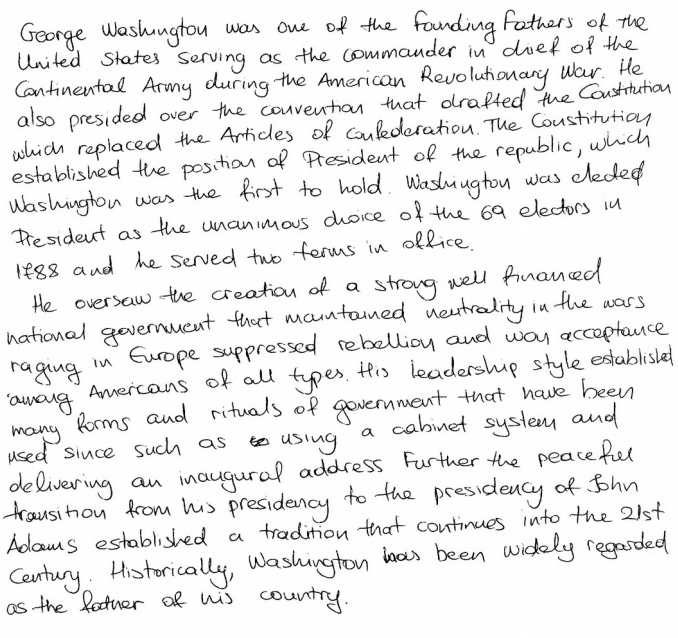


Figura 3.2.23: Texto manuscrito con desviación respecto a la alineación horizontal de sus líneas.

Altura X (Píxeles)	Frecuencia	Frecuencia con contexto
2	2	2.67
3	6	11.33
4	26	26.67
...	...	...
26	74	82.00
27	120	111.66
28	141	138.00
29	153	154.33
30	169	152.33
31	135	142.67
32	124	131.00
...	...	...
515	1	0.33
545	1	0.33
573	1	0.33

Tabla 3.2.8: Resultado de la ejecución de nuestro sistema para la figura 3.2.23.

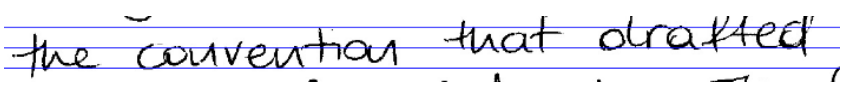


Figura 3.2.24: Segmentación de texto de la figura 3.2.23 con una altura de 29 píxeles.

4. Experimentación

4.1 Corpus

El corpus que hemos utilizado para realizar la experimentación es del *ICDAR 2013*[1] *handwriting segmentation contest*, ICDAR es el acrónimo en inglés de la conferencia internacional de reconocimiento y análisis de documentos. Es importante tener en cuenta que nuestro sistema no necesita de una fase de aprendizaje, por este motivo hemos utilizado solamente el subconjunto de test del corpus compuesto por 150 textos.

Como no disponíamos de la altura de la *x* de cada texto de la imagen del corpus la hemos obtenido nosotros, para ello hemos utilizado nuestro sistema para cada imagen del corpus. A continuación hemos revisado a mano las alturas de la *x* obtenidas y en los casos que ha sido necesario las hemos corregido.

Hemos escalado diez veces cada imagen del corpus, para realizar el escalado hemos obtenido aleatoriamente un valor de escala entre 10% y 60%, un valor de escala superior al 60% en el caso de aumentar la imagen provoca que el peso de las imágenes sea muy grande. Cuando escalamos una imagen podemos aumentar su tamaño o disminuirlo, esta opción de aumentar o disminuir la hemos obtenido también aleatoriamente.

Para cada imagen escalada hemos calculado la altura de la *x* de referencia y su altura de la *x* estimada. La altura de la *x* de referencia la obtenemos respecto a las filas y columnas de la imagen original mediante el valor de escalado, la opción de escalado y la altura de la *x* de la imagen original, esta altura de la *x* de referencia es la altura de la *x* que consideramos que tiene la imagen de texto escalada. La altura de la *x* estimada es el resultado que nos proporciona nuestro sistema para cada imagen escalada.

4.2 Medidas

Las medidas de la experimentación se obtienen a partir de la altura de la *x* de referencia y la altura de la *x* estimada que hemos almacenado a partir de cada imagen escalada.

1. Nikolaos Stamatpoulos, Basilis Gatos, Georgios Louloudis, Umapada Pal, and Alireza Alaei. Icdar2013 handwriting segmentation contest. In 12th International Conference on Document Analysis and Recognition., pages 1434–1406, 2013.

Para interpretar los resultado obtenidos nos podemos basar en realizar una diferencia en valor absoluto entre la altura de referencia y estimada de la *x* y contar cuantos resultados no están dentro de un margen de píxeles que nos aporta información acerca de la tolerancia en valor absoluto, lo vemos en el siguiente ejemplo:

Altura de la *x* referencia (Píxeles)	**Altura de la *x* estimada (Píxeles)**	**| Diferencia | (Píxeles)**
17	19	2
36	35	1
24	26	2
40	43	3
18	18	0
9	11	2
20	21	1

Tabla 4.2.1: Ejemplo artificial de medias obtenidas.

Margen (Píxeles)	**N.º de imágenes**	**Error obtenido**
> 1	3	(3/7)*100 = 42.86 %
> 2	1	(1/7)*100 = 14.26 %
> 3	0	(0/7)*100 = 0.00 %

Tabla 4.2.2: Resultados mediante márgenes del ejemplo de la tabla 4.2.1.

Los resultados obtenidos nos aportan un error absoluto respecto al margen de píxeles, este resultado no tiene en cuenta la relación entre la diferencia obtenida y la altura de la *x* de referencia, es decir, una diferencia de 2 píxeles con una altura de referencia de 50 píxeles tiene menos error que una diferencia de 2 píxeles de una altura de referencia de 20 píxeles, por este motivo es importante conocer el error relativo obtenido.

error_relativo = $\frac{1}{n} \cdot \sum_{i=1}^{n} \frac{\left| alturaXreal_i - alturaXestimada_i \right|}{alturaXreal_i}$, siendo n = número de datos

4.3 Resultados

Error absoluto

Para nuestra experimentación hemos utilizado márgenes de tolerancia de 1,2,3,4 y 5 píxeles, y como tenemos un corpus de 150 imágenes y escalamos aleatoriamente cada imagen 10 veces disponemos de 1500 imágenes en total, vemos los resultados obtenidos en la siguiente tabla.

Margen (Píxeles)	N.º de imágenes	Error obtenido
> 1	437	29.13 %
> 2	255	17.00 %
> 3	159	10.60 %
> 4	123	8.20 %
> 5	94	6.27 %

Tabla 4.3.1: Error absoluto clasificado por márgenes obtenido en la experimentación.

Error relativo

Error relativo	7.91 %

Tabla 4.3.2: Error relativo obtenido en la experimentación.

6. Conclusiones y trabajos futuros

En este trabajo hemos implementado una herramienta que obtiene una estimación de la altura de la *x,* de los caracteres de una página. Para ello se ha utilizado la transformada de Fourier para obtener el paso de línea, esta información nos ha permitido diseñar un método de segmentación sencillo basado en heurísticos simples. A partir de las líneas hemos obtenido una estimación de la altura mediante otro algoritmo sencillo. Hemos realizado ponderaciones de líneas y ponderaciones de partes verticales segmentadas de la imagen que nos permite elegir partes de texto que consideramos con información relevante. Como aproximación a la altura de la *x* para toda la página se ha optado por la moda con contexto.

Se ha explorado el comportamiento de la técnica desarrollada en textos difíciles, con mucho ruido y con lineas de texto no paralelas. Lo usual en los sistemas de reconocimiento de texto manuscrito es primero preprocesar la imagen para eliminar el ruido, corregir el skew y etiquetar el layout, cosa que simplifica mucho el resto de procesos.

Con la utilización del subconjunto de test de las imágenes del corpus *ICDAR 2013 handwriting segmentation contest* hemos realizado una experimentación, al no disponer de *ground truth* adecuado, lo hemos creado nosotros mismos. Los resultados que hemos proporcionado para demostrar la calidad de nuestro sistema han sido el error absoluto y el error relativo siendo este último el que pensamos que es de mayor interés.

En un futuro próximo vamos a medir el algoritmo de segmentación y a utilizar la información de paso de linea para intentar mejorar algoritmos de segmentación de líneas del estado del arte. También vamos a explorar esta técnica para realizar *layout analysis* puesto que en este trabajo se ha demostrado su capacidad para detectar bloques de texto.

7. Bibliografía

1. Open Source Computer Vision (OpenCV): http://opencv.org/
2. API C++. http://www.cplusplus.com/
3. GNUPLOT: http://www.gnuplot.info/
4. Conexión de GNUPLOT con C++: http://www.stahlke.org/dan/gnuplot-iostream/
5. Juan-Pablo Cáceres, S Aditiva, A Espectral – 2007. Teoria y análisis espectral de la transformada de Fourier. CCRMA Stanford University
6. Apuntes de tratamiento de señales digitales de la asignatura de cuarto Instrumentación Industrial (IIN), Grado en Ingeniería Informática, UPV. Profesor Dr. Pascual Pérez Blasco.
7. Apuntes de espacios de color y dominio frecuencial en OpenCV de la asignaturas de procesamiento audiovisual, Departamento de Informática y Sistemas, Universidad de Murcia. Profesor Dr. Ginés García Mateos.
8. Implementación de la transformada discreta de Fourier en OpenCV. http://docs.opencv.org/2.4/doc/tutorials/core/discrete_fourier_transform/discrete_fourier_transform.html
9. Corpus del contest HTR ICDAR 2013: Nikolaos Stamatpoulos, Basilis Gatos, Georgios Louloudis, Umapada Pal, and Alireza Alaei. Icdar2013 handwriting segmentation contest. In 12th International Conference on Document Analysis and Recognition., pages 1434–1406, 2013.

Índice de ilustraciones

Índice de tablas