Detección del Orden de Lectura de un documento en base a Inteligencia Computacional.

Fraga López, Daniel Sebastián

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Larco Bravo, Julio Cesar

19 de marzo de 2021

Curiginal

Document Information

Analyzed document	DETECCIÓN DEL ORDEN DE LECTURA DE UN DOCUMENTO EN BASE A INTELIGENCIA COMPUTACIONAL .pdf (D99002765)
Submitted	3/20/2021 5:46:00 AM
Submitted by	
Submitter email	jclarco@espe.edu.ec
Similarity	1%
Analysis address	jclarco.espe@analysis.urkund.com

Sources included in the report

W	URL: https://www.specsavers.es/ayuda-y-preguntas/%C2%BFqu%C3%A9-es-deficiencia-visual Fetched: 3/20/2021 5:47:00 AM	1
W	URL: https://repositorio.uam.es/bitstream/handle/10486/679289/Romera_Vicente_Nerea_tfg. ... Fetched: 3/6/2020 2:45:10 PM	1
W	URL: https://acrobat.adobe.com/es/es/acrobat/about-adobe-pdf.html Fetched: 3/20/2021 5:47:00 AM	1
W	URL: https://es.mathworks.com/help/images/ref/imclose.html Fetched: 3/20/2021 5:47:00 AM	1
W	URL: https://es.wikihow.com/calcular-datos-at%C3%ADpicos Fetched: 3/20/2021 5:47:00 AM	1
W	URL: https://upcommons.upc.edu/bitstream/handle/2117/78924/TFG_thesis_Antea.pdf Fetched: 1/25/2020 7:48:24 AM	2
W	URL: https://www.who.int/es/news-room/detail/08-10-2019-who-launches-first-world-report ... Fetched: 3/20/2021 5:47:00 AM	1

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Detección del Orden de Lectura de un documento en base a Inteligencia Computacional** " fue realizado por el señor **Fraga López , Daniel Sebastián** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí,19 de marzo 2021

Firmado electrónicamente por:
JULIO CESAR
LARCO BRAVO

Firma:

.....

Larco Bravo, Julio Cesar

C.C.:1710638808

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

RESPONSABILIDAD DE AUTORÍA

Yo, **Fraga López, Daniel Sebastián**, con cédula de ciudadanía n° 1003955950, declaro que el contenido, ideas y criterios del trabajo de titulación: **Detección del Orden de Lectura de un documento en base a Inteligencia Computacional** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 19 marzo de 2021

Firma

.....

Fraga López, Daniel Sebastián

C.C.:1003955950

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN DE PUBLICACIÓN

Yo **Fraga López, Daniel Sebastián,** con cédula de ciudadanía n° 1003955950, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Detección del Orden de Lectura de un documento en base a Inteligencia Computacional** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 19 marzo de 2021

Firma

.....

Fraga López, Daniel Sebastián

C.C.: 1003955950

Dedicatoria

Dedico este pequeño logro principalmente a mi Padre Ernesto que es el que inculcó en mi todos los valores que hoy me definen como persona, a mi Madre Rosita que con su amor incondicional me enseñó a no rendirme a pesar de las necesidades, a mis dos Hermanos Bayronn y Pame que fueron un pilar fundamental en mi crecimiento, a mi Mateo que es como mi hermano menor y me enseño la responsabilidad y que a pesar de no tener tiempo, si lo necesito sé que puedo contar con él para lo que necesite, a mi enamorada Andrea que con su compañía hizo que la soledad de vivir lejos de casa sea más llevadera y a mi futuro hijo, que a pesar de no conocerlo trataré de ser el mejor ejemplo a seguir, A mi amigo Bolo y su Familia que a pesar de ser un total desconocido me abrieron las puertas de su casa para aceptarme como uno más de ellos y a todos mis compañeros y amigos con los que conviví toda mi vida universitaria.

Daniel Sebastián Fraga López

Agradecimiento

Agradezco a mi tutor, Ing. Julio Larco por su paciencia, amabilidad y por guiarme por este camino investigativo al compartir sus conocimientos para el desarrollar este proyecto.

A mi Familia, que a pesar de encontrarnos lejos nunca dejaron que me sintiera solo y me motivaron a seguir adelante a pesar de que a veces quería dejar todo y regresar con ellos.

A mis docentes universitarios que con sus conocimientos forjaron el camino a seguir para que llegue este día tan anhelado día.

A mis amigos que con sus consejos y ocurrencias hicieron que la vida lejos de casa no se sintiera tan mal.

Índice de contenidos

Portada 1

Hoja de Resultados de la Herramienta Urkund 2

Certificación Trabajo de Titulación 3

Responsabilidad de Autoría 4

Autorización de Pubicación 5

Dedicatoria 6

Agradecimiento 7

Índice de contenidos 8

Resumen 19

Abstract 20

CAPITULO 1 Introducción 21

- Estado del Arte 21
- Justificación e importancia 27
- Alcance 30
  - Digitalización del documento 30
  - Discriminación de bloques 31
  - Extracción de características de los bloques de información 32
  - Bloque de decisión 33
- Objetivos 34
  - General 34

Específicos 34
CAPITULO 2 Marco teórico 35
Introducción 35
Algoritmos de segmentación de texto en bloques de información 35
Adaptación Multigaussiana 35
Segmentación de documentos de periódicos no estructurados 37
Extracción de texto en imágenes de documentos: resalte el uso de puntos de esquinas 39
Análisis de diseño de documentos: enfoque de la región máxima homogénea 40
Digitalización de documentos históricos mediante análisis de diseño y profunda clasificación de contenido 42
Algoritmo de Suavizado de Longitud de Ejecución (LRSA) 44
Algoritmo de Suavizado de Longitud de Ejecución con OR (LRSO) 46
Métodos de detección de orden de lectura 47
La transformación de árbol 47
Sistema de pizarra 49
Enfoque sintáctico 50
Modelos ocultos de Markov 51
Aprendizaje 51
Fundamentos sobre Inteligencia Computacional 53
Lógica Difusa 53
Determinación del algoritmo a utilizar 62

Digitalización del documento 63
Extracción de bloques de información 65
Extracción de características de los bloques de información 67
Bloque de decisión 70
CAPITULO 3 Desarrollo 71
Introducción 71
Digitalización de documento PDF a PNG mediante Ghostscript 71
Instalación de Ghostscript 73
Conversión de PDF a PNG 79
Segmentación del documento 81
Pre - procesamiento 81
Implementación del algoritmo LRSA 89
Análisis de Componentes Conectados 98
Análisis de las características del BBox de los elementos 99
Redimensionamiento BBox de las líneas de texto 101
Desarrollo del algoritmo de detección de lectura 108
Identificación y redimensionamiento de divisores de página 108
Variables de entrada del Sistema de Control Difuso 113
Variable de salida "***S***" del sistema de Control Difuso 124
Reglas de Control 126
Implementación del Clasificador Difuso para la creación de la base de datos de Orden de Lectura de las imágenes 128
Interfaz Gráfica para presentación de resultados 137

CAPITULO 4 Pruebas y Análisis de Resultados 138

Interfaz gráfica para creación de base de datos para evaluación de resultados 138

Métricas para la evaluación del desempeño del algoritmo implementado 145

Porcentaje de aciertos mediante la comparación estricta del Orden de Lectura 146

Porcentaje de aciertos mediante la comparación de secuencias de 2 elementos del Orden de Lectura 149

Análisis de resultados 152

Porcentaje de aciertos mediante la comparación estricta del Orden de Lectura 152

Porcentaje de aciertos mediante la comparación de secuencia de 2 elementos del Orden de Lectura 156

CAPITULO 5 Conclusiones y Recomendaciones 158

Conclusiones 158

Recomendaciones y Trabajos futuros 161

Referencias 163

ANEXOS 167

ANEXO A: Segmentación y extracción de características 167

ANEXO B: Identificación de Orden de Lectura 178

Índice de Tablas

Tabla 1 *Funciones de membresía*. 57

Tabla 2 *Número de páginas de la base de datos*..... 72

Tabla 3 *Descripción de parámetros de Ghostscript.* 80

Tabla 4 *Información extraída por la función "regionprops"*..... 85

Tabla 5 *Matriz con los datos extraídos de la imagen*. 100

Tabla 6 *Estados de la variable "S".*..... 127

Tabla 7 *Datos de entrada del Clasificador Difuso.* 132

Tabla 8 *Asignación de salida S a las variables de entrada θ y dP*. 132

Tabla 9 *Renombramiento de base de datos.* 146

Tabla 10 *Orden de lectura contenido en Dato 1 para el ejemplo.* 147

Tabla 11 *Orden de lectura contenido en Dato 2 para el ejemplo*. 148

Tabla 12 *Orden de lectura contenido en Dato 3 para el ejemplo.* 150

Tabla 13 *Orden de lectura contenido en Dato 4 para el ejemplo*. 150

Tabla 14 *Resultados de aplicar la comparación de orden estricto.* 153

Tabla 15 *Resultado al aplicar la comparación entre pares de elementos*. 157

Índice de Figuras

Figura 1 *Ejemplo de orden de lectura. a) Texto original. b) Orden de lectura del texto.* 23

Figura 2 *Ejemplo de segmentación a) Documento Original. d) resultado final de la segmentación*. 32

Figura 3 *Ejemplo de segmentación. (a) Imagen original. (b) Segmentación utilizando adaptación Gaussianos. (Laiphangbam, Raghu, & Chakravarthy, 2017)* 37

Figura 4 *Ejemplo de segmentación. (a) Imagen original. (b) Segmentación utilizando el método propuesto. (Naik & Ramegowda, 2017)* 38

Figura 5 *Ejemplo. a) Imagen original. (b) Imagen segmentada por el Método de Puntos de Esquina. (Yadav & Ragot, 2016)* 40

Figura 6 *Ejemplo. a) Imagen original. (b) Imagen segmentada. (Tuan, Khuong, & Nhat, 2018)* 42

Figura 7 *Ejemplo. a) Imagen original segmentada mediante X-Y Cut. (b) Imagen segmentada por el método propuesto. (Corbelli, Baraldi, Grana, & Cucchiara, 2016)* 44

Figura 8 *Ejemplo. a) Imagen original. b) LRSA Horizontal. c) LRSA Vertical. d) Resultado final de la segmentación. (HASNAT, 2007)* 45

Figura 9 *a) Figura original. b) Regiones irregulares que entrega LRSO. (Ferilli, Basile, & Esposito, 2010)* 46

Figura 10 *a) Estructura lógica de árbol. b) Imagen original. c) Orden de lectura por el método de árbol. (Shuichi & Haruo, 1992)* 48

Figura 11 *Ejemplo de documento: a) Imagen a procesar. b) Imagen segmentada. c) Orden de lectura del documento. (Esposito, Malerba, & Semeraro, 1994)* 52

Figura 12 *Ejemplo de reglas de inferencia.* 56

Figura 13 *a) Función de membresía de lógica clásica. b) Función de membresía de lógica difusa.* 58

Figura 14 *Representación de variables lingüísticas* 59

Figura 15. *Representación de funciones de membresía.* 60

Figura 16 *Diagrama de bloques del Sistema de control difuso* 61

Figura 17 *Diagrama de bloques para la determinación del orden de lectura.* 63

Figura 18 *a) Página de un documento en PDF. b) Imagen Binarizada con Ghostscript* 64

Figura 19 *a) Imagen extraída del PDF. b) Imagen binarizada con Ghostscript*. . 65

Figura 20 *Ejemplo de segmentación* 66

Figura 21 *Ejemplo de características a extraer mediante el algoritmo de segmentación.* 67

Figura 22 *Representación de componentes adyacentes* 68

Figura 23 *Representación de componentes en columnas adyacentes* 69

Figura 24 *Ejemplo de regla de Orden de Lectura* 69

Figura 25 *Flujo de conversión de documentos PDF a PNG.* 73

Figura 26 *Opciones de descarga de Ghostscript. (Artifex, s.f.)* 74

Figura 27 *Instalación de Ghostscript.* 75

Figura 28 *Ventana de instalación de Ghostscript.* 75

Figura 29 *Propiedades de Windows ®.* 76

Figura 30 *Configuración avanzada del sistema*. 77

Figura 31 *Propiedades del sistema.* 77

Figura 32 *a) Variables del sistema. b) Variables de entorno. c) Ghostscript® como variable del sistema.* 78

Figura 33 *Llamado a Ghostscript® desde CMD.* 79

Figura 34 *Estructura general de los parámetros admitidos por Ghostscript.* 79

Figura 35 *Comando general para la conversión de PDF a PNG.* 80

Figura 36 *Dimensiones del elemento estructurador.* 82

Figura 37 *a) Imagen original. b) Imagen después de la eliminación de información innecesaria.* 83

Figura 38 *Bounding Box de los caracteres de la imagen.* 84

Figura 39 *Descripción de coordenadas. a) Imágenes. b) BBox.* 86

Figura 40 *Representación de los Límites de Decisión.* 88

Figura 41 *a) BBox con datos de interés para la elaboración del algoritmo LRSA. b) Tamaño horizontal (vh) y tamaño vertical (vv).* 89

Figura 42 *LRSA Horizontal.* 91

Figura 43 *División de la página para su análisis.* 92

Figura 44 *a) Sección de imagen. b) Histograma horizontal de la imagen.* 93

Figura 45 *Histograma con valores limitados.* 94

Figura 46 *Determinación de los puntos máximos del histograma. a) Histograma. b) Zoom de los picos del histograma.* 94

Figura 47 *a) Extracción de picos con C_1<1.3. b) Extracción de picos con C_1>1.3 c) Extracción de picos con C_1=1.3.* 96

Figura 48 *LRSA vertical.* 96

Figura 49 *a) Imagen Original. b) LRSA Vertical. c) LRSA Horizontal. d) AND entre LRSA Horizontal y Vertical*. 97

Figura 50 *a) BBox de la imagen. b) BBox de la imagen normalizada.* 97

Figura 51 *a) Suavizado horizontal de la imagen. b) BBox del suavizado horizontal.*..... 99

Figura 52 *Ejemplo de la información obtenida una imagen de un PDF.*..... 101

Figura 53 *BBox de los elementos después de la eliminación*. 102

Figura 54 *Ejemplo de vecindad. Donde el color azul representa al recuadro a analizar, el color rojo son sus vecinos y el color verde son los demás elementos.*..... 103

Figura 55 *a) Identificación de recuadros que cumplen las condiciones. b) Unión de recuadros*. 104

Figura 56 *a) Identificación de recuadros que cumplen las condiciones. b) Unión de recuadros*. 105

Figura 57 *a) BBox de información Originales. b) BBox de información después del procesamiento.* 106

Figura 58 *Distancia entre los vecinos de los recuadros de información*..... 107

Figura 59 *Análisis de BBox Aislados. a) Imagen original. b) BBox de la página.* 109

Figura 60 *a) BBox de la imagen. b) BBox del redimensionamiento.* 110

Figura 61 *Seccionamiento de página.* 111

Figura 62 *Análisis de las cadenas de texto la página dividida.* 112

Figura 63 *Representación de las variables de entrada del Sistema de Control Difuso.*..... 114

Figura 64 *Universo de discurso para dP.* 115

Figura 65 *Representación gráfica de los Conjuntos Difusos para la variable dP.* 118

Figura 66 *Universo de discurso para la variable θ.* 119

Figura 67 *Valores centrales iniciales de la Funciones de Membresía de θ.* 121

Figura 68 *Valores iniciales de los Conjuntos Difusos de "θ".* 122

Figura 69 *Ejemplo de preferencia de lectura.* 123

Figura 70 *a) Funciones de Membresía de la Variable "θ". b) Superficie de Control generada.* 124

Figura 71 *Funciones de Pertenencia para "S.* 125

Figura 72 *Interfaz inicial de la herramienta "Fuzzy Lógic Designer".* 128

Figura 73 *a) Diagrama de entradas y salidas del Clasificador difuso. Funciones de Membresía de las variables b) dP, c) θ y d) S.* 129

Figura 74 *Reglas de control que definen el sistema.* 130

Figura 75 *Exportación del Clasificador Difuso a un formato con extensión "fis".* 131

Figura 76 *Imagen Original.* 134

Figura 77 *BBox de las líneas de texto de la imagen original.* 135

Figura 78 *Orden de lectura de la imagen.* 136

Figura 79 *Descripción de la interfaz para la visualización del Orden de Lectura en Matlab®.* 137

Figura 80 *Diagrama de la interfaz gráfica para creación de base de datos.* 139

Figura 81 *Interfaz gráfica para la creación de la base de datos.* 140

Figura 82 *Cargar imagen a la interfaz gráfica.* 141

Figura 83 *Visualización de la imagen en la interfaz gráfica*..... 141

Figura 84 *Carga de BBox de la imagen*..... 142

Figura 85 *Agregar orden de lectura.* 143

Figura 86 *Ejemplo de orden de lectura generado mediante la interfaz gráfica.* 144

Figura 87 *Comparación entre Dato 1 y Dato2.*..... 148

Figura 88 *Comparación entre Dato 3 y Dato 4.*..... 151

Figura 89 *Ejemplo de documento sin fórmulas matemáticas. a) BBox del Documento. b) Orden de Lectura.* 154

Figura 90 *Ejemplo de documento con fórmulas matemáticas. a) BBox del documento b) Orden de lectura generado.* 155

Figura 91 *Espacio interlineado. a) Documento sin caracteres matemáticos b) Documento con caracteres matemáticos.*..... 156

Resumen

En la actualidad, los documentos digitales se han vuelto una parte esencial de nuestra vida cotidiana. Actualmente podemos encontrar un documento digital para casi cualquier libro o documento que necesitemos, pero una gran problemática es que muchos de estos documentos digitales son imágenes guardadas en formato PDF, lo que hace muy difícil la extracción de la información de manera digital. Debido a estas y otras problemáticas se han generado sistemas de procesamiento de imágenes que busca recuperar la información almacenada mediante el Reconocimiento Óptico de Caracteres (OCR) pero una gran limitante de este tipo de sistemas es que no puede definir un Orden de Lectura lógico. El Orden de Lectura no es más que la secuencia lógica de interpretación de la información contenida en un documento. Mediante el procesamiento de documentos en formato PDF y procesamiento digital de imágenes, en este proyecto se busca desarrollar un algoritmo capaz de identificar el Orden de Lectura de un documento que permita extraer su información de forma ordenada. Esto se lo realizará en base a Lógica Difusa, la cual se basa el Razonamiento Aproximado y en el uso de Reglas Lingüísticas. Este proyecto puede ser usado para la recuperación de información y así crear bibliotecas virtuales o aplicaciones que sirvan de ayuda a personas con discapacidad visual.

PALABRAS CLAVE:

- **LÓGICA DIFUSA**
- **RAZONAMIENTO APROXIMADO**
- **BOUNDING BOX**
- **ORDEN DE LECTURA**

Abstract

Today, digital documents have become an essential part of our daily lives. Currently we can find a digital document for almost any book or document that we need, but a great problem is that many of these digital documents are images saved in PDF format, which makes it very difficult to extract the information digitally. Due to these and other problems, image processing systems have been generated that seek to recover the information stored by means of Optical Character Recognition (OCR) but a great limitation of this type of system is that it cannot define a logical Reading Order. The Reading Order is nothing more than the logical sequence of interpretation of the information contained in a document. Through the processing of documents in PDF format and digital image processing, this project seeks to develop an algorithm capable of identifying the Reading Order of a document that allows the information to be extracted in an orderly manner. This will be done based on Fuzzy Logic, which is based on Approximate Reasoning and on the use of Linguistic Rules. This project can be used for information retrieval and thus create virtual libraries or applications that help people with visual disabilities.

KEYWORDS:

- **DIFFUSE LOGIC**
- **APPROXIMATE REASONING**
- **BOUNDING BOX**
- **READING ORDER**

CAPITULO 1

Introducción

Estado del Arte

La necesidad de extraer información de documentos para su digitalización y manipulación obligó a que se desarrollen métodos de análisis de estos documentos. Inicialmente, el análisis de documentos de texto se orientó a la extracción de información, para ello se desarrolló varios sistemas de Reconocimiento Óptico de Caracteres (OCR) para la extracción de información; estos algoritmos, pese a cumplir con su objetivo de extracción de información, tenía varias limitantes: se desarrollaron en base a documentos de estructura simple con base en color blanco y negro, lo que los hizo poco confiables ante documentos cuya estructura comprendían imágenes y/o dos o más columnas. Debido a estas limitaciones se buscaron nuevos métodos de análisis de documentos que permitan no solo extraer caracteres de los mismos, sino, transformar el contenido informativo de un documento en un formato electrónico que describe su contenido lógico (Cattoni, Coianiz, Messelodi, & Modena, 1998).

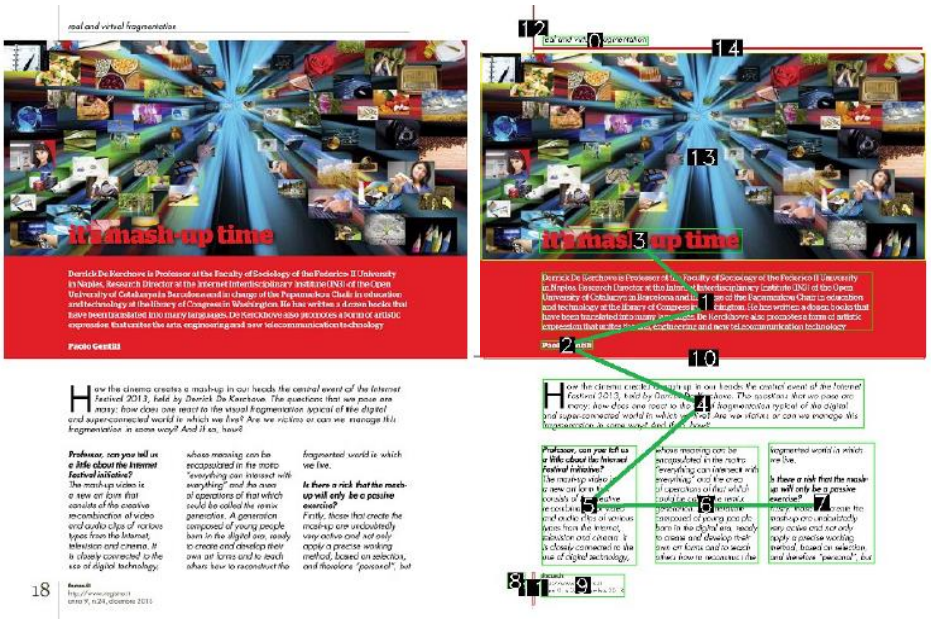
Actualmente la proliferación de documentos electrónicos se ha incrementado de tal forma que es necesario poseer una versión electrónica de casi cualquier libro, uno de los problemas de la digitalización de los libros o textos en general es que se lo hace en forma de imágenes lo que dificulta la extracción de información de los mismos. Debido a estas y otras dificultades se han generado sistemas de procesamiento de imágenes con el fin de detectar

automáticamente los componentes de una página. El objetivo de estos sistemas de procesamiento es brindar una herramienta para análisis eficaz de cualquier documento en cualquier tipo de formato y de esta forma realizar distintas tareas como: reproducción de documentos, bibliotecas digitales, recuperación de información, conversión de texto a voz o la identificación del orden de lectura de un documento a partir de su imagen siempre y cuando la calidad y el estado de la imagen lo permita (Pratikakis, Gatos, Danatsas, & Perantonis, 2005), (Vincent, 2007).

En este proyecto nos centraremos en la identificación de Orden de Lectura; el Orden de Lectura de un documento es la secuencia lógica de interpretación de la información contenida en un documento. En primer lugar, los documentos se caracterizan por dos estructuras importantes: la estructura de diseño y la estructura lógica. El primero se basa en la presentación del contenido del documento, mientras que el segundo es como se interpreta la información por el usuario. La detección de orden de lectura comprende el análisis de la distribución del documento para identificar las piezas clave como: títulos, resúmenes, secciones, imágenes, tablas, párrafos, entre otros, con el fin de asociarlos y dar una interpretación lógica y coherente de la información existente en el documento. Un ejemplo de detección de orden de lectura se muestra en la Figura 1, donde los número son los identificadores de los bloques de información y la línea verde que los une es el orden lógico de lectura (Thomas, 2003).

Figura 1

Ejemplo de orden de lectura. a) Texto original. b) Orden de lectura del texto.



a) b)

La conversión de texto a voz se vuelve un tema muy interesante al realizar este proyecto, según la Organización Mundial de la Salud (OMS) en 2019 se registraron al menos 2200 millones de personas con deficiencia visual o ceguera. Según la Organización Nacional de Ciegos Españoles (ONCE), la deficiencia visual se define como la persona que “con la mejor corrección posible solamente puede ver o distinguir, aunque con gran dificultad, algunos objetos a una distancia muy corta”. Las personas con deficiencia visual, a diferencia de las que poseen ceguera, aún conservan un grado de visión para su vida diaria. Las principales causas de estas afecciones se tiene: cataratas, el tracoma, miopía, retinopatía diabética, entre otros (Salud, 2019) (Specsavers Ópticas, s.f.). En Ecuador la cifra de personas con algún tipo de discapacidad visual es de 54.956

hasta el año 2019 (Social, 2019). Con un sistema de reconocimiento de orden de lectura se podría, en un futuro trabajo, realizar un sistema el cual permita la conversión directa entre cualquier tipo de texto, sin importar su estructura, a voz. Esto sería de gran ayuda ante el grupo vulnerable ya mencionado.

Existe gran variedad de trabajos relacionados con el tema de identificación de orden de lectura, entre los más importantes tenemos:

En 1998, R. Cattoni, T. Coianiz, S. Messelodi, y C. M. Modena realizan una recopilación de distintos métodos de análisis OCR de documentos con el fin de etiquetar cada uno de los elementos pertenecientes a un texto (encabezados, párrafos, columnas, etc.) y así establecer un orden lógico de lectura (Cattoni, Coianiz, Messelodi, & Modena, 1998). En el trabajo mencionado se abarca distintos métodos como:

- La transformación de árbol
- Sistema de pizarra
- Enfoque sintáctico
- Modelos ocultos de Markov
- Modelos basados en Aprendizaje o más conocidos como *Machine Learning.*

En 2003, Marco Aiello en sus artículo “Bidimensional Relations for Reading Order Detection” plantea un algoritmo de detección de orden de lectura

que usa una serie de proposiciones cualitativas de los rectángulos, como la ubicación del texto basado en coordenadas cartesianas, para la identificación de texto en bloques rectangulares, posterior a esta identificación se usan proposiciones, como: “si un rectángulo de texto se encuentra sobre otro rectángulo, este se debe leer primero”, para establecer el orden de lectura del documento. El principal problema de este algoritmo es que no puede detectar texto en forma de “L” presentes en revistas o periódicos, otro inconveniente es que las reglas utilizadas resultan ser triviales y propensas a errores además de ser muy costosas en su implementación (Marco & Arnold M.W., 2003), (Cattoni, Coianiz, Messelodi, & Modena, 1998).

Otro método de identificación de orden de lectura se presenta en el artículo “Optimized XY-cut for determining a page reading order” escrito por Jean-Luc Meunier en 2005. En este documento se presenta un método de discriminación de rectángulos de texto basado en la separación entre párrafos y columnas de texto. Tanto este método como el método anterior poseen el mismo problema, no pueden identificar secciones de texto que no sean rectangulares, además, este algoritmo posee el inconveniente que se debe conocer el ancho mínimo requerido de las franjas horizontales / verticales del texto para el reconocimiento de los bloques de lectura (Jean-Luc, 2005), (Marco & Arnold M.W., 2003).

Stefano Ferilli, Floriana Esposito y Domenico Redavid en su artículo “Abstract argumentation for reading order detection” utilizan un sistema de procesamiento denominado DoMInUS (Sistema Inteligente de Gestión Universal

de Documentos) para obtener la estructura de diseño de un documento mediante bloques y así, enfocarse únicamente en la detección de orden de lectura en base a argumentación abstracta. El procedimiento se basa en 3 reglas las cuales permiten asociar los bloques de texto en pares interconectados dando así el orden de lectura del documento (Stefano, Domenico, Domenico, & Floriana, 2014).

Todos los métodos mencionados se basan en reglas las cuales son aplicadas de cierta forma para detectar el orden de lectura en un documento previamente segmentado. En el proyecto que estamos desarrollando se buscará un algoritmo de inteligencia computacional para determinar el Orden de Lectura, para ello se lo puede definir en tres clasificaciones: algoritmos de razonamiento, algoritmos de planificación estratégica y algoritmos de aprendizaje (Gordillo, 2019).

Debido a la problemática, el algoritmo debe ser capaz de la toma de decisiones por lo tanto, los algoritmos de razonamiento son una buena alternativa para detectar el orden de lectura. Existen diferentes tipos de algoritmos de razonamiento, entre ellos se tiene:

- Redes neuronales artificiales
- Aprendizaje automático
- Lógica difusa

Los tres algoritmos mencionados serán la base para encontrar una solución al problema de detección de orden de lectura.

Un método tentativo, debido a la necesidad de toma de decisiones, es la Lógica Difusa. La lógica clásica es un enfoque en el cual cierta proposición solo puede tener dos valores, ya sea verdadero o falso, en nuestro caso 1 o 0, la Lógica Difusa permite que los valores de verdad estén comprendidos entre 0 y 1, es decir, las proposiciones pueden ser verdaderas o falsas en cierta medida a esto se lo denomina grado de pertenencia (Corina & Aldo, 2016) (Carlos).

A medida que se desarrolle el proyecto, se determinará si la Lógica Difusa satisface las necesidades de la problemática o si es necesario implementar alguno de los dos métodos mencionados anteriormente.

Justificación e importancia

La digitalización de documentos ha sido un tema muy explotado en las últimas décadas, mediante sistemas de Reconocimiento Óptico de Caracteres (OCR) se ha logrado almacenar información de: libros, revistas, periódicos, etc. Con el fin de convertirlo en formas simbólicas para su modificación, almacenamiento, recuperación, reutilización y transmisión. Pero los algoritmos de OCR poseen muchas limitantes al momento de procesar documentos con una estructura compleja (documentos con más de una columna, imágenes, tablas, etc).

En un inicio, el análisis de documentos logró extraer las cadenas de caracteres de un documento pero, era necesario establecer un orden lógico de

dicha digitalización para la compresión humana (Cattoni, Coianiz, Messelodi, & Modena, 1998), (Michelangelo, Annalisa, Donato, & Malerba).

Para satisfacer los requerimientos del problema se desarrollaron diversos algoritmos que permiten detectar un orden de lectura de los documentos simples, es decir de una sola columna y a blanco y negro. A medida que se mejoró la capacidad de procesamiento computacional se crearon nuevos algoritmos y otros se modificaron para procesar información de documentos cuya estructura era más compleja (dos o más columnas, encabezados, tablas, imágenes, etc.).

Los algoritmos desarrollados demostraron ser lo suficientemente potentes como para cumplir con los requisitos de muchos usuarios pero aún existe margen de mejora. Un aspecto común de todos los métodos es que dependen en gran medida del dominio específico y no son "reutilizables" cuando cambian las clases de documentos o la tarea en cuestión. Por ejemplo, la clasificación de bloques como "título" es apropiada para artículos de revistas, pero no para documentos administrativos. Además, los artículos occidentales y japoneses tienen diferentes reglas de codificación de documentos, este es el caso de los algoritmos basados en el aprendizaje (Michelangelo, Margherita, Giuseppe, & Donato, 2007). Otros algoritmos se basan en reglas lingüísticas para superar algunas limitaciones pero requieren de técnicas de Procesamiento de Lenguaje Natural (NLP), pero esto hace que los enfoques sean más complejos y dependan del idioma (Stefano, Domenico, Domenico, & Floriana, 2014). Muchos de los enfoques suponen que hay un único orden de lectura

correcto, esto es erróneo ya que esto depende de la estructura del documento y muchas veces del mismo idioma.

En este proyecto se estudiará la detección de orden de lectura mediante algoritmos de inteligencia computacional. En los últimos años, se ha convertido en un subcampo influyente de Inteligencia Artificial, con aplicaciones que van desde el control de máquinas hasta la administración electrónica. Para el desarrollo del algoritmo se tendrá en cuenta la información del diseño del documento, las características de los bloques de texto y una serie de reglas implementadas mediante un algoritmo de inteligencia computacional, en este caso, un algoritmo de razonamiento.

Una de las futuras aplicaciones de este proyecto es la implementación de un sistema de conversión de texto a voz, esto no es una idea nueva, existen varios softwares que hacer esta labor pero, estos no pueden trabajar con documentos cuya estructura es compleja. Como se muestra en (Salud, 2019) y (Social, 2019) existe un gran número de personas con alguna discapacidad visual, las cuales se vería beneficiadas por un sistema como ya mencionado, ellos podrían acceder a cualquier tipo de información almacenada en documentos. Una de las posibles ventajas es que se podría facilitar el aprendizaje de estas personas ya que no necesitarían aprender algún método de lectura. De esta forma se aportaría al derecho de una educación inclusiva que se describe en el capítulo segundo, sección tercera, Art.28 de la Ley Orgánica de Discapacidades (Asamblea Nacional del Ecuador, 2012).

Alcance

Este proyecto pretende realizar un método de segmentación de documentos en bloques, basado en procesamiento digital de imágenes, para la extracción de información de dichos documentos y su posterior análisis con el fin de determinar su orden lógico de lectura. El orden de lectura se determinará mediante un algoritmo de inteligencia computacional que comprende una serie de reglas que proporcionan un enfoque general para modelar el razonamiento no monotónico (Kacprzyk, 2008).

El algoritmo se dividirá de la siguiente manera:

Digitalización del documento

Para procesar los documentos es necesario convertirlos en un formato que sea manipulable, dicho formato será en imágenes, esta digitalización se la realizará mediante el software Ghostscript® el cual se encargará de realizar la conversión de los documentos en formato PDF a un formato PNG en blanco y negro (0 o 1). (Artifex, s.f.)

Las imágenes a color y en escala de grises deben ser Binarizadas, este proceso consiste en convertir los distintos valores de los pixeles de una imagen a dos valores: 0 o 1 dependiendo de un umbral previamente establecido. Este proceso ayuda de gran manera a la reducción de la cantidad de datos contenidos en la imagen.

Discriminación de bloques

Se implementará un algoritmo basado en procesamiento digital de imágenes y algoritmos de segmentación ya conocidos como el algoritmo *Run-Length Smoothing Algorithm* (LRSA) para obtener los elementos que constituyen el documento (Stefano, Domenico, Domenico, & Floriana, 2014).

El algoritmo LRSA consta de los siguientes pasos:

a. Primero, un procedimiento de segmentación subdivide el área de un documento en regiones (bloques). cada una de las cuales debe contener solo un tipo de datos (texto, gráfico, imagen de medios tonos, etc.).

b. Se aplica a una secuencia binaria en la que los píxeles blancos están representados por 0 y los negros por 1. El algoritmo transforma una secuencia binaria a en una secuencia de salida y de acuerdo con las siguientes reglas:

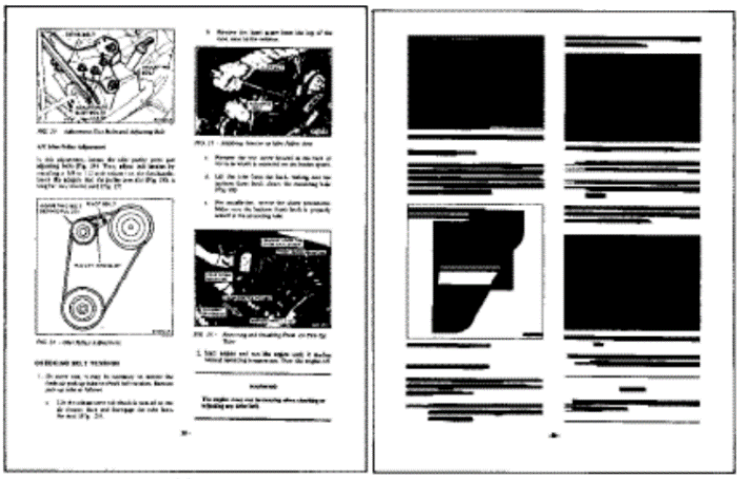
i. Los 0 en a se cambian a 1 en y si el número de 0 adyacentes es menor o igual a un límite predefinido C.

ii. Los 1 en a no cambian en b.

El LRSA se aplica fila por fila, así como columna por columna a un documento, produciendo dos mapas de bits distintos. El resultado final es la segmentación del documento. Figura 2. (HASNAT, 2007)

Figura 2

Ejemplo de segmentación a) Documento Original. d) resultado final de la segmentación.



(a) (d)

Extracción de características de los bloques de información

El algoritmo *LRSA* proporciona una serie de características de los bloques de texto segmentados, pero estas características no son suficientes para determinar un orden lógico de lectura. Por esta razón, en base a la bibliografía se definirán reglas generales que determinen cuál es el orden de lectura que deben seguir los documentos.

Entre las reglas a aplicar están (Stefano, Domenico, Domenico, & Floriana, 2014):

a. Los componentes adyacentes horizontal o verticalmente son candidatos para ser leídos en consecuencia.

b. Un componente en la parte inferior de la página puede ir seguido de un componente en la parte superior de una columna adyacente.

c. Un componente de la derecha puede ser seguido por un componente de la izquierda en una fila adyacente.

A medida que se desarrolle el proyecto se agregarán más reglas con el objetivo de realizar un algoritmo más confiable (Corina & Aldo, 2016).

Bloque de decisión

Mediante las reglas definidas y las características obtenidas por medio de LRSA se analizará los bloques de información por medio de algoritmos de Inteligencia Computacional para determinar la secuencia de lectura lógica del documento.

Al finalizar el proyecto se obtendrá un algoritmo que permita la segmentación de documentos y establezca un orden lógico de lectura de dicho documento.

Para evaluar el desempeño del algoritmo lo ideal sería realizar la comparación con un algoritmo existente pero, esto no se lo puede realizar debido a que los códigos de los algoritmos diseñados no se encuentran implementados o no se cuenta con la base de datos con la cual se realizaron las pruebas, por esta razón la comparación con otros algoritmos queda descartada. En la actualidad no existen métricas específicas para evaluar el rendimiento del

algoritmo a diseñar, a medida que se desarrolle el proyecto se implementarán indicadores que permitan evaluar las prestaciones y desempeño del algoritmo. Una métrica podría ser el número de errores vs el número de aciertos en la detección de lectura del documento.

Objetivos

General

Desarrollar un algoritmo de detección de orden de lectura mediante algoritmos de Inteligencia Computacional.

Específicos

- Realizar un estudio del estado del arte de los métodos y algoritmos de segmentación y detección de orden de lectura de documentos.
- Implementar el algoritmo de segmentación de documentos y extracción de características.
- Implementar el algoritmo de detección de orden de lectura, basado en Inteligencia Computacional.
- Desarrollar la interfaz gráfica para el manejo y presentación de los resultados.
- Implementar métricas que permitan evaluar el desempeño y rendimiento del sistema realizado.

CAPITULO 2

Marco teórico

Introducción

En este capítulo se dará los conceptos básicos de la teoría necesaria para la comprensión y entendimiento del proyecto a realizar. En primer lugar se revisará algunos de los algoritmos de segmentación para extracción de información de los documentos, posteriormente se analizará diversos métodos de extracción de orden de lectura de documentos y se determinará los algoritmos para el desarrollo del proyecto. Al finalizar el capítulo se explicará la importancia del trabajo en la inclusión de personas con discapacidad visual.

Algoritmos de segmentación de texto en bloques de información

A continuación se realiza una revisión de métodos de segmentación en distintos artículos científicos.

Adaptación Multigaussiana

Laiphangbam Melinda, Raghu Ghanapuram y Chakravarthy Bhagvati en su artículo *"Document Layout Analysis using Multigaussian Fitting"* proponen un método de segmentación de documentos tipo Manhattan basado en calcular un histograma de las alturas de los recuadros delimitadores de los componentes conectados y, mediante un ajuste multigaussiano se logra identificar los puntos óptimos de división entre las categorías: ruido, texto, títulos y gráficos en función de su altura. El en trabajo realizado se indica que el gaussiano con el pico más

alto corresponde al tipo de texto más común utilizado en el documento. (Laiphangbam, Raghu, & Chakravarthy, 2017)

Posteriormente, las regiones de texto se agrupan en bloques mediante el análisis de vecinos más cercanos y mediante una clasificación de segundo nivel se refinan dichas regiones

El algoritmo propuesto en el artículo tiene los siguientes pasos:

- Binarización: utiliza la binarización de imagen de documento adaptativa de Sauvola, ya que maneja variaciones en la iluminación y el ruido.
- Componentes conectados (CC) y Bounding Box ($BBox$)
- Análisis de $BBox$ mediante histograma: el histograma de altura se suaviza mediante una función gaussiana para facilitar la búsqueda de picos y valles más estables.
- Ajuste multigaussiano
- Clasificación de contenedores
- Clasificación de no texto

El resultado del algoritmo propuesto se muestra a continuación:

Figura 3

Ejemplo de segmentación. *(a) Imagen original. (b) Segmentación utilizando adaptación Gaussianos. (Laiphangbam, Raghu, & Chakravarthy, 2017)*

Segmentación de documentos de periódicos no estructurados

En el artículo *"Segmentation of Unstructured Newspaper Documents"* escrito por Santosh Naik y sus colaboradores se plantea un método de segmentación de arriba hacia abajo en el cual se segmenta las imágenes del documento y luego se separa los títulos en él, después de esto se separa las columnas y finalmente se divide el texto en párrafos, líneas y luego palabras. (Naik & Ramegowda, 2017)

Los pasos para la segmentación en este algoritmo son las siguientes:

- Binarización de la imagen de entrada.
- Segmentación de encabezados.
- Separando las columnas
- Separando las imágenes y líneas de las columnas.
- Separando finalmente las palabras

La deficiencia de este algoritmo es que se enfoca en la segmentación de palabras además de ser poco adaptable para otro tipo de documentos con una estructura más compleja. Un ejemplo aplicativo del código en cuestión es el mostrado en la Figura 4.

Figura 4

Ejemplo de segmentación. (a) Imagen original. (b) Segmentación utilizando el método propuesto. (Naik & Ramegowda, 2017)

(a) (b)

Extracción de texto en imágenes de documentos: resalte el uso de puntos de esquinas

Vikas Yadav y Nicolas Ragot proponen un método novedoso para identificar el texto en documentos, dicho método se basa en la identificación de puntos de esquina. (Yadav & Ragot, 2016) El algoritmo se basa en la técnica $FAST\ Corner\ Detection$, los pasos son los siguientes:

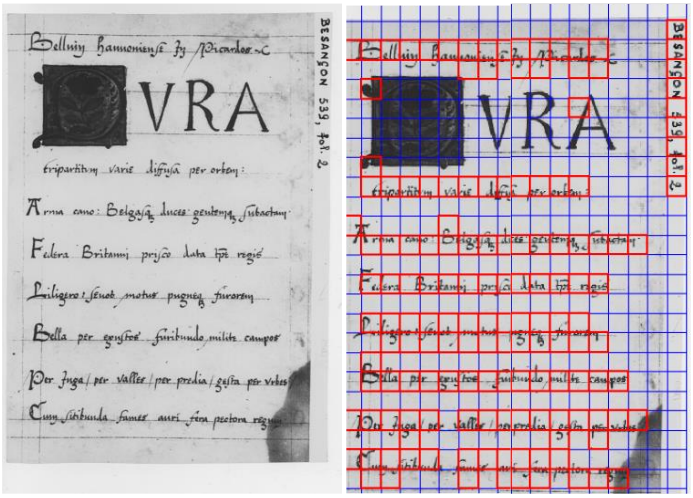
- Suavizado de la imagen de entrada mediante el filtro gaussiano
- Determinación los puntos de esquina de la imagen mediante *FAST Corner Detection*
- Dividir la imagen en bloques de $32\ x\ 32$ pixeles
- Se define un umbral con el bloque con más puntos de esquina
- Los bloques que poseen más puntos de esquina que el umbral se definen como texto y se analizan los bloques vecinos para identificar cadenas de palabras
- Los bloques que tienen menos puntos de esquina que el umbral son clasificados como imágenes, fondos o ruido.

Este método sirve para la extracción de información de documentos con desviaciones en su estructura debidas a malas técnicas en escaneo o para documentos con ruido, si bien es un algoritmo muy útil, no se ajusta a las necesidades del proyecto planteado, ya que, los documentos a utilizar en este

proyecto no tienen este tipo de inconvenientes en la digitalización. Un ejemplo del uso del procedimiento de segmentación se muestra en la Figura 5.

Figura 5

Ejemplo. a) Imagen original. (b) Imagen segmentada por el Método de Puntos de Esquina. (Yadav & Ragot, 2016)



(a) *(b)*

Análisis de diseño de documentos: enfoque de la región máxima homogénea

Tuan Anh Tran en su artículo *"Document Layout Analysis: A Maximum Homogeneous Region Approach"* realiza un análisis de los espacios en blanco en regiones homogéneas horizontal y vertical para clasificar elementos de texto y elementos no textuales. (Tuan, Khuong, & Nhat, 2018)

Los pasos del algoritmo desarrollado son los siguientes:

- Partiendo de un documento en binario se corta el documento de entrada verticalmente para obtener las regiones homogéneas verticales
- Posterior a esto se realiza la segmentación horizontal para extraer las regiones homogéneas horizontales.
- Se realiza la identificación de los espacios en blanco para identificar los elementos que son de texto y no son texto mediante el uso del método estadístico.
- En el siguiente paso, los componentes de texto se agrupan para obtener las regiones de texto extrayendo líneas de texto mediante morfología matemática.

El algoritmo propuesto es un método interactivo el cual depende de los espacios en blanco que contiene el documento para su convergencia, en la práctica, esto se vuelve muy repetitivo y genera altos tiempos de ejecución del programa. En la Figura 6 se muestra el resultado obtenido para una imagen.

Figura 6

Ejemplo. a) Imagen original. (b) Imagen segmentada. (Tuan, Khuong, & Nhat, 2018)

Digitalización de documentos históricos mediante análisis de diseño y profunda clasificación de contenido

Andrea Corbelli utiliza el algoritmo XY-CUT y lo combina con un método de clasificación de abajo hacia arriba que se basa en características de geometría locales. (Corbelli, Baraldi, Grana, & Cucchiara, 2016)

El algoritmo desarrollado se divide en 3 secciones:

A. Segmentación de diseño

Las regiones se clasifican en diferentes tipos de diseños que utilizan las características extraídas de una red neuronal convolucional combinados con un clasificador Random Forest, incluido en la librería de OpenCV, analizado los espacios en blanco en regiones homogéneas horizontal y vertical para clasificar elementos de texto y elementos no textuales. (OpenCV, 2021)

B. Clasificación de contenido

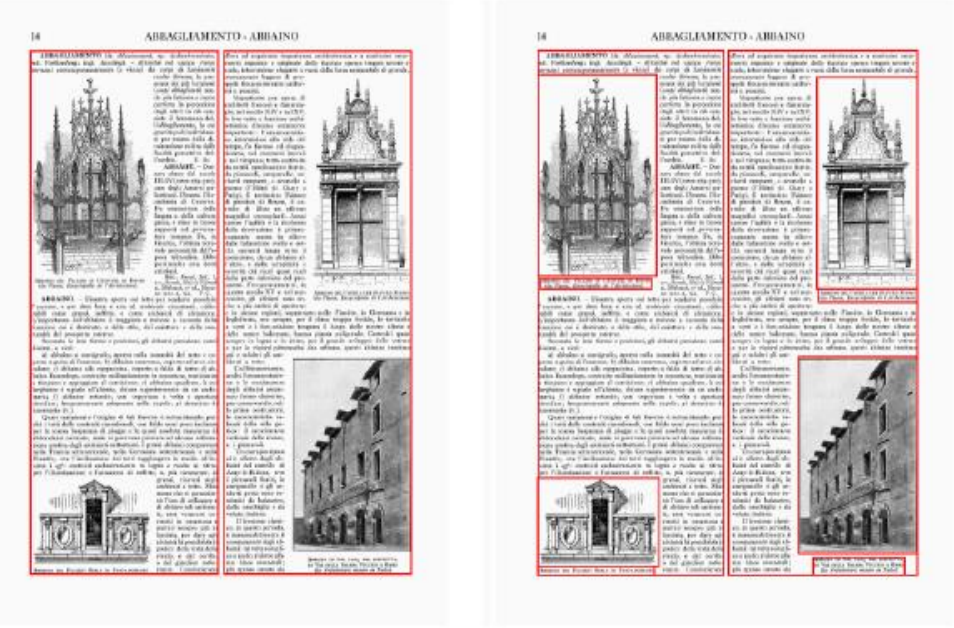
Para la extracción de las regiones de texto, aplicaron la morfología matemática. Posteriormente, mediante el uso de técnicas de aprendizaje automático, los elementos no textuales se clasifican en separadores, tablas, imágenes. El resultado se muestra en la Figura 7.

C. Mapeo de OCR

A demás de la segmentación y la clasificación de contenidos, en dicho artículo se presenta una técnica para mapear texto que, para el proyecto a realizar no es pertinente su estudio.

Figura 7

Ejemplo. a) Imagen original segmentada mediante X-Y Cut. (b) Imagen segmentada por el método propuesto. (Corbelli, Baraldi, Grana, & Cucchiara, 2016)



Algoritmo de Suavizado de Longitud de Ejecución (*LRSA*)

Uno de los métodos más exitosos es el Algoritmo de Suavizado de Longitud de Ejecución o *LRSA* en el cual, a parte de los métodos de binarización se caracteriza por realizar un análisis bit a bit de la imagen digitalizada (Ferilli, Basile, & Esposito, 2010).

El método tiene como pasos los siguientes:

- Suavizado horizontal, realizado en filas sobre la imagen con umbral th;
- Suavizado vertical, realizado en columnas sobre la imagen con umbral tv;
- AND lógico de las imágenes obtenidas en los pasos 1 y 2;
- Suavizado horizontal con umbral ta en la imagen obtenida en el paso 3, para rellenar tramos blancos dentro de los bloques descubiertos.

El algoritmo devuelve regiones rectangulares (Figura 8, d), cada una de las cuales define un área de información.

Figura 8

Ejemplo. a) Imagen original. b) LRSA Horizontal. c) LRSA Vertical. d) Resultado final de la segmentación. (HASNAT, 2007)

Algoritmo de Suavizado de Longitud de Ejecución con OR (LRSO)

Este algoritmo se basa en el Algoritmo de Suavizado de Longitud de Ejecución con la diferencia que se una operación lógica *OR* en lugar de una operación *AND*. (Ferilli, Basile, & Esposito, 2010)

Los pasos para este algoritmo son:

- Suavizado horizontal, realizado en filas sobre la imagen con umbral th;
- Suavizado vertical, realizado en columnas sobre la imagen con umbral tv;
- OR lógico de las imágenes obtenidas en los pasos 1 y 2.

Mientras que *RLSA* devuelve marcos rectangulares, el algoritmo *RLSO* identifica regiones irregulares, como se muestra en la Figura. 9

Figura 9

a) Figura original. b) Regiones irregulares que entrega LRSO. (Ferilli, Basile, & Esposito, 2010)

a) b)

Métodos de detección de orden de lectura

Con el paso del tiempo se han implementado distintos algoritmos de detección de orden de lectura, a continuación se realiza una revisión de algunos métodos en distintos artículos científicos.

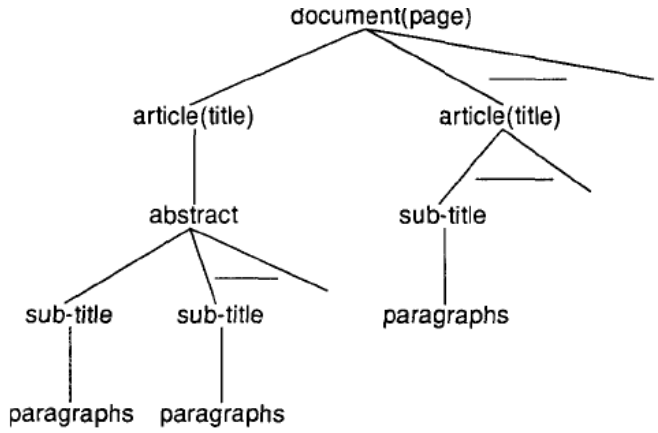
La transformación de árbol

Dado un documento con un contexto similar al propuesto por el estándar ODA (*Open Document Architecture*), el análisis de diseño lógico es el proceso para construir el árbol lógico a partir de la información reunida en el árbol geométrico. En el sistema de Tsujimoto y Asada, la transformación del árbol se realiza con un conjunto de reglas deterministas, posiblemente repetidas, que etiquetan los bloques y definen su orden de lectura. En experimentos realizados en documentos de diferentes categorías (revistas, revistas, periódicos, libros, manuales, cartas y artículos científicos), la estructura lógica se determinó correctamente para 94 de los documentos de 106. (Shuichi & Haruo, 1992)

La transformación del árbol se realiza con un conjunto de reglas ponderadas, luego el árbol lógico más probable se valida haciendo coincidir el contenido del texto con el conocimiento almacenado en vocabularios. Ver Figura 10.

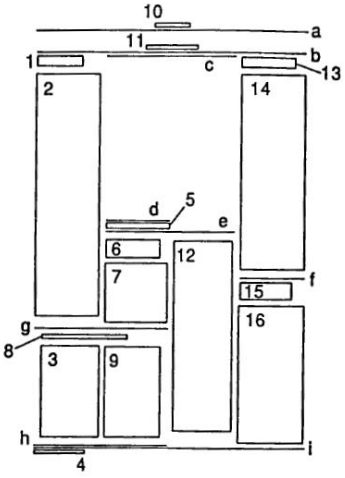
Figura 10

a) Estructura lógica de árbol. b) Imagen original. c) Orden de lectura por el método de árbol. (Shuichi & Haruo, 1992)



a)

b)



c)

Sistema de pizarra

Un enfoque típico para el control de un sistema de IA (Inteligencia artificial) es el enfoque de pizarra, muy popular en los años 80. La idea básica de un sistema basado en pizarra es dividir un problema complejo en subtareas débilmente acopladas, cada una gestionada por un procedimiento especializado que codifica un conocimiento particular y actualiza un área de datos común, la pizarra.

Durante el cálculo, cada procedimiento está asociado a una puntuación dinámica que codifica su grado de aplicabilidad al contexto actual. Un planificador codifica la estrategia para dividir el problema en subtareas y seleccionar el mejor procedimiento en cada paso, utilizando reglas fijas o ponderadas de “si-entonces”. Las ventajas y desventajas del enfoque de pizarra, ampliamente discutido en la comunidad de IA, están fuera del enfoque de este documento.

Srihati y sus colaboradores desarrollaron un sofisticado sistema de pizarra organizado como una jerarquía de tres niveles para reconocer los bloques de direcciones en los correos. Se utiliza una combinación de marcos y reglas para modelar el conocimiento para seleccionar el procedimiento que se aplicará y para calcular el grado de plausibilidad de un etiquetado particular. (Srihari, Wang, Palumbo, & Hull, 1987)

Yeh y colaboradores, proponen una versión restringida de un sistema de pizarra aplicada a la lectura automática de direcciones postales. El análisis de diseño lógico se basa en reglas secuenciales con heurística aplicada a los

bloques segmentados: no encontrar etiquetas aceptables determina una nueva segmentación de página con diferentes umbrales. De esta manera, el análisis de diseño lógico y geométrico se complementa. (Pen-Shu, Antoy, Litcher, & Rosenfeld, 1987)

Enfoque sintáctico

En este enfoque, el conocimiento requerido para segmentar la página en bloques y etiquetarlos se representa mediante gramáticas formales (generalmente libres de contexto)

Por lo tanto, el análisis de diseño geométrico y lógico se realiza con programas para el análisis sintáctico (por ejemplo, un analizador sintáctico), obtenido de las gramáticas formales. George Nagy desarrolló un sistema sintáctico que trabaja con revistas técnicas. Definen un conjunto de gramáticas libres de contexto apropiadas, cada una define reglas para agregar píxeles en entidades más y más estructuradas, hasta los objetos lógicos. A partir de las gramáticas, se obtienen automáticamente programas para el análisis sintáctico (analizadores sintácticos): luego se utilizan para realizar la segmentación y el etiquetado en la misma fase (simultáneamente). Se utiliza un conjunto de gramáticas alternativas para permitir que se extraigan y verifiquen diferentes estructuras de documentos. Un algoritmo de ramificación y búsqueda busca la mejor estructura. El criterio para optimizar y depurar la búsqueda se basa en el área acumulativa de los bloques etiquetados: cuanto mayor sea el área, mejor será la estructura. (Nagy, 1992)

Modelos ocultos de Markov

Las probabilidades apenas se han mencionado en los enfoques anteriores que tienden a utilizar heurística determinista en los espacios de búsqueda. Kopec y Chou propusieron una técnica completamente diferente basada en una extensión de HMM (Modelos Ocultos de Markov) para la extracción de textos de páginas amarillas de guías telefónicas. Ampliar los HMM para gestionar regiones de imagen bidimensionales no es sencillo y plantea problemas tanto teóricos como computacionales. En el enfoque de Kopec y Chou, la característica básica en la que trabajan los HMM es el glifo, una porción de imagen que representa un solo carácter. Esta solución permite una tasa de reconocimiento muy alta, pero el tiempo requerido es enorme. (Kopec & Chou)

Aprendizaje

Las técnicas de aprendizaje que apuntan a una automatización del proceso de suministro de conocimiento parecen ser una buena solución hacia una mayor generalidad y flexibilidad, pero, pueden fallar con documentos que no pertenecen al dominio para el que fue entrenado.

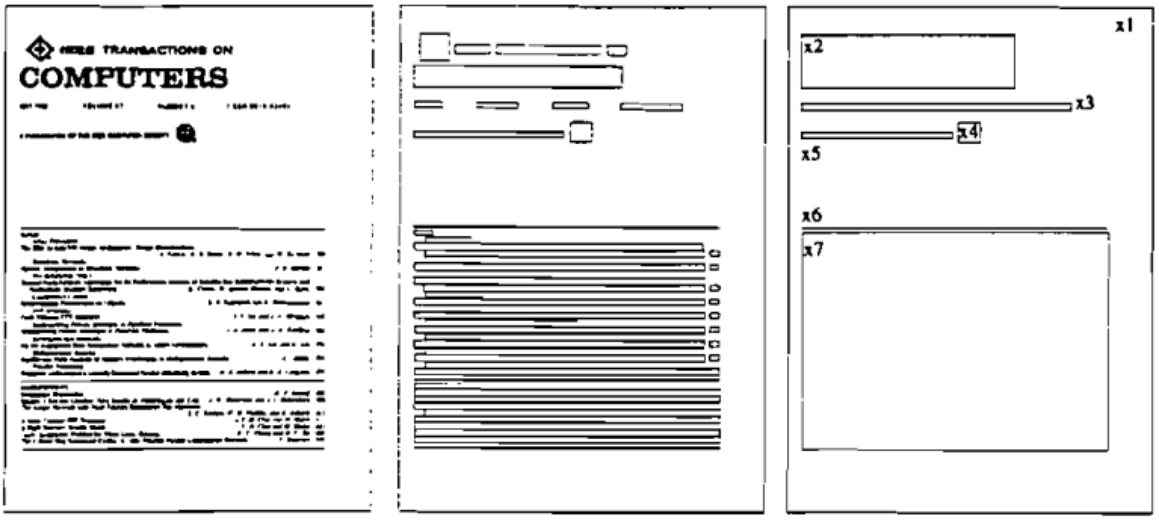
Un sistema particularmente interesante ha sido desarrollado por Floriana Esposito y sus colaboradores en el que se da una gran relevancia al problema de la clasificación automática de documentos destinada al proceso de identificación de la categoría de un documento de imagen de entrada. Abordan la tarea de clasificación mediante técnicas de aprendizaje automático para adquirir directamente las reglas de clasificación de un conjunto de documentos de capacitación. Incluso el análisis de diseño lógico se aborda como otro

problema de aprendizaje supervisado: el sistema tiene que aprender el mapeo entre las características extraídas de la imagen del documento y la estructura lógica del documento. Al procesar un documento de entrada, el análisis de diseño geométrico extrae los bloques y algunas características relevantes que permiten que el subsistema de clasificación identifique la categoría de membresía del documento. Esta información simplifica enormemente el análisis de diseño lógico que utiliza las reglas aprendidas específicas para la categoría particular para reconstruir la estructura lógica. (Esposito, Malerba, & Semeraro, 1994)

En la Figura 11 se muestra un ejemplo aplicativo del método propuesto por Esposito y sus colaboradores.

Figura 11

Ejemplo de documento: a) Imagen a procesar. b) Imagen segmentada. c) Orden de lectura del documento. (Esposito, Malerba, & Semeraro, 1994)



a) *b)* *c)*

Fundamentos sobre Inteligencia Computacional

La Inteligencia Computacional se ocupa del diseño y desarrollo de distintas aplicaciones para solventar distintas problemáticas mediante mecanismos adaptativos. Comúnmente, estos mecanismos son inspirados en la lingüística o en la biología. (Pérez, 2010)

A partir de la década de los 60 y 70 se comenzó a implementar algoritmos que combinaban el aprendizaje, razonamiento, adaptación y evolución para generar sistemas inteligentes que se adapten a las necesidades del problema a solventar, entre las técnicas desarrolladas están:

- Redes Neuronales Artificiales.
- Máquinas de Vectores Soporte.
- Computación Evolutiva.
- Inteligencia Colectiva.
- Aprendizaje Automático.
- Lógica Difusa.

La técnica para solventar el problema propuesto es la Lógica Difusa debido a que se adapta a las necesidades del sistema.

Lógica Difusa

Lotfi Zadeh desarrolló la teoría de conjuntos difusos en 1965. Esta lógica pretende representar el razonamiento aproximado e impreciso. Para entender su

funcionamiento en primer lugar se debe explicar la Lógica Clásica. (Pérez, 2010)

La Lógica Clásica únicamente admite dos estados en sus proposiciones, 0 o 1. Para comprender esto se presenta el siguiente ejemplo:

Con 0 o 1 solo se puede representar dos estados, si se quisiera representar el clima de un día podemos decir lo siguiente (Pérez, 2010):

- Hace frío
- Hace calor

Donde:

- Hace frío → representado por 0
- Hace calor → representado por 1

Como se observa, únicamente se admiten dos estados para expresar el clima lo que limita las posibilidades para representar más circunstancias.

En Lógica Difusa se puede representar más de dos estados, tomando el ejemplo anterior se podría representar los siguientes casos:

- Hace mucho calor → representado por 1
- Hace calor → representado por 0.75
- Está templado → representado por 0.50
- Hace frío → representado por 0.25
- Hace mucho frío → representado por 0

Entre las principales ventajas que tiene este tipo de sistemas son las siguientes:

- Los problemas complejos se vuelven más simples al utilizar el razonamiento aproximado.
- No se necesita un modelo matemático que describa el comportamiento de la planta, por ende, no se necesita la linealización.
- No es necesario aproximar el modelo siempre y cuando se conozca las reglas que definan el fenómeno.

Existen otros conceptos que son de gran importancia para comprender esta metodología y se muestran a continuación.

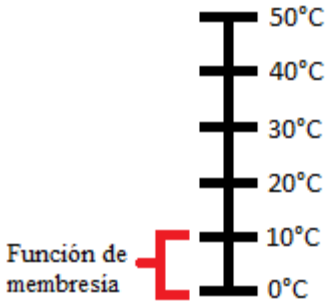
Sistema de inferencia

El sistema de inferencia se compone por las funciones de membresía y las reglas de inferencia. Donde las funciones de membresía son todos los elementos de un conjunto asignados a una variable lingüística y esta a su vez forma parte de un universo dado llamado universo de discurso. (Pérez, 2010)

Un ejemplo sería el expresar la temperatura de una habitación que va de 0 a 50°C (universo de discurso) la representación se muestra en la Figura 12.

Figura 12

Ejemplo de reglas de inferencia.



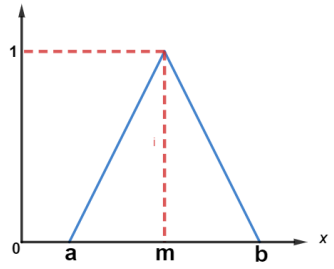
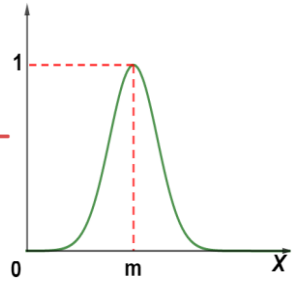
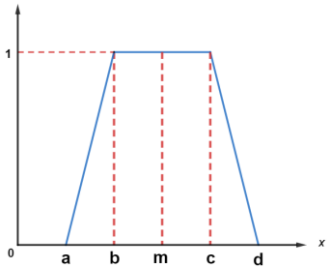
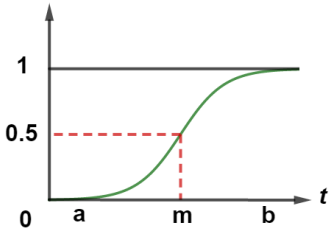
Las reglas de inferencia es un conjunto de reglas que interactúan con las variables de entrada y salida, a su vez, definen el comportamiento del sistema de control.

Por su parte, el número de funciones de membresía pueden ser tantas como se necesite para solventar al fenómeno.

Existen varios tipos de funciones de membresía entre ellas se tiene:

Tabla 1

Funciones de membresía.

Funciones de membresía	
Triangula	Gaussiana
	$f(x) = ae^{\frac{(x-b)^2}{2c^2}}$ 
Trapezoidal	Sigmoidal
Donde a, b son los límites de la función y $\boldsymbol{m}$ representa el valor medio de la función	Donde a, b y c son constantes reales (c>-1) y m representa el valor medio de la función
	$P(t) = \frac{1}{1+e^{-t}}$ 
Donde a, d son los límites de la función; b y c son os valores donde la función alcanza su máximo valor y $\boldsymbol{m}$ representa el valor medio de la función	Donde a, b son los límites de la función y m representa el valor de t donde la función alcanza 0.5 del valor máximo que puede tomar

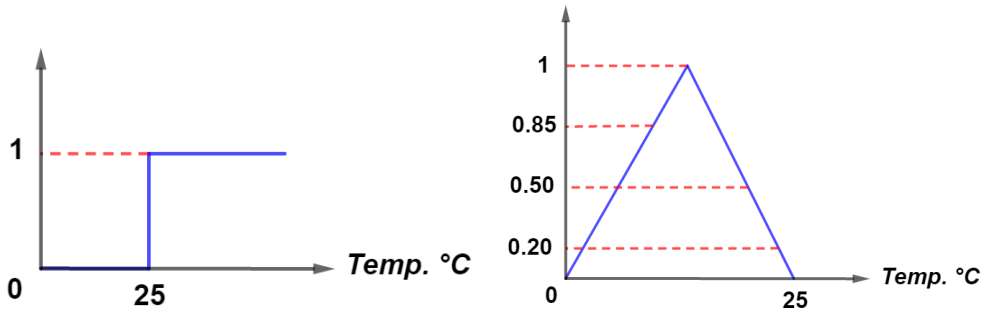
La utilización de cada una de las funciones de membresía depende de la aplicación, por ejemplo, la función Sigmoidal es utilizada cuando los fenómenos describen curvas de aprendizaje que necesitan un cambio gradual al inicio y hasta su estado final.

Las funciones de membresía más utilizadas son la función triangular y la función trapezoidal.

Para comprender en qué se diferencian las funciones de membrecía de lógica difusa con las funciones de membresía de la lógica clásica se debe analizar la Figura 13.

Figura 13

a) Función de membresía de lógica clásica. b) Función de membresía de lógica difusa.



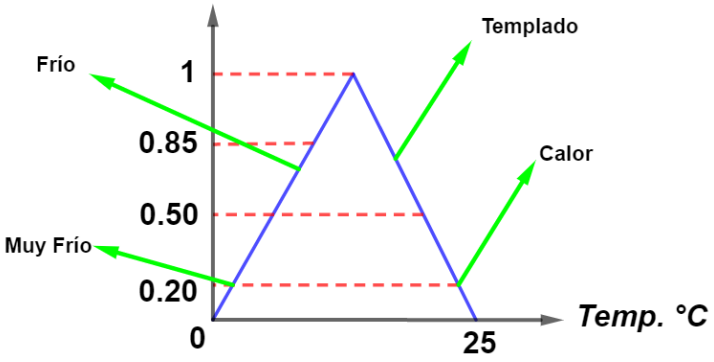
a) b)

En la Figura 13 a) se representa los dos únicos estados de activación en la lógica clásica al llegar a un valor determinado, en este caso, la temperatura. Por otra parte, en la Figura 13 b) se muestra la función de membresía triangular utilizada en lógica difusa en la que, para cada valor de la temperatura corresponde un valor de veracidad, a cada uno de estos valores se los representa con variables lingüísticas. (Duarte, 1999)

Las variables lingüísticas sirven para representar de forma semántica diferentes valores de una variable. En la Figura 14 se muestra que para cada grupo de valores de la variable temperatura se asigna una representación lingüística.

Figura 14

Representación de variables lingüísticas.



De esta forma, si la temperatura es de unos 10°C se podría decir que está frío, esto se lo puede realizar para cada valor posible de la variable temperatura.

En primer lugar, para representar las variables lingüísticas mencionadas anteriormente se debe proponer un ejemplo que se adapte a la realidad y realizar un cambio en la representación:

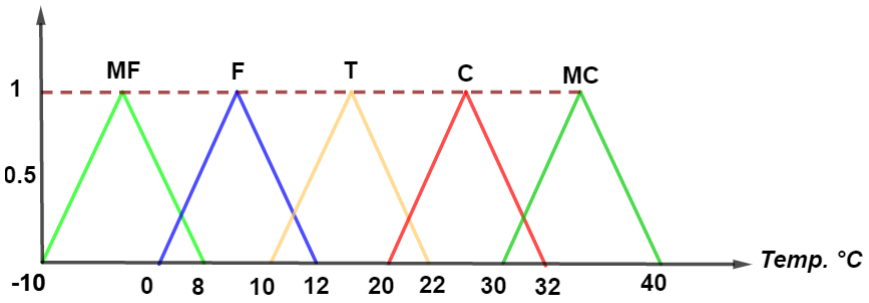
En el ejemplo de la Figura 15 se representa un universo de discurso que va desde los
-10°C a los a los 40°C, para ello se representa las siguientes 5 variables lingüísticas

- Muy frío (MF)
- Frío (F)
- Templado (T)
- Caliente (C)
- Muy caliente (MC)

A cada función de membresía se le asigna una variable lingüística. El número de variables lingüísticas dependerá de la exactitud con la que se requiera representar los estados de universo de discurso.

Figura 15.

Representación de funciones de membresía.



Para la variable lingüística MF corresponderán los valores entre -10°C a 8°C, esto se lo puede repetir para cada función de membresía.

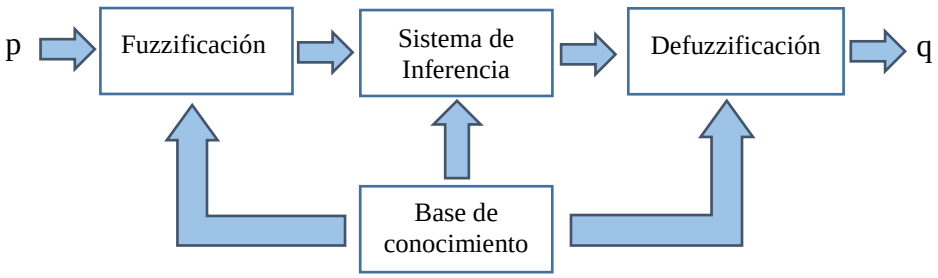
Sistema de Control Difuso

Una vez explicados estos conceptos se puede tener idea de cómo funciona la lógica difusa, en la Figura 16 se muestra un diagrama de bloques

que muestra como se procesa la información con conceptos no vistos hasta el momento.

Figura 16

Diagrama de bloques del Sistema de control difuso.



Dada una variable de entrada "p" el primer paso para un control difuso es la Fuzzificación que se encarga de otorgar valores lingüísticos a la entrada del sistema basado en el conocimiento otorgado por el diseñador el controlador.

Posterior a esto, en el bloque de Sistema de inferencia se evalúa la información Fuzzificada dependiendo de reglas difusas las cuales son usadas para formular las expresiones condicionales que abarca la lógica difusa y generar acciones de control lingüísticas.

$$A \rightarrow B \equiv If\ p\ is\ A\ then\ q\ is\ B$$

- Donde A y B son variables lingüísticas definidas en los rangos de los universos de discurso.
- "If" es el condicional de la regla difusa denominado antecedente o premisa.

- *"Then"* es la consecuencia o cómo va actuar el control en función de las condiciones.

Ejemplo:

"Si el clima es frío entonces colocarse una chompa."

En el ejemplo,

- "Si" es el condicional
- "clima " es la variable de entrada y "frío" es su expresión en forma lingüística
- "entonces" es la consecuencia
- "colocarse una chompa" es la acción expresada en forma lingüística

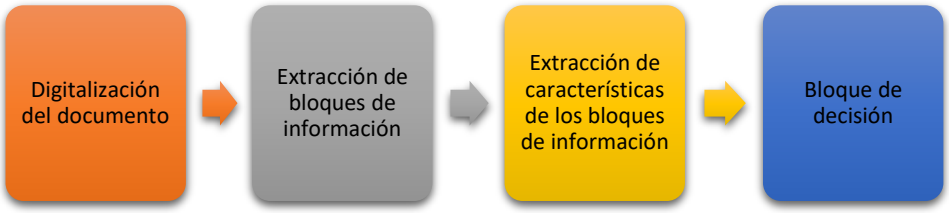
Y la parte final del sistema de control es la Defuzzicación, que es el proceso inverso a la Fuzzificación la cual se encarga de dar valores numéricos a las variables lingüísticas de salida para su acción de control.

Determinación del algoritmo a utilizar

Mediante la revisión bibliográfica se pudo determinar que el proyecto en rasgos generales posee 4 grandes bloques:

Figura 17

Diagrama de bloques para la determinación del orden de lectura.



Digitalización del documento

La digitalización de la información contenida en un archivo en formato PDF es el primer paso para la elaboración del presente proyecto. Para ello se utilizará el software Ghostscript® que se especializa en obtener y analizar datos de texto. A pesar de que este programa nos permite identificar las regiones del texto y analizar el *Boundin Box* del mismo, únicamente se lo utilizará para la conversión del formato PDF a una imagen en blanco o negro, es decir binarizada. (Shinyama, 2013)

En la Figura 18 se muestra un ejemplo de digitalización realizada a través de Ghostscript.

Figura 18

a) Página de un documento en PDF. b) Imagen Binarizada con Ghostscript.

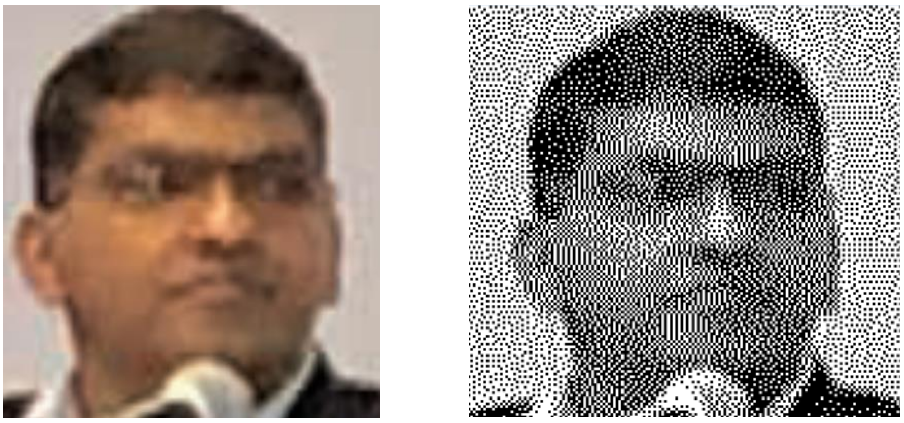
a) *b)*

Si bien, la imagen b) de la Figura 18 pareciera estar en escala de grises, no lo está, es una imagen completamente binarizada es decir, la información en la imagen únicamente corresponde a unos y ceros.

El programa logra el efecto de escala de grises mediante la concentración de puntos negros en las áreas con imágenes, para su mejor comprensión, en la Figura 19 se muestra la concentración de puntos en las imágenes.

Figura 19

a) Imagen extraída del PDF. b) Imagen binarizada con Ghostscript.



a) b)

La binarización que realiza el software es de gran utilidad ya que evita la pérdida de información al usar umbrales para la binarización, además con esta información se realizará un análisis de Componentes Conectados para determinar si una región de información es una imagen o un texto.

Extracción de bloques de información

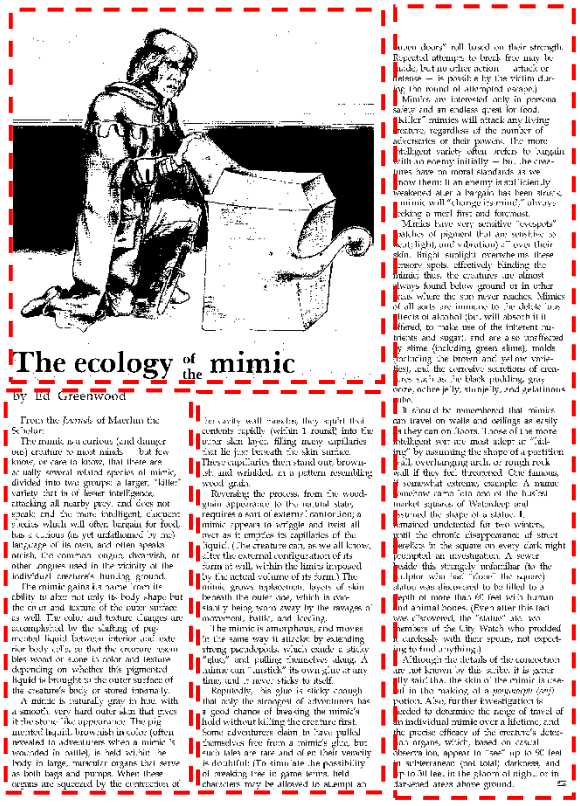
El siguiente paso que se debe realizar es identificar las regiones que poseen información en la imagen. Uno de los algoritmos más exitosos y el que se implementará es el algoritmo LRSA (Algoritmo de Suavizado de Longitud de Ejecución) en el cual se realiza un suavizado horizontal y un suavizado vertical y posteriormente una AND lógico entre las dos imágenes, el resultado se observa en la Figura 8. Este algoritmo resulta de gran utilidad debido a su fácil implementación y excelentes resultado, si bien, no es el algoritmo más complejo para esta tarea, recordemos que el proyecto se centra en la detección de orden

de lectura, la depuración del algoritmo de segmentación de información se dejará planteado para trabajos futuros.

A demás de implementar el algoritmo LRSA se realizará las adecuaciones pertinentes para que el código sea de utilidad para el proyecto a realizar. El resultado a obtener es el siguiente:

Figura 20

Ejemplo de segmentación.



The ecology of the mimic

by Ed Greenwood

From the Journals of Maerlun the Scholar:

The mimic is a curious (and dangerous) creature to most minds — but few know, or care to know, that there are actually several related species of mimic, divided into two groups: a larger, "killer" variety that is of lesser intelligence, attacking all nearby prey, and does not speak; and the more intelligent, eloquent species which will often bargain for food, has a curious (as yet undiscovered by me) language of its own, and often speaks with the common tongue, dwarvish, or other tongues used in the vicinity of the individual creature's hunting ground.

The mimic gains its name from its ability to alter not only its body shape but the color and texture of the outer surface as well. The color and texture changes are accomplished by the shifting of pigmented liquid between interior and exterior body cells, so that the creature's resemblance to wood or stone in color and texture (depending on whether this pigmented liquid is brought to the outer surface of the creature's body or stored internally).

A mimic is naturally gray in hue, with a smooth, very hard outer skin that gives it the stone-like appearance. The pigmented liquid, brownish in color (often revealed in adventurers when a mimic is wounded in battle), is held within the body in large, muscular organs that serve as both bags and pumps. When these organs are squeezed by the contraction of [illegible] cavity wall—sending they can't flow contents rapidly (within 1 round) into the inner skin layer, filling many capillaries that lie just beneath the skin surface. These capillaries then stand out, brownish and wrinkled, in a pattern resembling wood grain.

Reversing the process, from the wood-grain appearance to the normal state, requires a sort of extreme contraction; a mimic appears to wriggle and twist all over as it empties its capillaries of the liquid. (The creature can, as we all know, alter the external configuration of its form at will, within the limits imposed by the actual volume of its form.) The mimic grows pale-colored, layers of skin beneath the outer one, which is constantly being worn away by the struggles and movement, battle, and feeding.

The mimic is amorphous, and moves in the same way it attacks: by extending strong pseudopods which exude a sticky "glue," and pulling themselves along. A mimic can "unstick" its own glue at any time, and it never sticks to itself.

Reputedly, the glue is sticky enough that only the strongest of adventurers has a good chance of breaking the mimic's hold without killing the creature first. Some adventurers claim to have pulled themselves free from a mimic's glue, but such tales are rare and of questionable veracity is doubtful. (To simulate the possibility of breaking free in game terms, held characters may be allowed to attempt an [illegible] doors" roll based on their strength. Repeated attempts to break free may be made, but any other action — attack or defense — is possible by the victim during the round of attempted escape.)

Mimics are interested only in persons and find an endless quest for food. "Killer" mimics will attack any living creature, regardless of the number of adversaries or their powers. The more intelligent variety often prefers to bargain with an enemy initially — but the creatures have no moral standards as we know them: if an enemy is at last defeated or a bargain has been struck, a mimic will "change its mind," always seeking a meal first and foremost.

Mimics have very sensitive "receptors" (slicks of pigment that are sensitive to sound, light, and vibration) all over their skin. Bright sunlight quenches these sensory spots effectively, hindering the mimic; thus, the creatures are almost always found below ground or in other areas where the sun never reaches. Mimics of all sorts are immune to the delicate effects of alcohol (for, will absorb it if offered, to make use of the inherent nutrients and sugar), and are also unaffected by slime (including green slime), molds (including the brown and yellow varieties), and the corrosive secretions of creatures such as the black pudding, gray ooze, ochre jelly, monolily, and gelatinous cube.

It should be remembered that mimics can travel on walls and ceilings as easily as they can on floors. Those of the more intelligent sort are most adept at "disguising" by assuming the shape of a particular tall, overhanging arch or rough rock wall if they feel threatened. One famous, if somewhat extreme, example: A mimic became part of the arch of the busiest market square of Waterdeep and remained undetected for two winters, until the chronic disappearance of street beggars in the square on every dark night prompted an investigation. A sewer beside this strangely unfamiliar (to the explorers who had "found" the square) datum was discovered to be filled to a depth of more than 60 feet with human and animal bones. (Even after this fact was discovered, the "statue" and two barrooms off the City Watch who prodded it carelessly with their spears, not expecting to find anything.)

Although the details of the concoction are not known to the scribe, it is generally said that the skin of the mimic is useful in the making of a *polymorph (any)* potion. Also, further investigation is needed to determine the range of travel of an individual mimic over a lifetime, and the precise efficacy of the creature's detection organs, which, based on crucial observation, appear to "see" up to 90 feet in subterranean (not total) darkness, and up to 30 feet in the gloom of night... or in darkened areas above ground.

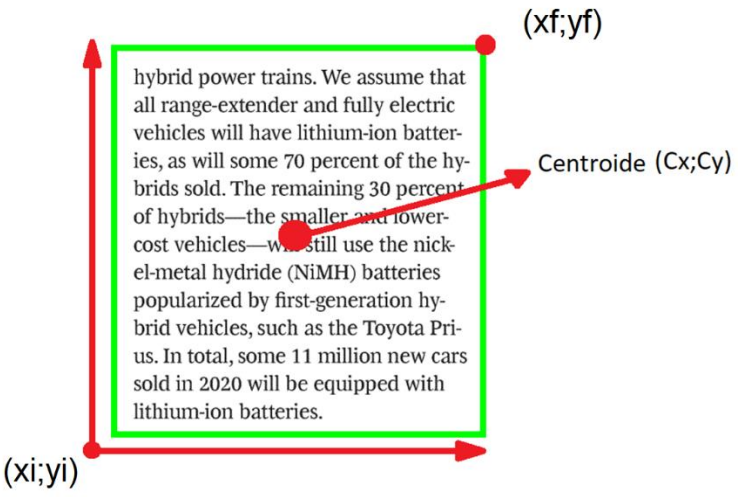
En la Figura 20 los recuadros rojos representan los bloques de información a analizar para determinar el Orden de Lectura.

Extracción de características de los bloques de información

Como se mencionó anteriormente, la implementación del algoritmo LRSA proporciona algunas características de los bloques de información como las coordenadas x e y de los recuadros de información, el área y el centroide, pero estas características no son insuficientes para determinar el orden lógico de lectura. Por lo cual se determinarán características adicionales para ayudar en la identificación del orden de lectura.

Figura 21

Ejemplo de características a extraer mediante el algoritmo de segmentación.

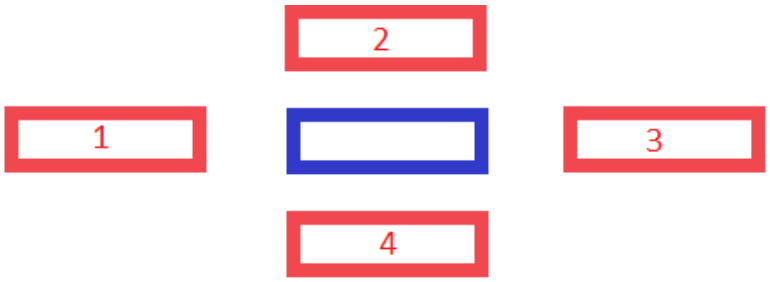


Entre las reglas a aplicar están (Stefano, Domenico, Domenico, & Floriana, 2014):

a. Los componentes adyacentes horizontal o verticalmente son candidatos para ser leídos en consecuencia.

Figura 22

Representación de componentes adyacentes.



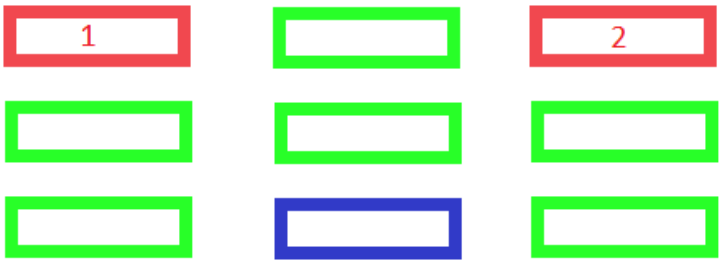
En la Figura 22, se muestra la representación de los componentes donde el bloque azul representa a la componente en análisis, los bloques rojos 1 y 3 representan los componentes adyacentes horizontales y los componentes rojos 2 y 4 son los componentes verticales.

El componente azul puede ser leído después del componente rojo 1 o antes del componente rojo 3 en el caso horizontal, o puede ser leído después del componente rojo 2 o antes del componente rojo 4 en el caso vertical.

b. Un componente en la parte inferior de la página puede ir seguido de un componente en la parte superior de una columna adyacente.

Figura 23

Representación de componentes en columnas adyacentes.

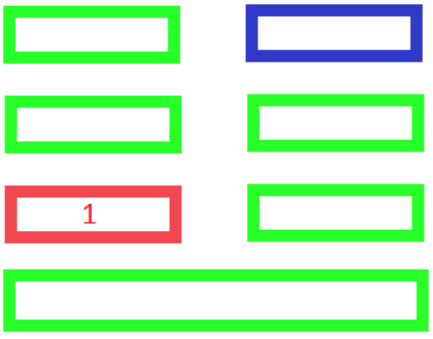


En la Figura 23 se muestra que el componente azul puede ser leído después del componente rojo1 o antes del componente rojo 2.

c. Un componente de la derecha puede ser seguido por un componente de la izquierda en una fila adyacente.

Figura 24

Ejemplo de regla de Orden de Lectura.



Para comprender esta regla se muestra la Figura 24. En este caso particular, el componente azul se debe leer después del componente rojo 1 entendiendo que, el recuadro color verde inferior es un componente que marca una división de página.

A medida que se desarrolle el proyecto se agregarán más reglas con el objetivo de realizar un algoritmo más confiable (Corina & Aldo, 2016).

Bloque de decisión

Una vez que se tenga las características de los bloques de información de deberá realizar un algoritmo que tome esas características y las utilice para generar un orden lógico de lectura.

Una técnica poco estudiada para este tipo de problemática es el uso de Lógica Difusa. La Lógica Difusa es de gran utilidad cuando se requiere contextualizar reglas, en este caso resulta de gran utilidad debido a que se puede definir reglas difusas para identificar qué bloque de información deberá ser leído posteriormente, es decir identificar el orden de lectura.

Una forma de entender la Lógica Difusa es mediante reglas de “Si – Entonces” con la diferencia que este método permite identificar el grado de pertenencia a las condiciones que se establezcan en las reglas.

CAPITULO 3

Desarrollo

Introducción

En el presente capítulo se describe a detalle cada etapa del proyecto, desde el algoritmo la metodología para la digitalización de los documentos hasta el código de identificación de Orden de Lectura. Adicionalmente se explica los cambios y adecuaciones del código con respecto a los códigos que se han encontrado en la bibliografía.

Digitalización de documento PDF a PNG mediante Ghostscript

Los documentos PDF es un formato para “presentar e intercambiar documentos de manera fiable, independientemente del software, el hardware o el sistema operativo. El PDF es ahora un estándar abierto, reconocido por la Organización Internacional para la Estandarización (ISO). Los documentos PDF pueden contener vínculos y botones, campos de formulario, audio, vídeo y lógica empresarial.” (ADOBE ACROBAT, 2021)

Para la elaboración del proyecto se utilizó archivos en formato PDF de una, dos y tres columnas en su estructura lo cuales fueron convertidos imágenes a blanco y negro, una imagen por cada página con una resolución de 500 ppp (píxeles por pulgada) para su análisis. En la Tabla 2 se muestra el número total de páginas utilizadas para cada estructura y el respectivo total.

Tabla 2

Número de páginas de la base de datos

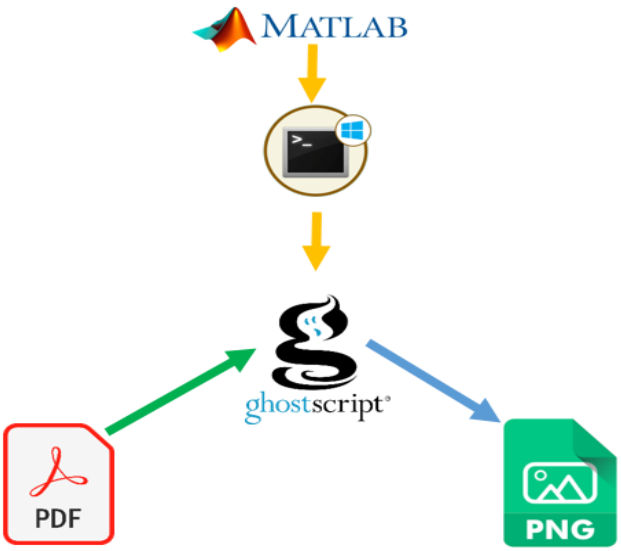
Páginas de Documentos Utilizados	
Una columna	200
Dos columnas	150
Tres columnas	180
Total	**530**

Como se mencionó anteriormente, se utilizó el software Ghostscript® para la conversión de los documentos en formato PDF en imágenes en formato PNG a blanco y negro, pero esto se lo realizó conjuntamente con el software Matlab® debido a la gran cantidad de documentos.

El medio de enlace entre Matlab® y Ghostscript® es CMD de Windows® (Símbolo de sistema), para dar las órdenes de conversión desde Matlab®, Ghostscript® debe ser un programa ejecutable desde CMD. Ver Figura 25.

Figura 25

Flujo de conversión de documentos PDF a PNG.



Pero, ¿qué es Ghostscript?, Ghostscript® es un programa de código abierto que permite interpretar archivos PDF y de lenguaje PostScript el cual es un tipo de lenguaje de descripción de páginas. Dicho programa permite la renderización y conversión de alto nivel mediante dispositivos de salida de vectores. (Artifex, s.f.)

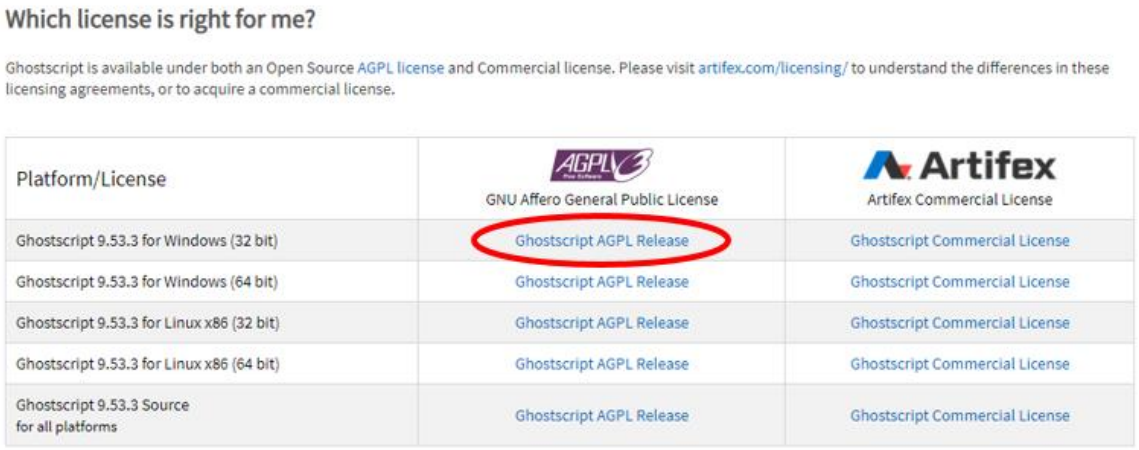
Instalación de Ghostscript

Ghostscript® es *"un intérprete del lenguaje PostScript y archivos PDF. Está disponible en régimen de licencia GPL de GNU Affero o licencia para el uso comercial de Artifex Software, Inc. Ha estado en desarrollo activo durante más de 30 años y se ha adaptado a varios sistemas diferentes durante este tiempo."* (Artifex, s.f.)

El primer paso para usar dicho programa es su instalación, el programa se lo puede descargar desde su sitio oficial[1].

Figura 26

Opciones de descarga de Ghostscript. (Artifex, s.f.)



Which license is right for me?

Ghostscript is available under both an Open Source AGPL license and Commercial license. Please visit artifex.com/licensing/ to understand the differences in these licensing agreements, or to acquire a commercial license.

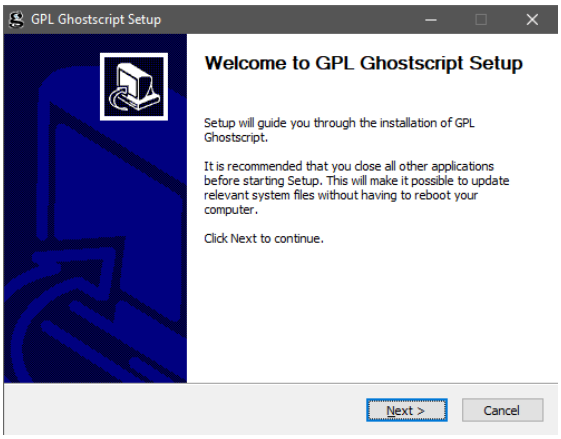
Platform/License	GNU Affero General Public License	Artifex Commercial License
Ghostscript 9.53.3 for Windows (32 bit)	Ghostscript AGPL Release	Ghostscript Commercial License
Ghostscript 9.53.3 for Windows (64 bit)	Ghostscript AGPL Release	Ghostscript Commercial License
Ghostscript 9.53.3 for Linux x86 (32 bit)	Ghostscript AGPL Release	Ghostscript Commercial License
Ghostscript 9.53.3 for Linux x86 (64 bit)	Ghostscript AGPL Release	Ghostscript Commercial License
Ghostscript 9.53.3 Source for all platforms	Ghostscript AGPL Release	Ghostscript Commercial License

En la Figura 26 se muestra las distintas opciones de descarga de Ghostscript, en este caso se utilizó un sistema operativo Windows® de 64 bits, por lo que se escogió la segunda opción de descarga.

[1] https://www.ghostscript.com/download/gsdnld.html

Figura 27

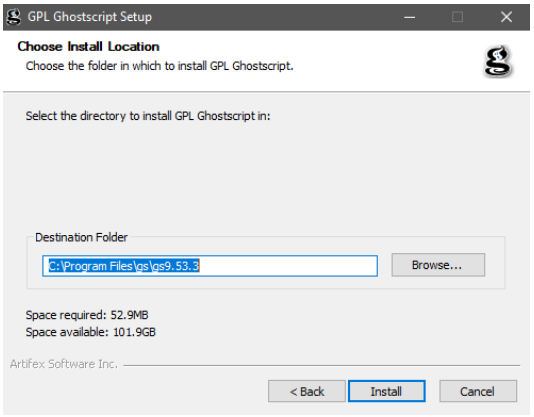
Instalación de Ghostscript.



Al ejecutar el instalador se despliega la pestaña mostrada en la Figura 27 en el cual se procede con la configuración por defecto de la instalación.

Figura 28

Ventana de instalación de Ghostscript.



Una vez culminada la instalación es necesario agregar el programa al Path del sistema para que este pueda ser ejecutado mediante el CMD del

sistema. Este es un paso muy importante ya que, al ser agregado al Path del sistema, el programa podrá ser ejecutado mediante Matlab® lo que facilita en gran medida el procesamiento de los documentos utilizados para hacer las pruebas del algoritmo desarrollado. Para ello se debe realizar los siguientes pasos:

Se debe acceder a “Este equipo” en una pestaña en Windows ® y clickear en el apartado de propiedades, ver Figura 29.

En la Figura 30 se muestra la ventana con la información del sistema, en la cual se debe dar clic en “Configuración avanzada del sistema”.

A continuación se debe ingresar en la opción “Variables de entorno” en la ventana “Propiedades del sistema”. Ver Figura 31.

Se da doble clic en Path y se agrega la dirección en donde se encuentra instalado Ghostscript. Ver Figura 32.

Figura 29

Propiedades de Windows ®.

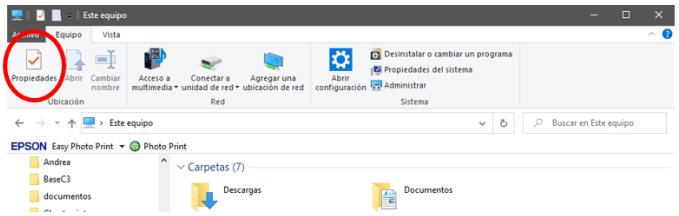


Figura 30

Configuración avanzada del sistema.

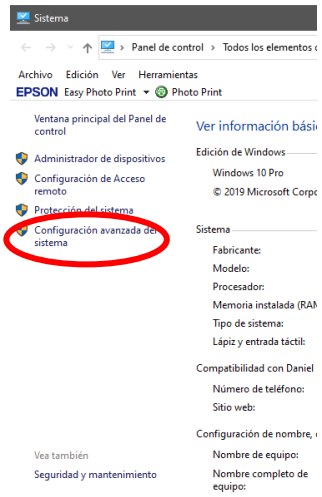


Figura 31

Propiedades del sistema.

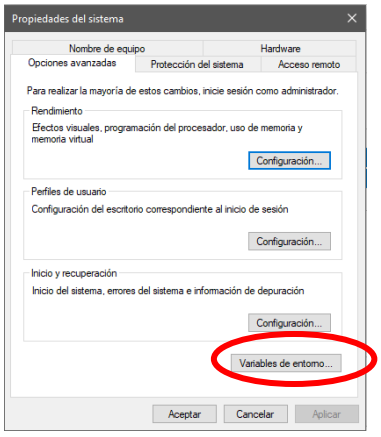
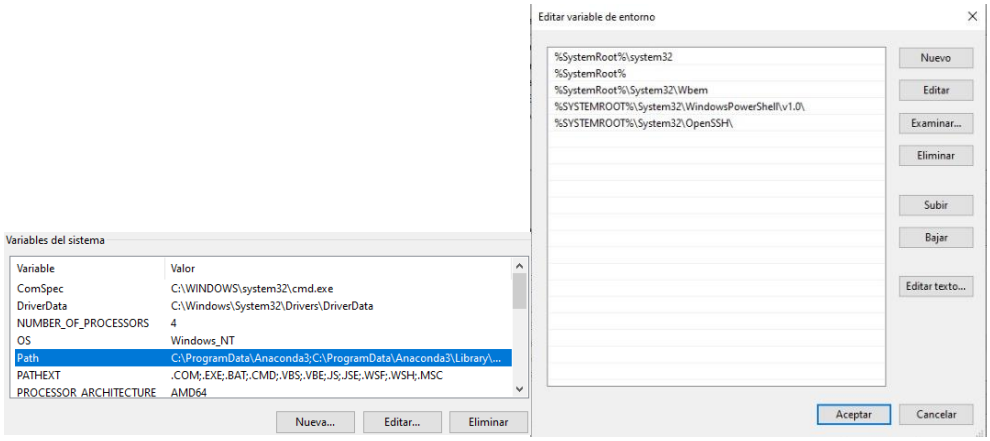
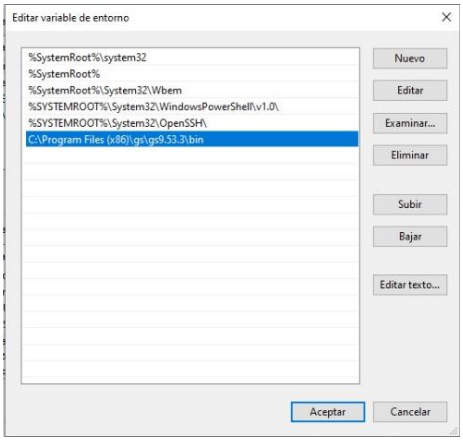


Figura 32

a) Variables del sistema. b) Variables de entorno. c) Ghostscript® como variable del sistema.



a) b)

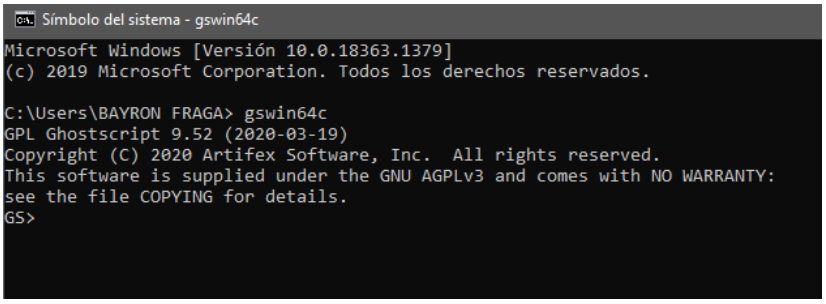


c)

Una vez realizados estos pasos, se puede llamar a Ghostscript® desde el CMD del sistema digitando “gswin32c” o “gswin64c” dependiendo del sistema operativo en el que se esté trabajando. Ver Figura 33.

Figura 33

Llamado a Ghostscript® desde CMD.



Conversión de PDF a PNG

El formato PNG es un formato que permite comprimir imágenes sin pérdidas de información, soporta gran variedad de colores y es ideal para realizar logotipos o cualquier otro tipo de trabajos donde se necesite imágenes con buena resolución. (Baética, 2019)

Para convertir un PDF a PNG por medio de Ghostscript® primero se deben analizar los parámetros que admite el programa.

Los parámetros generales de Ghostscript® son los siguientes:

Figura 34

Estructura general de los parámetros admitidos por Ghostscript.

```
gswin64c [opciones] {nombre de archivo 1} ... [opciones] {nombre de archivo N } ...
```

La estructura que se utilizará para la conversión de los archivos PDF a PNG es la que se muestra en la Figura 35.

Figura 35

Comando general para la conversión de PDF a PNG.

```
gswin64c -dSAFER -dBATCH -dNOPAUSE -sDEVICE=pngmonod -r500 -sOutputFile= destino%00d.png origen.pdf
```

En la siguiente tabla se especifica cada función de los parámetros.

Tabla 3

Descripción de parámetros de Ghostscript.

Parámetro	Función
gswin64c	Llamada a Ghostscript
-d|SAFER	Habilita controles de acceso a archivos (leer, escribir, borrar/renombrar).
-dBATCH	Hace que Ghostscript® se cierre después de procesar todos los archivos.
-dNOPAUSE	Desactiva el mensaje y la pausa al final de cada página.
-sDEVICE	Selecciona el dispositivo de salida debe usar Ghostscript.
pngmonod	Dispositivo de salida a blanco y negro.
-r500	Especifica la resolución horizontal y vertical del dispositivo en píxeles por pulgada.
-sOutputFile	Especifica la dirección de salida del documento.
destino%00d.png	Para renderizar todas las páginas de un documento PDF a imágenes en formato PNG.
origen.pdf	Dirección de origen del documento PDF.

La comunicación directa entre Ghostscript® y Matlab® no es posible, debido a esto se utilizó el CMD de Windows ® y la función "`system`" de

Matlab®, que llama al CMD para ejecutar el comando especificado. (MathWorks, 2021)

Se generó el código en Matlab® que convierte todos los archivos PDF de una carpeta en imágenes PNG a blanco y negro dividido en carpetas, una carpeta por cada archivo PDF. Un ejemplo de resultado de la conversión es el mostrado en la Figura 19.

En la Tabla 2 se muestra el total de páginas convertidas con la ayuda de Matlab®.

Segmentación del documento

Una vez creada la base de datos con la que se realizará las pruebas, se debe implementar un algoritmo de segmentación de imagen para identificar las regiones de información.

Pre - procesamiento

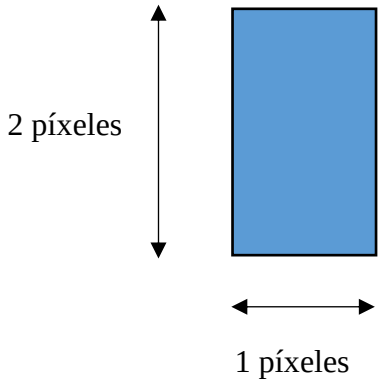
Antes de aplicar cualquier algoritmo es necesario preparar la imagen a ser procesada, en este caso, se busca la eliminación de información que no aporta con el análisis de la imagen.

El primer paso es utilizar un elemento para degradar los rasgos de los caracteres y el posible ruido, representado por puntos producto de la digitalización, con el objetivo de aislar la información y que las regiones de información no se unan entre ellas.

La mayor cantidad de ruido en la imagen tiene un tamaño de 1 píxel cuadrado. Mediante la función "`strel`"[2] se utilizó un elemento estructurador morfológico rectangular de tamaño
[2, 1] píxeles como se muestra en la Figura 36. (MathWorks, 2021)

Figura 36

Dimensiones del elemento estructurador.



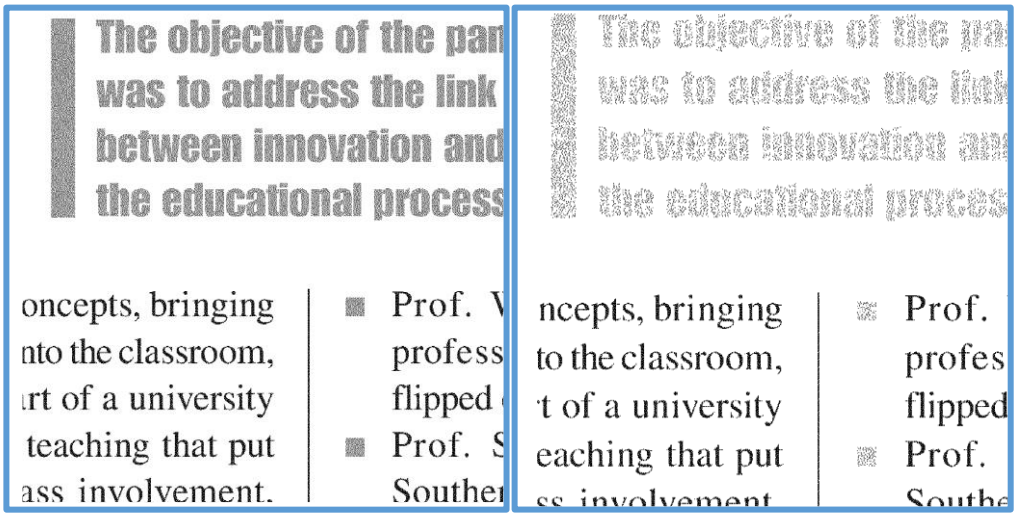
La función de este elemento estructurador es eliminar toda la información que sea menor o igual al elemento conector, esto se lo realiza con la ayuda de la función "`imclose`"[3] de Matlab®.

La función "`imclose`" "realiza el cierre morfológico en la escala de grises o imagen binaria, devolviendo la imagen cerrada". (MathWorks, 2021)

[2] https://es.mathworks.com/help/images/ref/strel.html
[3] https://es.mathworks.com/help/images/ref/imclose.html

Figura 37

a) Imagen original. b) Imagen después de la eliminación de información innecesaria.



a) b)

En la Figura 37 se muestra la imagen antes (a) y después (b) de eliminar la información innecesaria de la imagen.

Discriminación de datos atípicos mediante percentiles

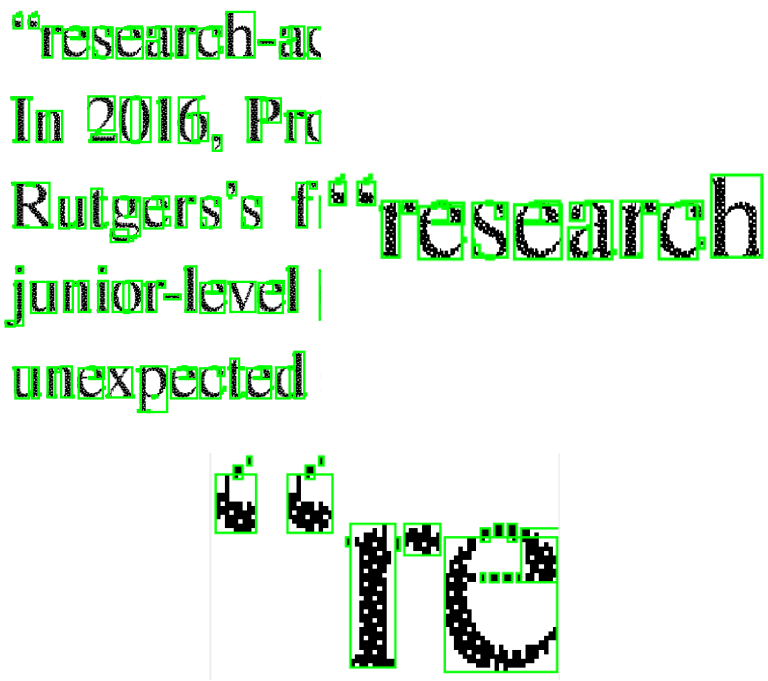
Un dato atípico es un dato que es considerablemente distinto a los demás valores de la muestra, estos datos deben ser retirados de la base de datos para el análisis posterior.

Un dato importante para la elaboración del algoritmo de segmentación es *el Bounding Box* $(BBox)$ de los caracteres y elementos del texto. Mediante la

función "`regionprops`"[4] de Matlab® se puede extraer algunas propiedades de las regiones de la imagen. (MathWorks, 2021)

Figura 38

Bounding Box de los caracteres de la imagen.



Los recuadros verdes de la Figura 38 son generados por la función "`regionprops`" de Matlab®, esta función extrae la siguiente información:

[4] https://es.mathworks.com/help/images/ref/regionprops.html

Tabla 4

Información extraída por la función "`regionprops`"

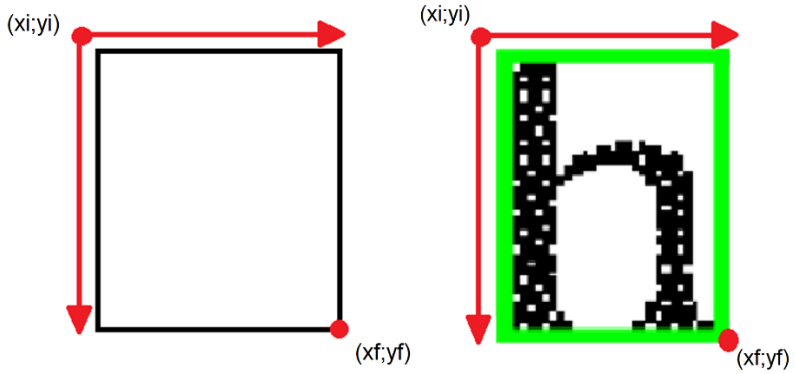
Información extraída por "`regionprops`"	
Área	Devuelve el área de las regiones de información expresada en píxeles
Centroide	Devuelve el centroide de los elementos conectados
Bounding Box (BBox)	Devuelve las coordenadas iniciales y las proyecciones en x e y de las áreas de información

Las coordenadas que se maneja para el procesamiento de las imágenes se muestran en la Figura 39 a) y en la Figura 39 b) se muestra la descripción de coordenadas del $BBox$. Donde $(xi; yi)$ representan las coordenadas iniciales y $(xf; yf)$ son las coordenadas finales.

Para el procesamiento de las imágenes, Matlab® interpreta que los valores positivos de y son hacia abajo y los valores positivos de x hacia la derecha

Figura 39

Descripción de coordenadas. a) Imágenes. b) BBox.



a) b)

Como se puede observar en la Figura 38, el $BBox$ (recuadros verdes) de los elementos de la imagen contiene información que no aporta al análisis, ya sea por su pequeño o gran tamaño, y solo demora el proceso de análisis.

Para esto se realizó una discriminación mediante percentiles de la información extraída del documento, el procedimiento es el siguiente (wikiHow, s.f.):

- Se ordena los elementos de mayor a menor, en este caso se trabajó con el área de los recuadros que contienen la información de la imagen.
- Mediante la función "`prctile`" se extrajo el primer cuartil $(Q1)$ y el tercer cuartil $(Q3)$ de los datos.
- Con estos valores se encontró el rango intercuartil

$$Q3 - Q1 = rango\ intercuartil$$

- Existen dos tipos de límites, los límites internos y externos, debido a la aplicación de este proyecto, los límites internos no fueron de ayuda en la implantación del algoritmo por eso, únicamente se trabajó con los límites externos para la discriminación de la información,
- Para encontrar los límites internos de decisión de eliminación de información se multiplica por 1.5 al rango intercuartil y se lo resta a $Q1$ y se lo suma a $Q3$

$$Q1 - (rango\ intercuartil * 1.5) = Límite\ interno\ inferior$$

$$Q3 + (rango\ intercuartil * 1.5) = Límite\ interno\ superior$$

- Se determina los límites externos de decisión multiplicando el rango intercuartil por 3 y se lo resta a $Q1$ y se lo suma a $Q3$

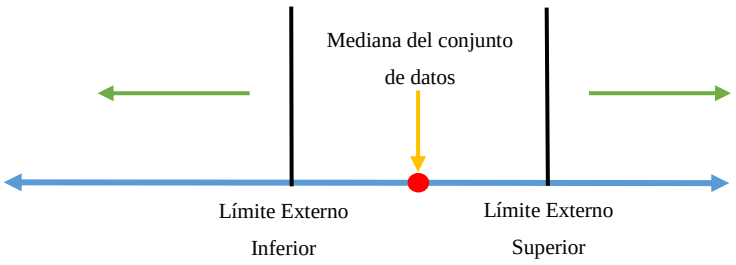
$$Q1 - (rango\ intercuartil * 1.5) = Límite\ externo\ inferior$$

$$Q3 + (rango\ intercuartil * 1.5) = Límite\ externo\ superior$$

- De esta forma se tiene el rango en los que se conservará o eliminará la información.

Figura 40

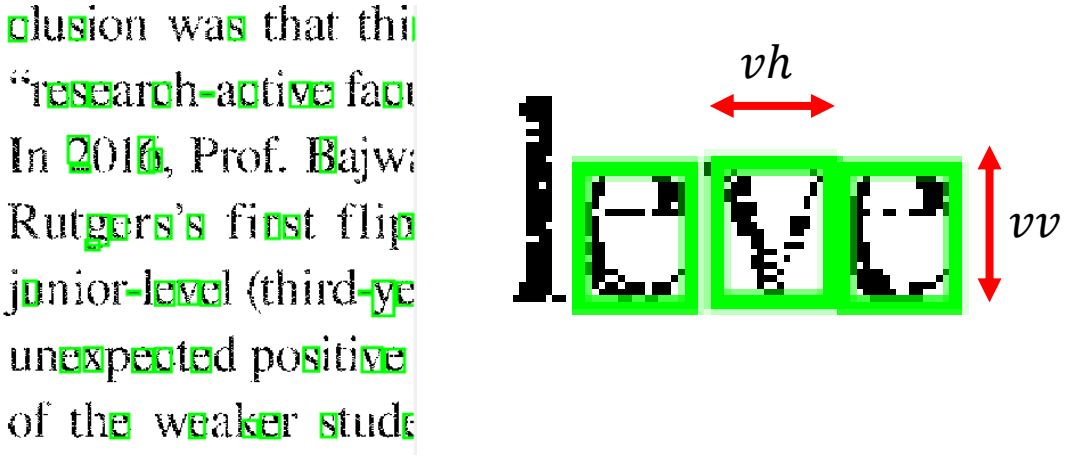
Representación de los Límites de Decisión.



- Los datos que quedan por fuera de los límites externos son eliminados (Figura 40), lo que se intenta con este procedimiento es conservar únicamente los valores de las áreas que definen las características del documento y así, reducir el costo computacional y facilitar posteriores análisis que dependen del Bounding Box. El resultado se muestra en la Figura 41 a).

Figura 41

a) BBox con datos de interés para la elaboración del algoritmo LRSA. b) Tamaño horizontal (vh) y tamaño vertical (vv).



a) *b)*

Si bien, al parecer se pierde mucha información de $BBox$ de los caracteres pero esta información no aporta a los análisis posteriores y se logró disminuir el coste computacional al analizar alrededor de un 5% de los datos originales.

Implementación del algoritmo LRSA

El algoritmo LRSA necesita dos valores de suavizado para realizar el suavizado en el eje x y otro en el eje y, posterior a esto se realiza una operación lógica *"*AND*"* entre las dos imágenes suavizadas para obtener las regiones donde se encuentran la información (HASNAT, 2007).

Es necesario realizar cambios en el algoritmo detallado en la literatura debido a la interpretación de la información que brinda Matlab® es diferente. Matlab® interpreta los datos en negro con ceros y los datos en blanco con unos.

Para la mejor comprensión, en este trabajo se mostrarán las imágenes con las siguientes observaciones:

- El color negro representa secciones donde existe información de la imagen
- El color blanco representa secciones donde no hay información de la imagen.

El algoritmo transforma una secuencia binaria a en una secuencia de salida y de acuerdo a las siguientes reglas:

A. Los unos en a_1 se cambian a ceros en b_1 si el número de unos adyacentes es menor o igual a un límite predefinido C.

B. Los unos en a_1 no cambian en b_1.

Tomando como ejemplo la siguiente secuencia:

$$a_1 = 1\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0$$

Con un valor de $C = 4$, la salida y será igual a:

$$b_1 = 1\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0$$

LRSA Horizontal

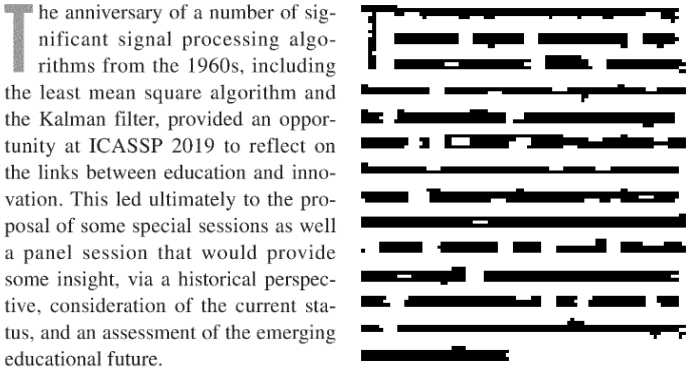
Tomando en cuenta el *Bounding Box* de los caracteres se toma el tamaño horizontal más común de los componentes (vh), ver Figura 41 b).

Se aplica el algoritmo bit a bit a la imagen original en el eje x con un valor de $C = C' * vh$. Donde C' es un entero, mediante prueba y error se determinó que su valor es de 2, es decir:

$$C = 2 * vh$$

Figura 42

LRSA Horizontal.



Si el valor de C es muy grande se uniría regiones de información que no deberían unirse como columnas adyacentes.

LRSA Vertical

Se realiza el mismo procedimiento que el *LRSA* horizontal pero esta vez en dirección con la diferencia que el valor de C dependerá del interlineado del documento (HASNAT, 2007). Los pasos para encontrar el interlineado son:

De los datos del $BBox$ de la imagen se toma el valor de la altura que más se repite (vv). Ver Figura 41 b).

Se divide en 3 la página (Figura 43) para calcular la *distancia media de la línea de texto* ($mtld$).

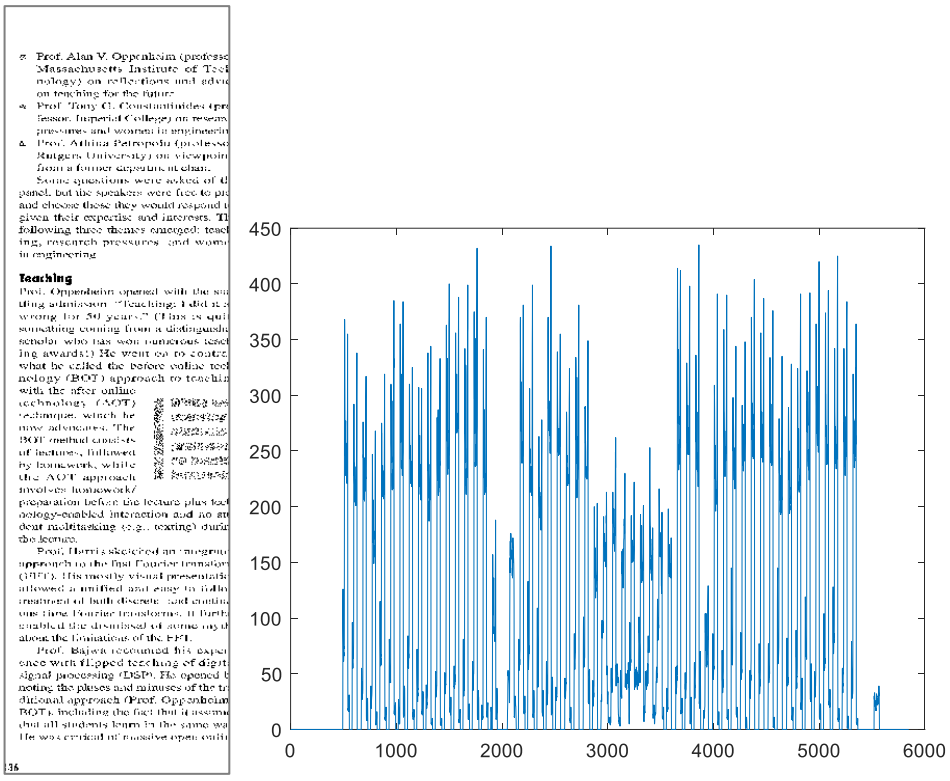
Figura 43

División de la página para su análisis.

Se calcula el histograma horizontal de las componentes para cada división de la imagen.

Figura 44

a) Sección de imagen. b) Histograma horizontal de la imagen.

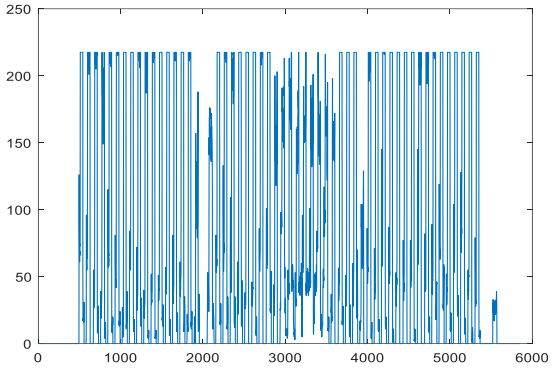


a) b)

Los picos de la Figura 44 son demasiado irregulares para su análisis, mediante la observación se determinó que la información del histograma es regular entre el 60% y 40% del valor máximo del histograma, por esta razón se limitó los valores del histograma a la mitad del valor máximo Figura 45.

Figura 45

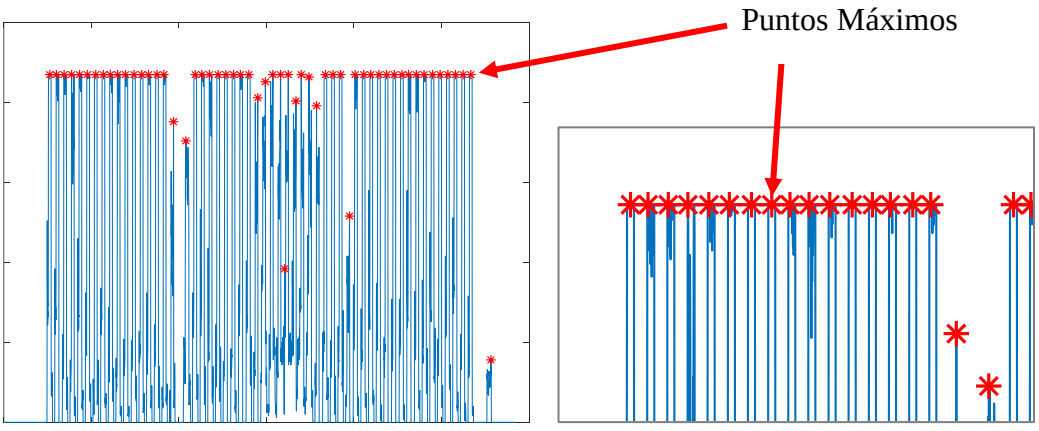
Histograma con valores limitados.



El siguiente paso es determinar los puntos máximos del histograma, estos puntos representan una línea de texto.

Figura 46

Determinación de los puntos máximos del histograma. a) Histograma. b) Zoom de los picos del histograma.



a) b)

La determinación de los picos máximos del histograma se lo realiza con la función "`islocalmax`" de Matlab®, tomando una ventana de

$$C_1 * vv, donde\ C1 = 1.3$$

De igual manera, el valor de $C_1 = 1.3$ se lo tomó experimentalmente. Un valor más grande de C_1 ocasiona pérdidas en los picos del histograma y, para valores más pequeños de C_1 aparecen picos que aportan información errónea al análisis. Ver figura 47.

Se determina la diferencia de los pixeles entre los picos máximos del histograma para cada una de las secciones de la imagen, este es el espacio entre las líneas de texto.

Entre todas las diferencias de los puntos se toma el que más se repite $(mtld)$ y se toma ese valor para realizar el algoritmo LRSA Vertical. El resultado se muestra en la Figura 48.

Figura 47

a) Extracción de picos con C_1<1.3. b) Extracción de picos con C_1>1.3

c) Extracción de picos con C_1=1.3.

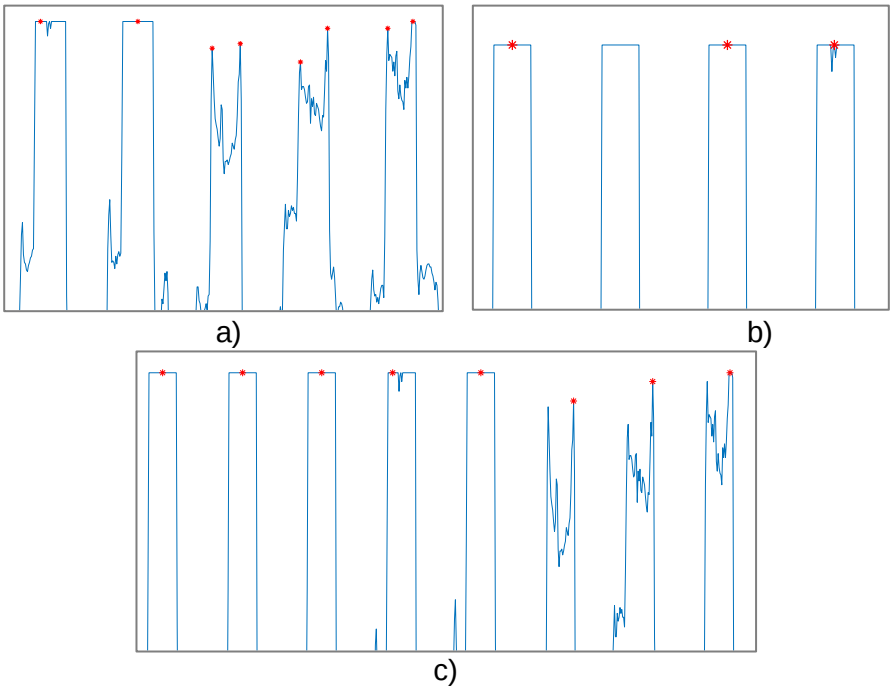
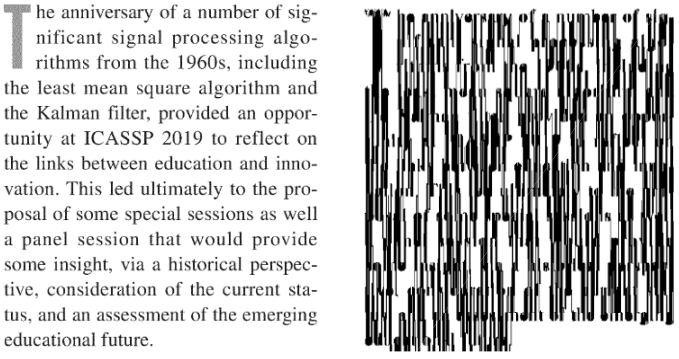


Figura 48

LRSA vertical.



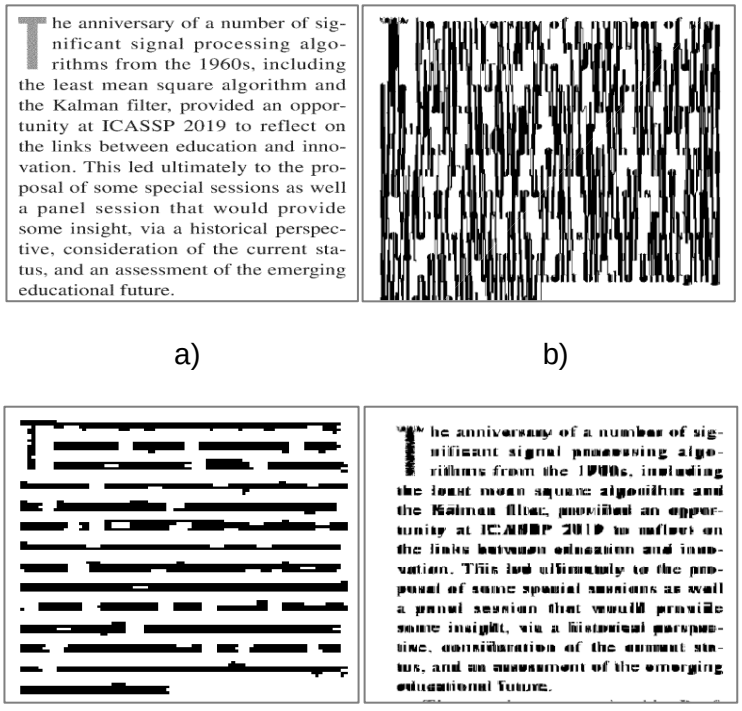
The anniversary of a number of significant signal processing algorithms from the 1960s, including the least mean square algorithm and the Kalman filter, provided an opportunity at ICASSP 2019 to reflect on the links between education and innovation. This led ultimately to the proposal of some special sessions as well a panel session that would provide some insight, via a historical perspective, consideration of the current status, and an assessment of the emerging educational future.

Operación lógica And entre LRSA horizontal y LRSA vertical

Bit a bit se realiza la operación lógica "AND" entre las imágenes LRSA horizontal y vertical. El resultado es el mostrado en la Figura 49.

Figura 49

a) Imagen Original. b) LRSA Vertical. c) LRSA Horizontal. d) AND entre LRSA Horizontal y Vertical.



Si bien, el resultado no es similar a la bibliografía, con este algoritmo se logró normalizar el tamaño de los caracteres. Ver Figura 50.

Figura 50

a) BBox de la imagen. b) BBox de la imagen normalizada.

a)

b)

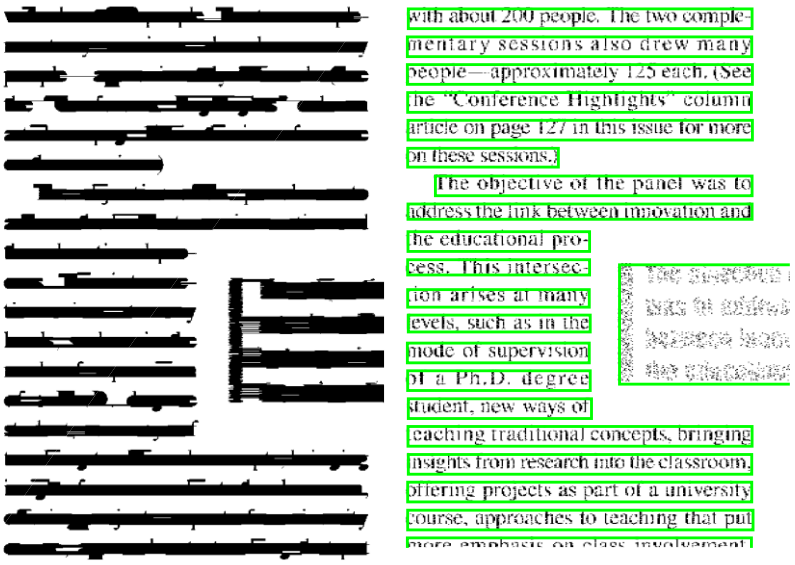
Este paso es de gran utilidad porque con estas nuevas dimensiones de los caracteres se puede realizar un suavizado horizontal para unir las líneas de texto sin que se unan los caracteres de distintas columnas.

Análisis de Componentes Conectados

Una vez que se realizó el normalizado de los elementos de la imagen se realizó un suavizado horizontal. Mediante prueba y error se determinó que el valor para el elemento conector horizontal es de 2.5 veces el valor horizontal más probable de los caracteres normalizados. El resultado de utilizar el elemento conector con el comando "`imopen`" de Matlab® es el mostrado en la Figura 51. (MathWorks, 2021)

Figura 51

a) Suavizado horizontal de la imagen. b) BBox del suavizado horizontal.



a) b)

Análisis de las características del BBox de los elementos

Como se explicó anteriormente, la función "`regionprops`" devuelve los valores: área, centroide y $BBox$ de las regiones de información, pero esta información debe ser normalizada debido a los distintos tamaños de las imágenes.

La normalización de la información se lo realizó a 1000 x 1000 píxeles, es decir, los valores máximos de la información extraída únicamente podrán llegar a ser de 1000 píxeles tanto en x como en y. El objetivo de esta conversión es trabajar con valores similares para cualquier imagen, reducir el peso de la información y lograr un procesamiento más eficiente.

Una vez realizada la conversión se agregaron nuevos valores a esta tabla como: un identificativo a cada uno de los recuadros del $BBox$, el tamaño original de la imagen en x $(Torx)$, el tamaño original en y $(Tory)$ y el valor de vv y vh.

En la tabla 5 se muestra el formato de almacenamiento de las características del documento.

Tabla 5

Matriz con los datos extraídos de la imagen.

Área	**Centroide**		**Bounding Box**				**vv**	**vh**	**id**	**Torx**	**Tory**
Valor 1	Valor 1 en x	Valor 1 en y	Coordenada 1 en x	Coordenada 1 en y	Distancia 1 en x	Distancia 1 en y	Valor	Valor	1	Valor	Valor
Valor 2	Valor 2 en x	Valor 2 en y	Coordenada 2 en x	Coordenada 2 en y	Distancia 2 en x	Distancia 2 en y	Valor	Valor	2	Valor	Valor
Valor 3	Valor 3 en x	Valor 3 en y	Coordenada 3 en x	Coordenada 3 en y	Distancia 3 en x	Distancia 3 en y	Valor	Valor	3	Valor	Valor
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.

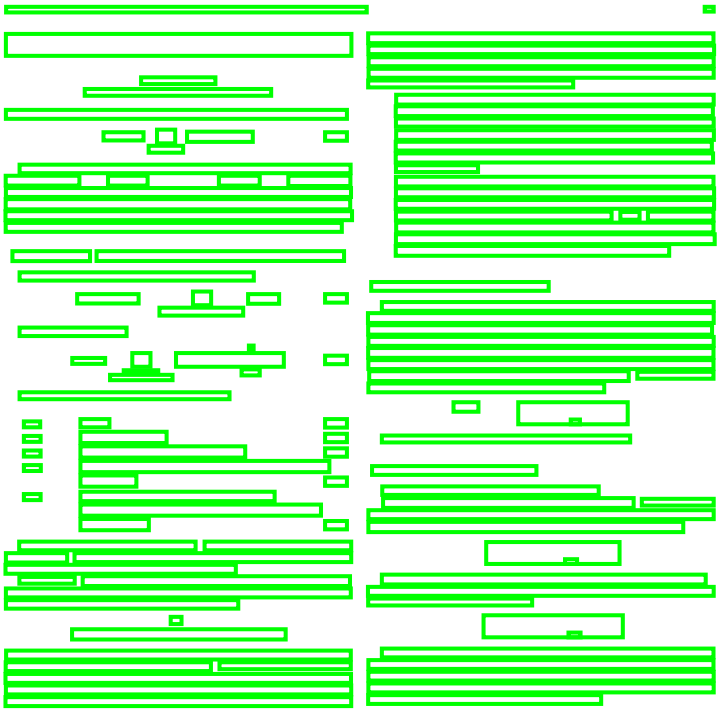
Este procedimiento se lo realizó para cada una de las 530 imágenes generadas por Ghostscript® generando así la base de datos con la que se va a trabajar. La información de cada imagen se almacenó en un archivo en formato “mat” para su posterior análisis.

Redimensionamiento BBox de las líneas de texto

Una vez extraída la información de todos los documentos, se debe analizar esta información debido a que no todos los elementos sirven para determinar el orden de lectura. Un ejemplo de esto se muestra en la Figura 52.

Figura 52

Ejemplo de la información obtenida una imagen de un PDF.



El primer paso que se realizó es determinar el valor de la altura que más se repite (moda) de los *BBox* y se eliminó a los recuadros que poseen una altura inferior al 80% de moda de todos los recuadros. Con esto se busca eliminar los recuadros de posibles caracteres aislados o eliminar información irrelevante. Ver figura 53.

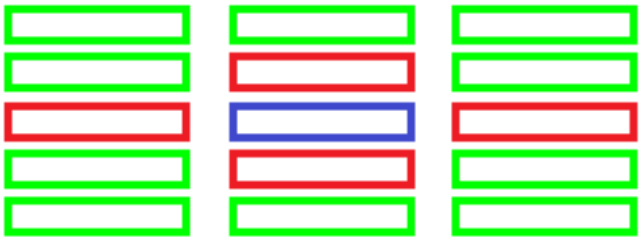
Figura 53

BBox de los elementos después de la eliminación.

Antes de realizar cualquier análisis se debe explicar lo que se entiende por “vecindad de un recuadro”. La vecindad de un recuadro se conforma con los recuadros más cercanos a él en cuatro direcciones: arriba, abajo, izquierda y derecha. Ver Figura 54.

Figura 54

Ejemplo de vecindad. Donde el color azul representa al recuadro a analizar, el color rojo son sus vecinos y el color verde son los demás elementos.



Explicado esto, se implementó un algoritmo que permite identificar los recuadros segmentados entre las líneas de texto (Figura 55. a)) y unirlos (Figura 55. b)), para ello se analizó la vecindad de cada y se tomaron las siguientes consideraciones:

- Los recuadros que tengan en común el vecino superior y el vecino inferior se leerán consecutivamente de izquierda a derecha si son vecinos entre sí.
- Los recuadros que cumplan la condición anterior serán eliminados y se creará un solo recuadro con las coordenadas iniciales del recuadro más a la izquierda y las finales del recuadro más a la derecha.
- Los recuadros que tengan en común su vecino superior e inferior y que no sean vecinos entre sí no se modificarán.
- Los recuadros que tengan en común el vecino superior o el vecino superior no se modifican.
- Los demás recuadros fuera de esas condiciones no se modifican.

El resultado de la implementación de estas condiciones se muestra a continuación.

Figura 55

a) Identificación de recuadros que cumplen las condiciones. b) Unión de recuadros.

a) b)

El siguiente tratamiento que se le dio a los datos es la eliminación de recuadros consecutivos teniendo en cuenta las siguientes consideraciones:

- Los recuadros que tengan en común el vecino superior o el vecino inferior y la distancia con uno de ellos sea menor al 10% de la moda de las líneas de texto se leerán consecutivamente de izquierda a derecha, si son vecinos entre sí.
- Los recuadros que cumplan la condición anterior serán eliminados y se creará un solo recuadro con las coordenadas iniciales del

recuadro más a la izquierda y las finales del recuadro más a la derecha.

- Los recuadros que tengan en común su vecino superior e inferior y que no sean vecinos entre sí no se modificarán.
- Los demás recuadros fuera de esas condiciones no se modifican.

Los resultados se muestran en la Figura 56.

Figura 56

a) Identificación de recuadros que cumplen las condiciones. b) Unión de recuadros.

a) b)

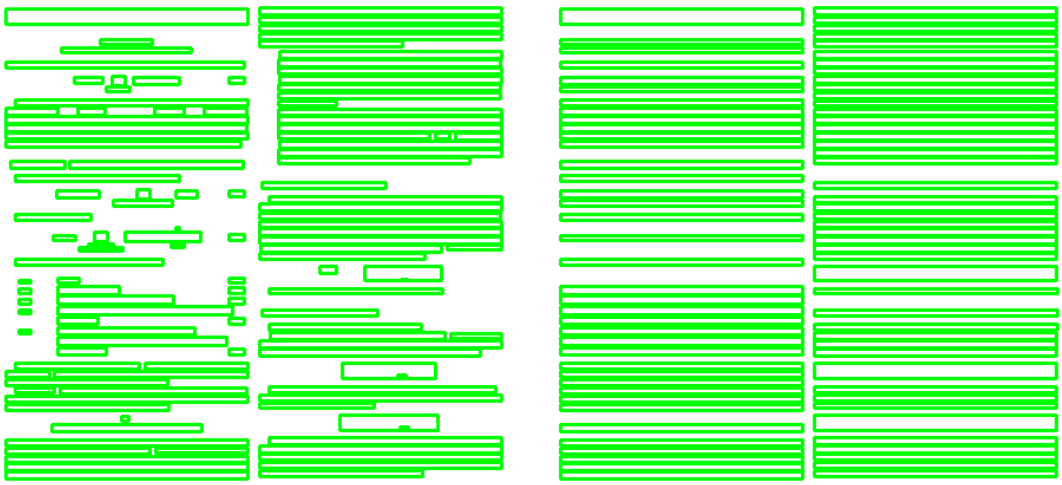
El último tratamiento que se le dio a los recuadros es el redimensionamiento según las siguientes condiciones.

- El recuadro en análisis asumirá el tamaño en x de su vecino superior en el caso de que exista y en el caso de que este tenga mayor tamaño en x.
- Los recuadros fuera de esta condición no se modifica.

La comparativa entre los $BBox$ originales y los $BBox$ finales se muestra a continuación.

Figura 57

a) BBox de información Originales. b) BBox de información después del procesamiento.



a) b)

Haciendo un pequeño resumen, hasta el momento se tiene las siguientes características de los recuadros de información:

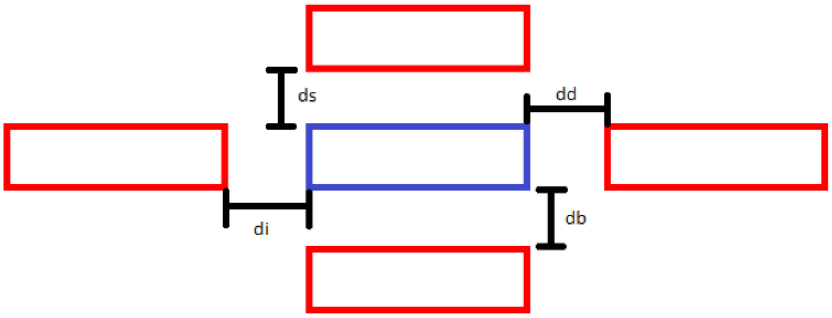
- Área de los recuadros de información.
- Centroide de los recuadros.

- $BBox$ (Coordenadas iniciales en x, y con su respectivo desplazamiento).
- La moda de la altura de los recuadros (vh).
- La moda del ancho del $BBox$ de los caracteres.
- El tamaño original de la imagen.
- Tamaño de todas las imágenes de 1000x1000 pixeles.
- Tamaño original de la imagen antes de la normalización.

Además, con el algoritmo desarrollado en la sección 3.2.4.1. Se identificó la vecindad para cada uno de los recuadros de información así como también su distancia en píxeles hacía los elementos de la vecindad.

Figura 58

Distancia entre los vecinos de los recuadros de información.



Donde:

- ds: distancia del recuadro con su vecino superior
- di: distancia del recuadro con su vecino izquierdo
- dd: distancia del recuadro con su vecino derecho
- db: distancia del recuadro con su vecino inferior

Desarrollo del algoritmo de detección de lectura

Lo que realizó en este trabajo es, diseñar un “Clasificador Difuso” que se encarga de ponderar las cadenas de texto o de información dependiendo de sus variables de entrada; el Orden de Lectura dependerá de esta ponderación siendo las cadenas de texto o información con mayor puntuación las candidatas a ser leídas primero, el proceso se lo detalla a continuación.

Identificación y redimensionamiento de divisores de página

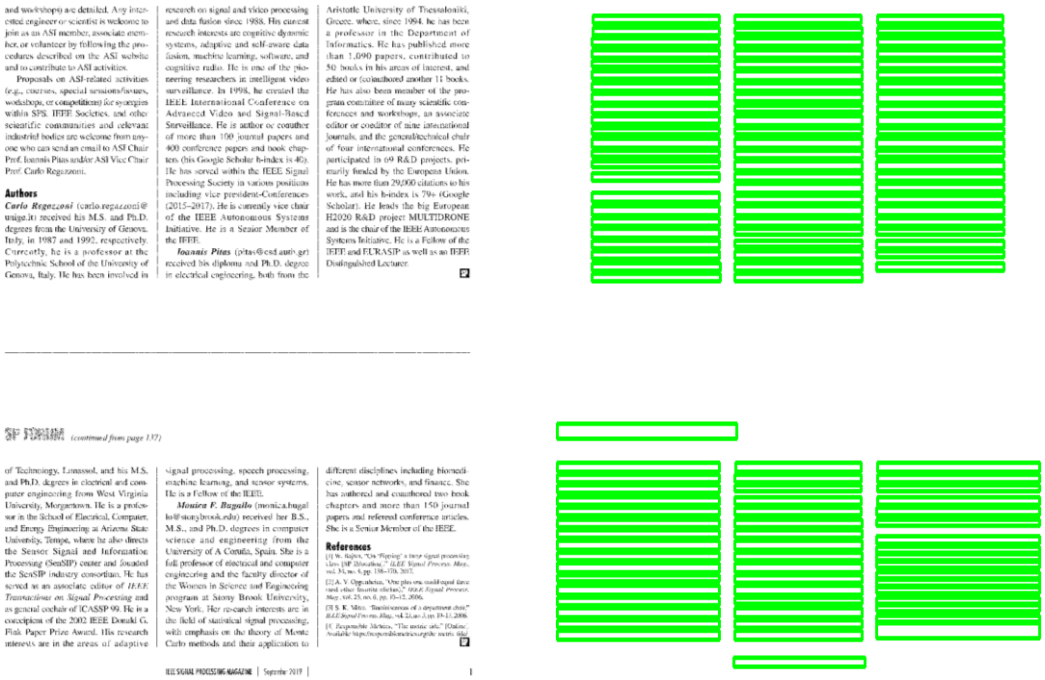
Lo primero que se consideró para realizar el algoritmo de detección de Orden de Lectura es realizar una prolongación de los $BBox$ aislados, durante el trabajo realizado se evidenció que estos $BBox$ son posibles pies, encabezados o divisores de página.

En la Figura 59 se observa que el título intermedio define una división de página, por esta razón, lo que se encuentra debajo se debe tratar como si fuera una página nueva. Para definir si es un título o no se tomó en cuenta la distancia media de la línea de texto $(mtld)$, mediante experimentación se determinó que si cumple las siguientes condiciones se puede considerar que es un encabezado, pie o divisor de página:

- Si un $BBox$ se encuentra a una distancia 2 veces el $mtld$ de su vecino inferior y superior es un divisor de página y,
- No posee vecinos a su izquierda o derecha.

Figura 59

Análisis de BBox Aislados. a) Imagen original. b) BBox de la página.

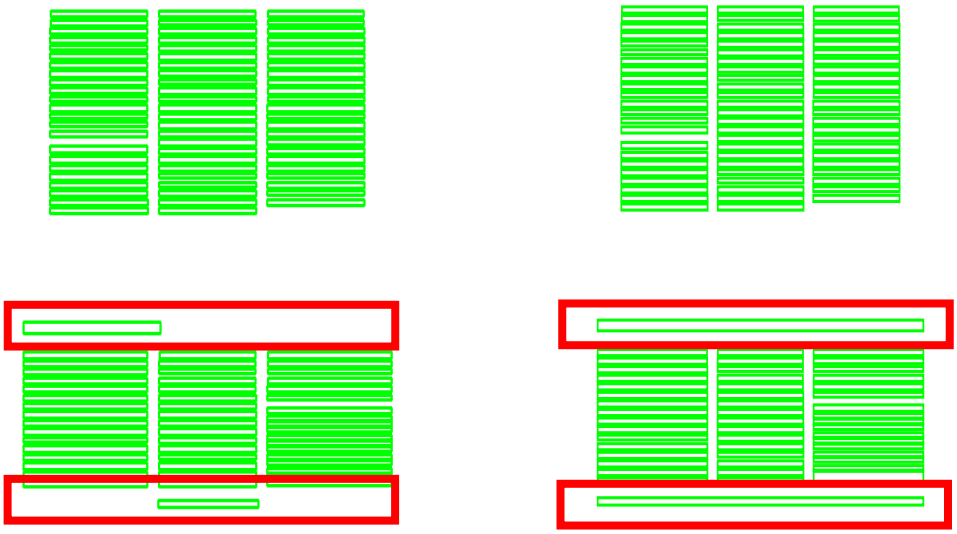


a) b)

Si un $BBox$ cumple estas condiciones deberá ser redimensionado hasta los extremos horizontales de la página para dejar en claro la división. Ver Figura 60.

Figura 60

a) BBox de la imagen. b) BBox del redimensionamiento.



a) b)

Sin ninguna duda, los encabezados se leen primero y los pie de página al final del documento. Con esta consideración, mediante el análisis de las de las distintas imágenes se determinó que:

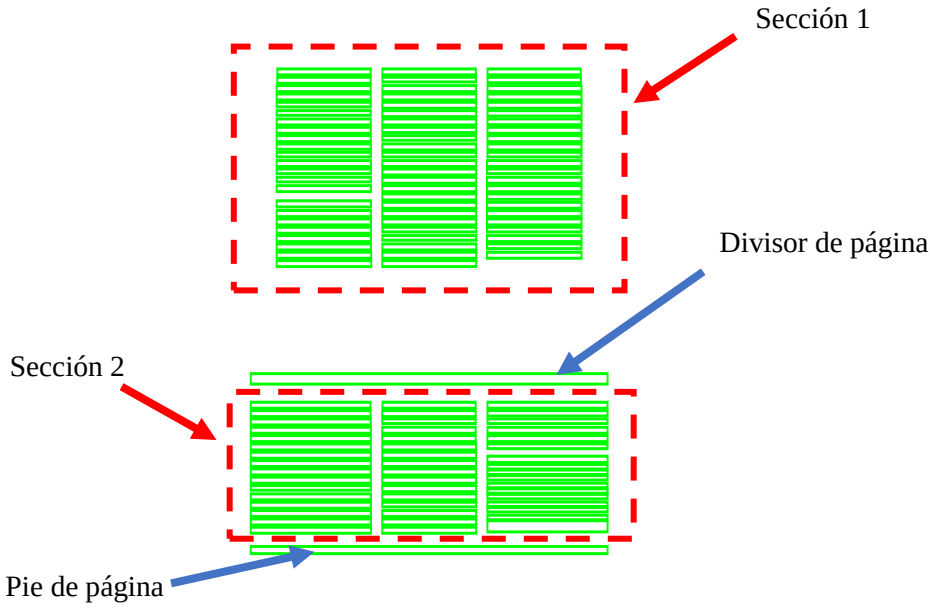
- Si el $BBox$ ubicado más arriba de la página se distancia más de 2 veces el vv con su vecino inferior, dicho $BBox$ y sus vecinos horizontales serán considerados como el encabezado de la página y se leerán primero.
- Si el $BBox$ ubicado abajo de la página se distancia más de 2 veces el vv con su vecino inferior, dicho $BBox$ y sus vecinos horizontales serán considerados como pie de página y se leerán al final de la página.

Después de redimensionar los $BBox$ e identificar el encabezado y pie de página se divide la página cuando un recuadro ocupa más del 80% del ancho de la imagen, es decir, 800 píxeles para realizar un análisis por separado de la imagen.

Con estas consideraciones, la imagen de la Figura 60 puede ser analizada en dos secciones debido al $BBox$ divisor, el elemento inferior es definido como pie de página. Ver Figura 61.

Figura 61

Seccionamiento de página.

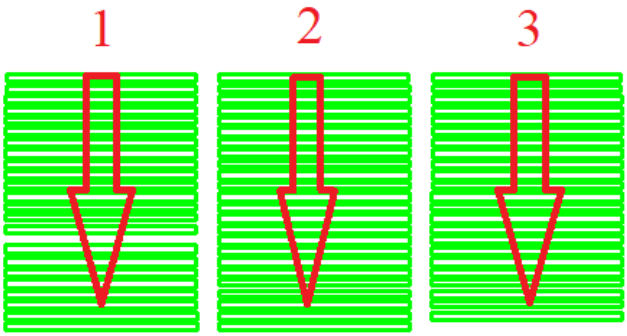


El divisor de página siempre será leído al final de la sección superior, en este ejemplo, el elemento divisor se leerá al finalizar la sección 1.

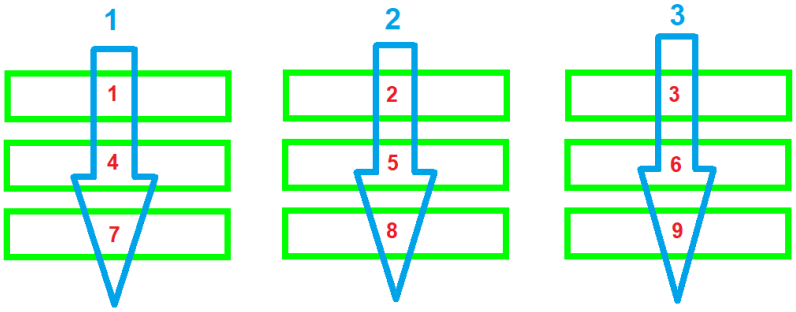
El siguiente paso es generar cadenas de texto con los $BBox$ adyacentes superiores e inferiores de la página divida.

Figura 62

Análisis de las cadenas de texto la página dividida.



a)



b)

En la Figura 62 a) se muestra un ejemplo de los *BBox* de la página segmentada, para comprender cómo se generan as cadenas de texto, en la Figura 62 literal b se muestra un ejemplo de cómo se etiquetan los *BBox*, Con estas

etiquetas se analiza los vecinos superior e inferior de cada columna y se genera la cadena de texto, de esta forma se tendría:

- Cadena 1 conformada por los *BBox* 1, 4 y 7.
- Cadena 2 conformada por los *BBox* 2,5 y 8.
- Cadena 3 conformada por los *BBox* 3, 6 y 9

A primera vista se puede identificar que la cadena de texto 1 de la Figura 62 se debe leer primero, posterior la cadena 2 y por último la cadena 3. Aquí es en donde entra la Lógica Difusa.

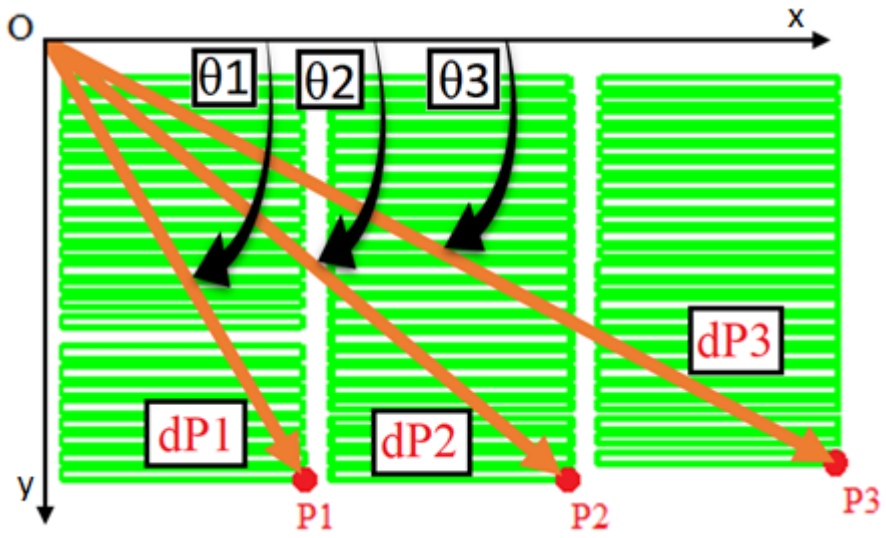
Variables de entrada del Sistema de Control Difuso

En la presente sección se detallará las variables de entrada y salida del sistema de Control Difuso como también el tratamiento que se les va dar a cada una de ellas para generar el Orden de Lectura del documento.

Mediante distintas pruebas se determinó que la mejor opción, como variables de entrada, es utilizar la distancia y ángulo que forman las coordenadas finales de cada cadena de texto.

Figura 63

Representación de las variables de entrada del Sistema de Control Difuso.



En la Figura 63 se tiene los siguientes elementos:

- $\theta 1, \theta 2$ y $\theta 3$ son los ángulos que forman las coordenadas con el eje x,
- $P1, P2$ y $P3$ son las coordenadas finales de las cadenas de caracteres
- $dP1, dP2$ y $dP3$ son las distancias desde el origen a cada una de las coordenadas.

Ahora, es necesario establecer las Variables Lingüísticas necesarias para la Fuzzificación de las Variables de Entrada.

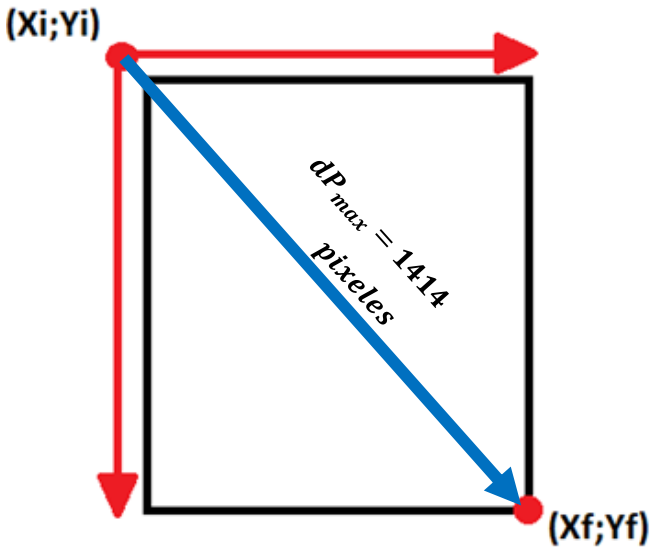
El primer paso para determinar las Variables Lingüísticas, es establecer los universos de discurso para cada una de las variables.

Universo de Discurso, Conjuntos Difusos y Funciones de Membresía de la variable de entrada "dP"

Para determinar el universo de discurso de la variable "dP" se debe analizar las coordenadas del plano donde se presentan las imágenes (Figura 64).

Figura 64

Universo de discurso para dP.



En este caso el valor máximo en pixeles que puede tomar un componente es:

$$dP_{max} = \sqrt{(coordenada\ x)^2 + (coordenad\ y)^2}$$

$$dP_{max} = \sqrt{(1000)^2 + (1000)^2}$$

$$dP_{max} = 1414{,}2 \approx 1414\ pixeles$$

Por lo explicado anteriormente, el universo de discurso para "dP" es de 0 a 1414 pixeles.

Una vez definido el universo de discurso, se debe establecer los Conjuntos Difusos con sus Variables Lingüísticas y sus Funciones de Pertenencia.

Para definir los Conjuntos Difusos de la variable "dP" se debe tomar en cuenta que; las coordenadas de las cadenas de texto más cercanas al origen tienen preferencia de lectura ante coordenadas más lejanas del origen.

Por esta razón, se definieron 3 variables lingüísticas las cuales van a ser ponderadas de la siguiente manera:

- Variable Lingüística: Grande (G) representa a las distancias “más cercanas” al origen.
- Variable Lingüística: Medio (M) representa a las distancias que se encuentran en la “mitad” de la distancia máxima. .
- Variable Lingüística: Pequeño (P) representa a las coordenadas “más lejanas” del origen.

Mediante prueba y error se determinó los valores de los Conjuntos Difusos y sus Funciones de Membresía.

El presente proyecto amerita una respuesta lineal debido a que no existen respuestas transitorias y no depende de variables continuas como el

tiempo. Por esta característica solo se usarán Funciones de Membresía triangulares y trapezoidales.

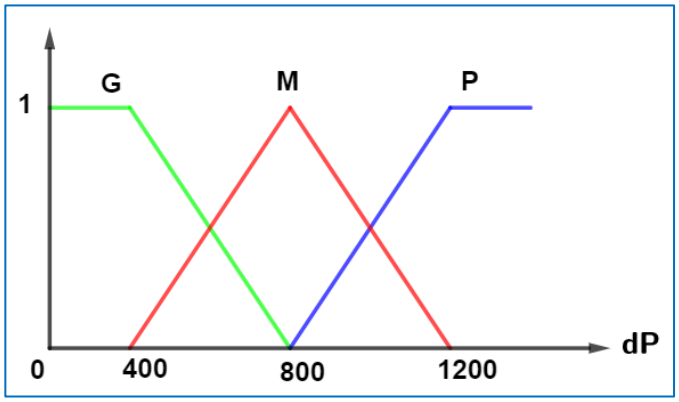
Mediante ensayo y error se determinaron os siguientes Conjuntos Difusos:

- Variable Lingüística Grande (G): Se consideró que la una distancia dP_n va a estar “cerca” del origen cuando esta distancia sea menor a 400 pixeles, a partir de este punto su valor de pertenencia decaerá y se dejará de considerar “cercano” cuando la distancia sea de 800 pixeles.
- Variable Lingüística Medio (M): Se considera que la distancia es “media” cuando se encuentre entre los intervalos de 400 a 1200 pixeles.
- Variable Lingüística Pequeño (P): Se considera que la distancia está “lejos” del origen cuando la distancia sea más de 1200 pixeles.

Todas las especificaciones anteriores se muestran en la Figura 65.

Figura 65

Representación gráfica de los Conjuntos Difusos para la variable dP.

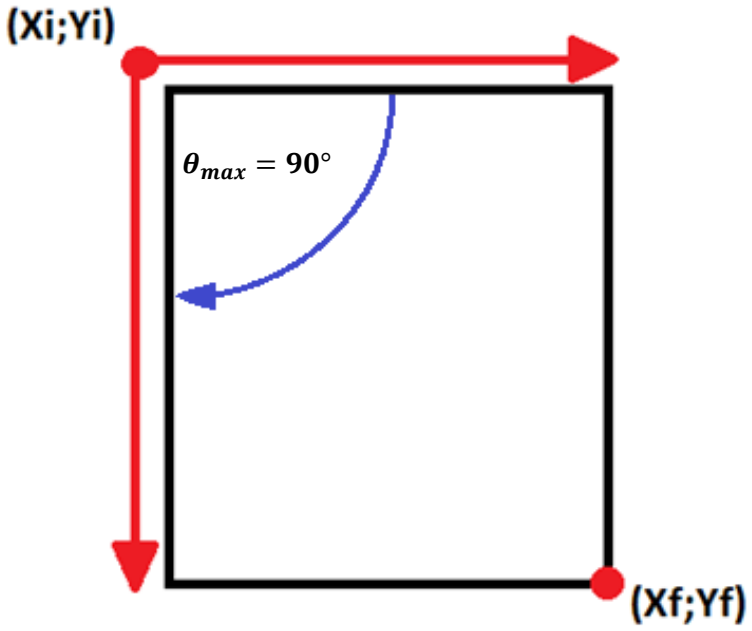


Universo de Discurso, Conjuntos Difusos y Funciones de Membresía de la variable de entrada "θ"

Como se observa en la Figura 66, los valores que puede tomar la variable "θ" van desde los $0°$ a los $90°$, cabe recordar que los valores de y positivos se representan hacia abajo.

Figura 66

Universo de discurso para la variable θ.



En la Figura 66 se observa claramente que el máximo valor que admite "θ" es de $90°$ por ende, el universo de discurso va desde los $0°$ a $90°$

Los Conjuntos Difusos se determinaron según el ángulo que forman las coordenadas con el eje x positivo, las cadenas de caracteres que formen un ángulo cercano a $90°$ son candidatas a ser leídas primero, de la misma manera, las componentes que formen un ángulo cercano a $0°$ son candidatas a ser leídas al final.

De igual forma, se definieron 3 variables lingüísticas,

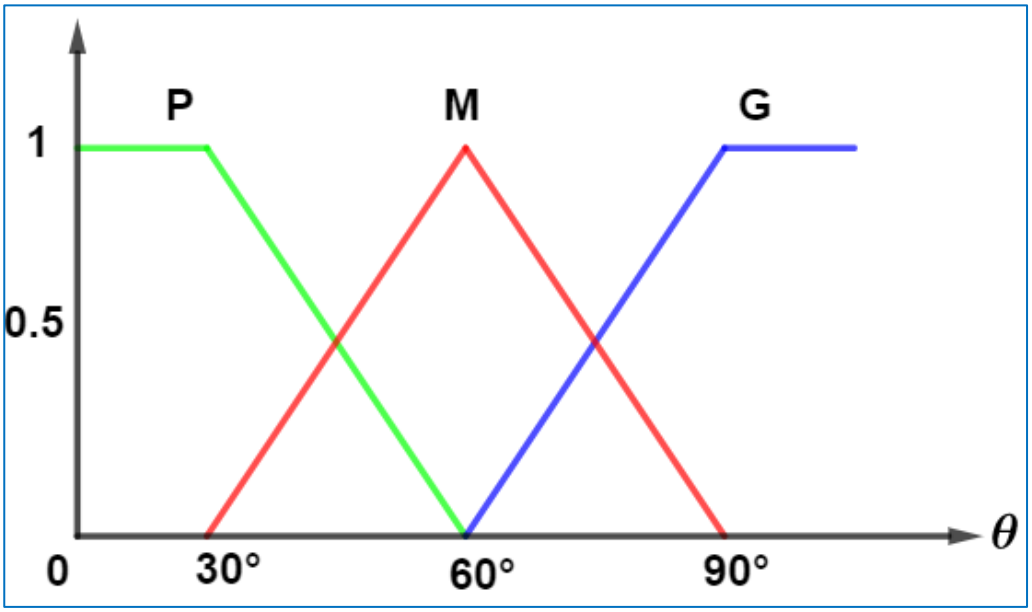
- Variable Lingüística: Grande (G) representa ángulos “próximos” al $90°$.
- Variable Lingüística: Medio (M) representa ángulos que se encuentran en la “mitad” del universo de discurso.
- Variable Lingüística: Pequeño (P) representa ángulos “cercanos” a $0°$.

Su ponderación se la realizó mediante prueba y error con la ayuda de la aplicación *Fuzzy Logic Designer* de Maltab; esta aplicación sirve para diseñar los Conjuntos Difusos de las variables de entrada y de salida.

A diferencia de la variable "dP", los Conjuntos Difusos de θ no se pueden, o es muy complicado deducir sus elementos, una tentativa opción para los valores centrales de las Funciones de Membresía son los valores de $30°$, $60°$ y $90°$.

Figura 67

Valores centrales iniciales de la Funciones de Membresía de θ.

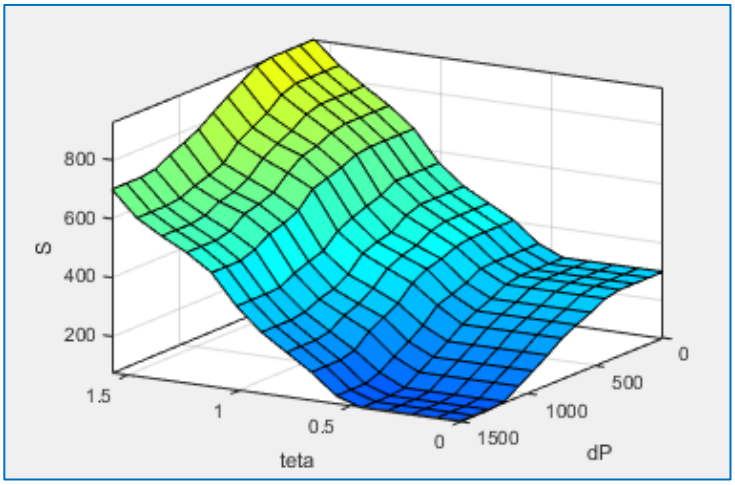


Para realizar las pruebas de las Funciones de Membresía es necesario establecer las Reglas de Control que se explicarán posteriormente.

Mediante la herramienta *"Fuzzy Logic Designer"* de Matlab®, las Reglas de Control y con las Funciones de Membresía de la Figura 67, se obtuvo la Superficie de Control de la Figura 68. Se probó esta Superficie de Control pero los resultados no fueron satisfactorios.

Figura 68

Valores iniciales de los Conjuntos Difusos de "θ".



Debido al bajo rendimiento del Clasificador Difuso y con la ayuda de la herramienta *"Fuzzy Logic Designer"* se rediseñó las Funciones de Membresía para la variable "θ", tomando en cuenta qué:

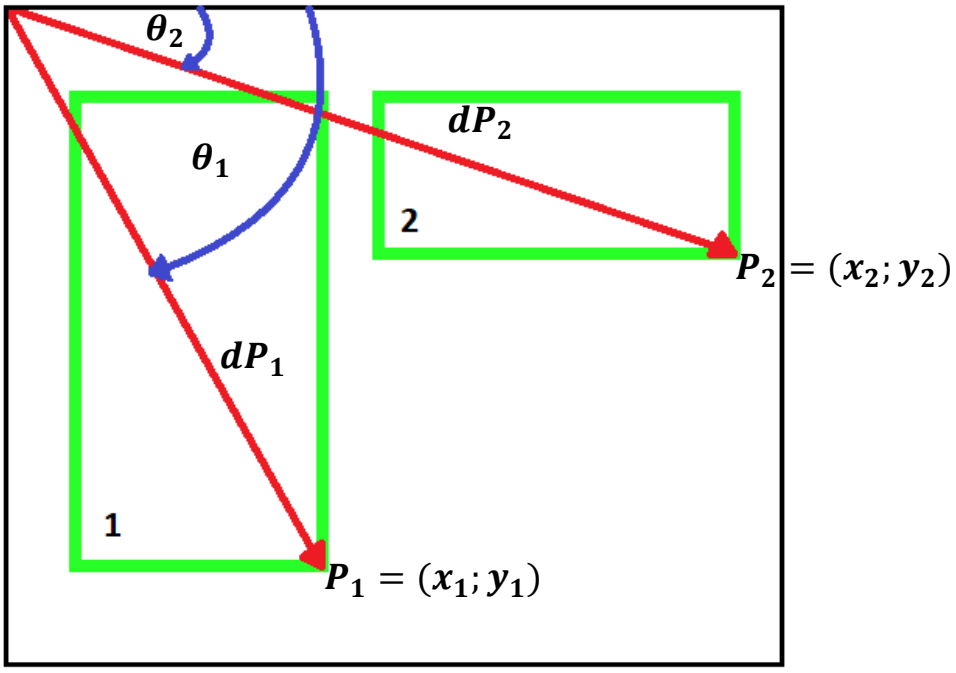
Las cadenas de texto cuyas coordenadas formen un ángulo "cercano" a $90°$ deberán leerse antes que una cadena de texto cuyo ángulo sea "cercano" a 0° a pesar de que su distancia al origen sea "cercana" (Figura 69).

En la Figura 69, según los criterios lógicos de lectura, se debe leer primero el bloque de información 1 y luego el bloque 2, sin importar las distancias de cada bloque al origen, con este ejemplo se puede determinar que la variable que predomina para imponer un Orden Lógico de Lectura es el ángulo que forman las componentes con el $eje\ x$. Y La variable dP servirá para

determinar el Orden de Lectura cuando los bloques de información formen ángulos similares con el $eje\ x$.

Figura 69

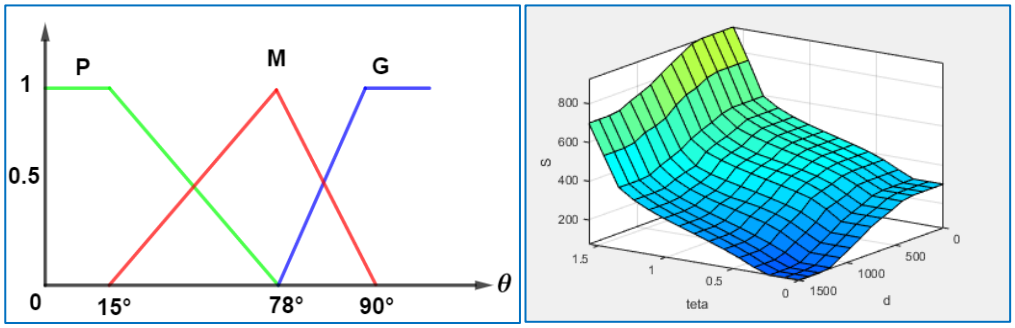
Ejemplo de preferencia de lectura.



Tomando en cuenta esta consideración se rediseñó las Funciones de Membresía de la variable θ (Figura 70 a)), la nueva Superficie de Control se muestra en la Figura 70 b).

Figura 70

a) Funciones de Membresía de la Variable "θ". b) Superficie de Control generada.



a) b)

Variable de salida "S" del sistema de Control Difuso

La variable de salida "S" servirá para otorgar una ponderación a cada cadena de texto, por esta razón se impuso un Universo de Discurso que va de 0 a 1000 unidades, se pudo haber impuesto cualquier otro rango, pero para el presente proyecto este rango obtuvo buenos resultados.

Para definir las Funciones de Membresía de "S" se debe tomar en cuenta que se establecieron 3 Funciones de Membresía tanto para "θ" y para "dP" lo que resulta en 9 posible combinaciones para las variables de entrada.

Las Variables Lingüísticas que se usaron son las siguientes:

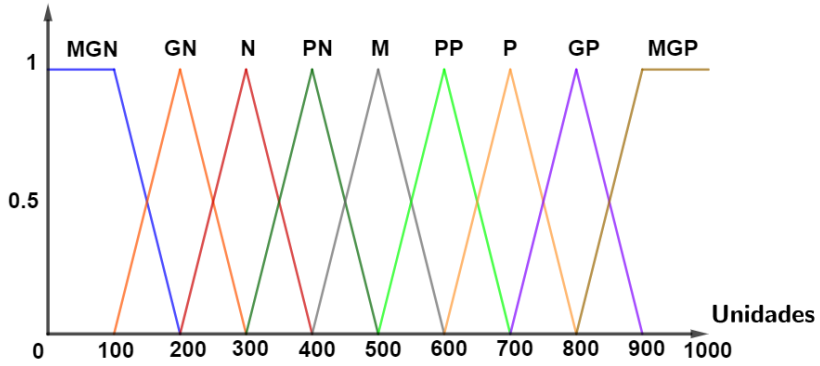
- Muy Grande Positivo *(MGP)*

- Grande Positivo *(GP)*
- Positivo *(P)*
- Pequeño Positivo *(PP)*
- Medio *(M)*
- Pequeño Negativo *(PN)*
- Negativo *(P)*
- Grande Negativo *(GN)*
- Muy Grande Negativo *(MGN)*

Las funciones de membresía para cada una de las expresiones lingüísticas anteriores se muestran en la Figura 71.

Figura 71

Funciones de Pertenencia para "S.



Como se puede ver, *"MGP"* Posee la mayor ponderación para la variable *"S"*.

Reglas de Control

El siguiente paso después de definir las variables de entrada y de salida, es expresar cómo se relacionan entre sí, para ello veamos el siguiente ejemplo:

Suponiendo que una entrada al sistema es Fuzzificada como:

- Primer caso: $\theta = G$ y $dP = G$

La variable de salida "S" tomará el valor lingüístico de *"Muy Grade Positivo (MGP)"* debido a que la fuzzificación de la entrada representa los valores máximos en su respectiva ponderación, por esta razón la salida también asumirá el valor máximo de ponderación, en este caso *"MGP".*

A esto se denominan Reglas de Control. La expresión lingüística de la regla mencionada es:

"Si θ es Grande y dP es Grande, la salida S será Muy Grande Positivo."

Se debe analizar los demás casos para asignar un valor lingüístico de la variable de salida *"S"*:

- Segundo caso: $\theta = G$ y $dP = M$
- Tercer caso: $\theta = G$ y $dP = P$
- Cuarto caso: $\theta = M$ y $dP = G$
- Quinto caso: $\theta = M$ y $dP = M$
- Sexto caso: $\theta = M$ y $dP = P$
- Séptimo caso: $\theta = P$ y $dP = G$

- Octavo caso: $\theta = P$ y $dP = M$
- Noveno caso: $\theta = P$ y $dP = P$

Como se mencionó anteriormente, la variable "θ" tiene mayor peso a la hora de definir las ponderaciones, por esta razón se asignaron los siguientes valores lingüísticos de *"S"*:

- Segundo caso: $\theta = G$ y $dP = M \rightarrow S = GP$
- Tercer caso: $\theta = G$ y $dP = P \rightarrow S = P$
- Cuarto caso: $\theta = M$ y $dP = G \rightarrow S = PP$
- Quinto caso: $\theta = M$ y $dP = M \rightarrow S = M$
- Sexto caso: $\theta = M$ y $dP = P \rightarrow S = PN$
- Séptimo caso: $\theta = P$ y $dP = G \rightarrow S = N$
- Octavo caso: $\theta = P$ y $dP = M \rightarrow S = GN$
- Noveno caso: $\theta = P$ y $dP = P \rightarrow S = MGN$

Este análisis se lo resume en la Tabla 6.

Tabla 6

Estados de la variable "S".

θ \ dP	**G**	**M**	**P**
G	MGP	GP	P
M	PP	M	PN
P	N	GN	MGN

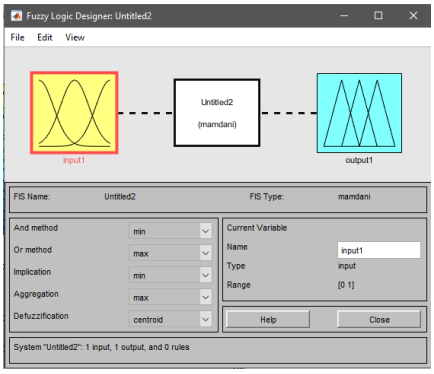
Implementación del Clasificador Difuso para la creación de la base de datos de Orden de Lectura de las imágenes

Con la ayuda de la herramienta *"Fuzzy Logic Designer"* de Matlab® se implementó todas las Funciones de Membresía y Reglas de Control para las variables de entrada y de salida del sistema.

El primer paso es abrir la Herramienta *"Fuzzy Logic Designer"* que se encuentra en las aplicaciones de Matlab® en el apartado de *"Diseño y Análisis de Sistemas de Control"*, se abrirá la ventana mostrada en la Figura 72.

Figura 72

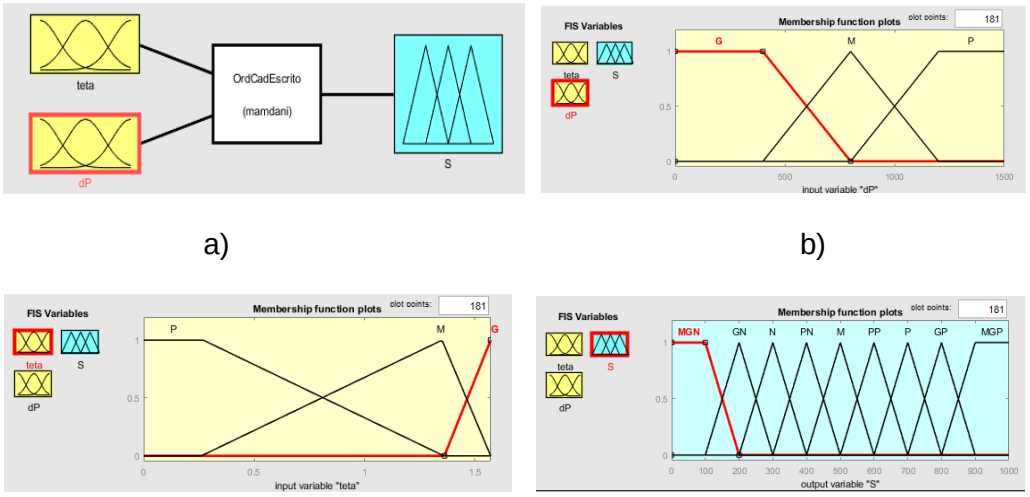
Interfaz inicial de la herramienta "Fuzzy Lógic Designer".



En esta ventana se deberán ingresar: el universo de discurso para cada una de las variables de entrada y de salida y las funciones de membresía con sus respectivas Variables Lingüísticas. Ver Figura 73.

Figura 73

a) Diagrama de entradas y salidas del Clasificador difuso. Funciones de Membresía de las variables b) dP, c) θ y d) S.



a) b)

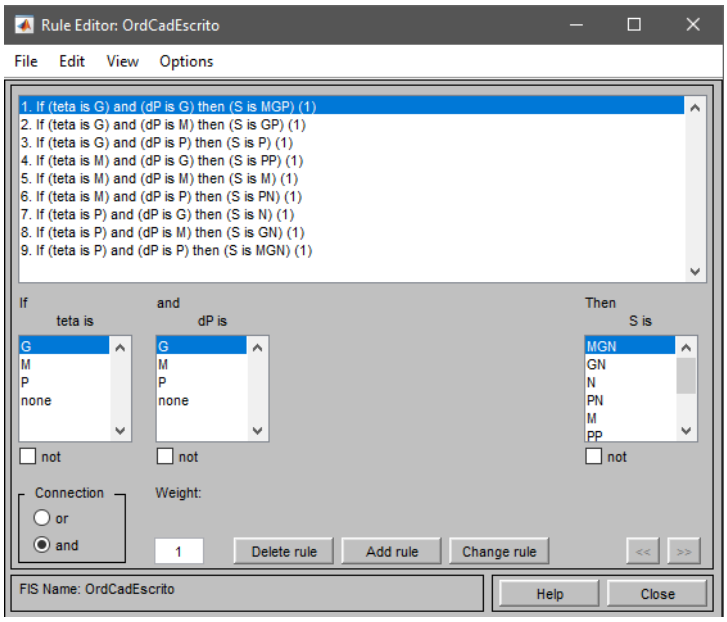
c) d)

Para el diseño del Clasificador Difuso, Matlab® permite realizarlo por medio de métodos: el método de Mandani o el método de Sugeno. En este proyecto se utilizó el método de Mandani debido a que se necesita una superficie de salida no continua y el método de Sugeno plantea superficies de salida continuas. (Sanahuja, 2017)

El siguiente paso es ingresar las Reglas de Control que definen el funcionamiento del sistema. Para esto, Matlab® ofrece la interfaz de la Figura 74 para definir las reglas.

Figura 74

Reglas de control que definen el sistema.

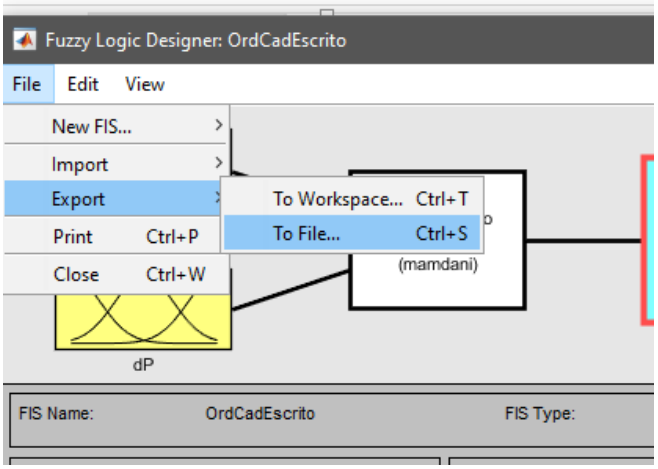


Una vez que se ingresó todos los datos se puede observar la superficie de control generada (Figura 70 b)) mediante el comando “CTRL + 6”.

Para comprobar el funcionamiento del sistema es necesario exportar el Clasificador Difuso mediante la función *“export”* en el menú *“File”* de la herramienta *“Fuzzy Logic Designer”* ver Figura 75.

Figura 75

Exportación del Clasificador Difuso a un formato con extensión "fis".



De esta forma se generó el archivo "*OrdCadEscrito.fis"* el cual se usó para la clasificación de los bloques de información.

Para cargar a Matlab® el Clasificador Difuso es necesario usar la función *"*`readfis`*"* y para evaluar las entradas se usó el comando *"*`evalfis`*"*.

Para la evaluación de las entradas es necesario explicar cómo se presentan dichas entradas; el algoritmo desarrollado entrega el ángulo y distancia desde el origen hasta el punto final de los bloques de información como se muestra en la Figura 63. Estos datos se presentan de la siguiente forma:

Tabla 7

Datos de entrada del Clasificador Difuso.

Pares de entrada del Clasificador Difuso	
$\boldsymbol{\theta}$	dP
$\boldsymbol{\theta_1}$	dP_1
$\boldsymbol{\theta_2}$	dP_2
$\boldsymbol{\theta_3}$	dP_3
.	.
.	.
.	.
$\boldsymbol{\theta_n}$	dP_n

Donde θ contiene el valor de todos los ángulos de los bloques de información y dP contiene las distancias de las coordenadas finales de cada bloque, el Clasificador Difuso toma en pares los valores de la Tabla 7 y realiza el análisis respectivo para generar la salida S para cada uno de los pares de $(\theta; dP)$. Ver Tabla 8.

Tabla 8

Asignación de salida S a las variables de entrada θ y dP.

Asignación de S para las entradas θ y dP			
$\boldsymbol{\theta}$	dP	➔	S
$\boldsymbol{\theta_1}$	dP_1	➔	S_1
$\boldsymbol{\theta_2}$	dP_2	➔	S_2
.	.	.	.
.	.	.	.
$\boldsymbol{\theta_n}$	dP_n	➔	dP_n

Los valores de S representan la jerarquía de orden de lectura de cada bloque de información, los bloques cuya salida S sea mayor serán los que se lean primero. Para esta clasificación se utilizó el comando "`sort`" de Matlab® que permite la clasificación de mayor a menor de la variable S.

Cabe recordar que cada bloque de información se compone por los *BBox* de las líneas de texto del documento.

Luego de ordenar los valores y de definir el orden de lectura de los bloques de información se concatena la información en un solo vector únicamente con el *id* de los *BBox* de las líneas de texto concatenando también el id de los elementos divisores de página, encabezados o pie de página en caso de ser necesario, el resultado es el Orden de Lectura.

Este procedimiento se realizó para las 530 imágenes de la base de datos y se generó una nueva base de datos de archivos con extensión *"mat"* con el siguiente formato:

"Nombre.mat"

Los cuales contienen los resultados de orden de lectura de cada documento para su posterior comparativa y análisis.

A continuación se muestra un ejemplo del orden de lectura extraído de una imagen.

Figura 76

Imagen Original.

and workshops) are detailed. Any interested engineer or scientist is welcome to join as an ASI member, associate member, or volunteer by following the procedures described on the ASI website and to contribute to ASI activities.

Proposals on ASI-related activities (e.g., courses, special sessions/issues, workshops, or competitions) for synergies within SPS, IEEE Societies, and other scientific communities and relevant industrial bodies are welcome from anyone who can send an email to ASI Chair Prof. Ioannis Pitas and/or ASI Vice Chair Prof. Carlo Regazzoni.

Authors

Carlo Regazzoni (carlo.regazzoni@unige.it) received his M.S. and Ph.D. degrees from the University of Genova, Italy, in 1987 and 1992, respectively. Currently, he is a professor at the Polytechnic School of the University of Genova, Italy. He has been involved in research on signal and video processing and data fusion since 1988. His current research interests are cognitive dynamic systems, adaptive and self-aware data fusion, machine learning, software, and cognitive radio. He is one of the pioneering researchers in intelligent video surveillance. In 1998, he created the IEEE International Conference on Advanced Video and Signal-Based Surveillance. He is author or coauthor of more than 100 journal papers and 400 conference papers and book chapters (his Google Scholar h-index is 40). He has served within the IEEE Signal Processing Society in various positions including vice president-Conferences (2015–2017). He is currently vice chair of the IEEE Autonomous Systems Initiative. He is a Senior Member of the IEEE.

Ioannis Pitas (pitas@csd.auth.gr) received his diploma and Ph.D. degree in electrical engineering, both from the Aristotle University of Thessaloniki, Greece, where, since 1994, he has been a professor in the Department of Informatics. He has published more than 1,090 papers, contributed to 50 books in his areas of interest, and edited or (co)authored another 11 books. He has also been member of the program committee of many scientific conferences and workshops, an associate editor or coeditor of nine international journals, and the general/technical chair of four international conferences. He participated in 69 R&D projects, primarily funded by the European Union. He has more than 29,000 citations to his work, and his h-index is 79+ (Google Scholar). He leads the big European H2020 R&D project MULTIDRONE and is the chair of the IEEE Autonomous Systems Initiative. He is a Fellow of the IEEE and EURASIP as well as an IEEE Distinguished Lecturer.

SP

SP FORUM *(continued from page 137)*

of Technology, Limassol, and his M.S. and Ph.D. degrees in electrical and computer engineering from West Virginia University, Morgantown. He is a professor in the School of Electrical, Computer, and Energy Engineering at Arizona State University, Tempe, where he also directs the Sensor Signal and Information Processing (SenSIP) center and founded the SenSIP industry consortium. He has served as an associate editor of *IEEE Transactions on Signal Processing* and as general cochair of ICASSP 99. He is a corecipient of the 2002 IEEE Donald G. Fink Paper Prize Award. His research interests are in the areas of adaptive signal processing, speech processing, machine learning, and sensor systems. He is a Fellow of the IEEE.

Mónica F. Bugallo (monica.bugallo@stonybrook.edu) received her B.S., M.S., and Ph.D. degrees in computer science and engineering from the University of A Coruña, Spain. She is a full professor of electrical and computer engineering and the faculty director of the Women in Science and Engineering program at Stony Brook University, New York. Her research interests are in the field of statistical signal processing, with emphasis on the theory of Monte Carlo methods and their application to different disciplines including biomedicine, sensor networks, and finance. She has authored and coauthored two book chapters and more than 150 journal papers and refereed conference articles. She is a Senior Member of the IEEE.

References

[1] W. Bajwa, "On 'flipping' a large signal processing class [SP Education]," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 158–170, 2017.

[2] A. V. Oppenheim, "One plus one could equal three (and other favorite cliches)," *IEEE Signal Process. Mag.*, vol. 23, no. 6, pp. 10–12, 2006.

[3] S. K. Mitra, "Reminiscences of a department chair," *IEEE Signal Process. Mag.*, vol. 23, no. 3, pp. 10–13, 2006.

[4] Responsible Metrics, "The metric tide." [Online]. Available: https://responsiblemetrics.org/the-metric-tide/

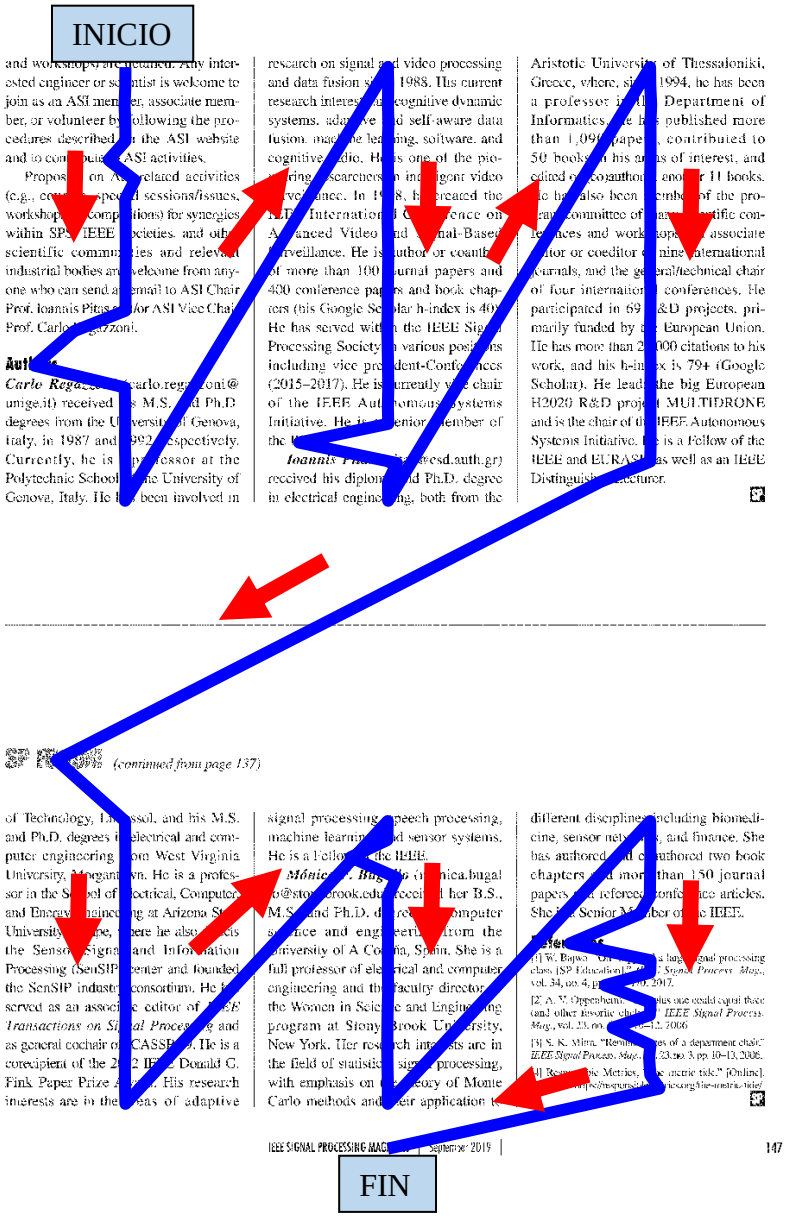
SP

Figura 77

BBox de las líneas de texto de la imagen original.

Figura 78

Orden de lectura de la imagen.



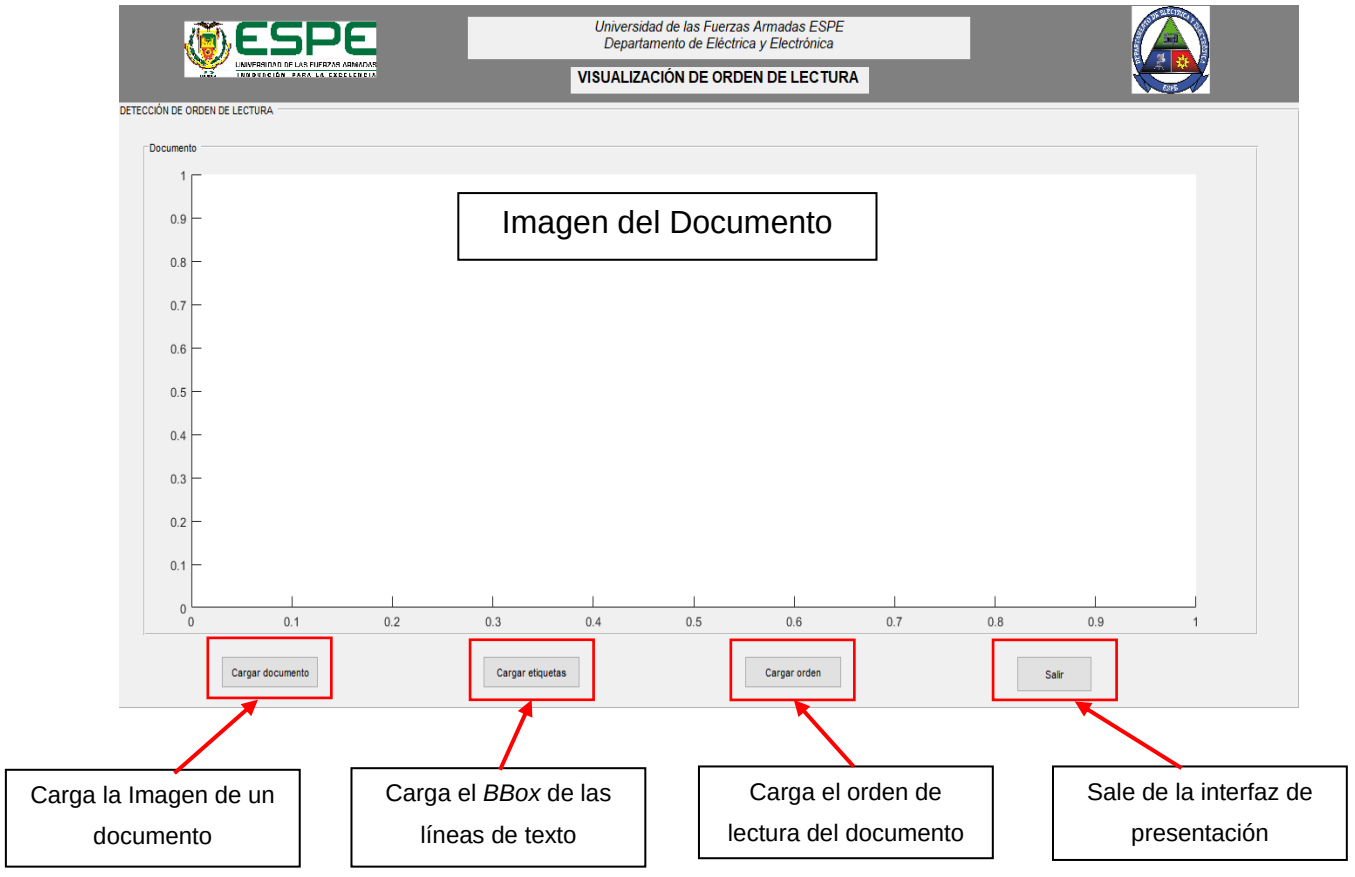
La línea azul de la Figura 78 muestra la Orden de Lectura generado por el algoritmo.

Interfaz Gráfica para presentación de resultados

Una vez que se creó la base de datos con el Orden de Lectura de cada uno de los archivos es necesario implementar una interfaz gráfica que permita visualizar el resultado obtenido por el algoritmo desarrollado.

Figura 79

Descripción de la interfaz para la visualización del Orden de Lectura en Matlab®.



CAPITULO 4

Pruebas y Análisis de Resultados

Interfaz gráfica para creación de base de datos para evaluación de resultados

El objetivo de esta interfaz es crear una base de datos que permita comparar y evaluar el desempeño del algoritmo desarrollado.

Se implementó una interfaz gráfica en el GUIDE[5] de Matlab® en la que se visualiza la imagen con los respectivos *BBox* de las líneas de texto, esta interfaz permite al usuario ingresar de forma manual el Orden de Lectura de la imagen y almacenarlo en un archivo en formato "*mat"* para su posterior análisis en Matlab®. (MathWorks, 2021)

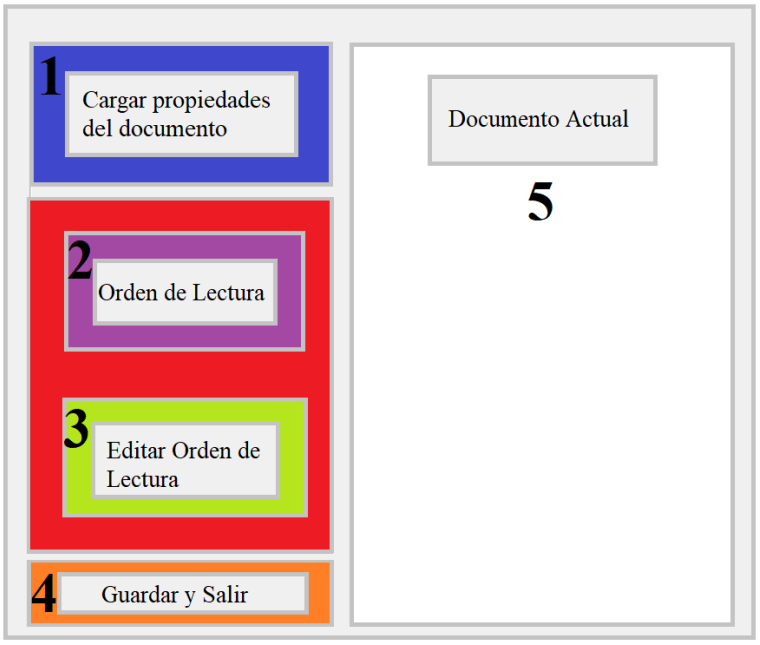
En la Figura 80 se puede ver la distribución de la interfaz con 5 secciones. La sección uno permite que se cargue la imagen *PNG* de un documento, sus *BBox* y el Orden de Lectura (en caso de que existiera uno). En la segunda sección habilita la opción de ingresar el orden de lectura y se visualiza el Orden de Lectura mediante el *id* de cada uno de los *BBox* de las líneas de información. La tercera sección permite la modificación del Orden de Lectura que se está ingresado, esto se lo realizó para modificar este orden en caso de errores al momento de seleccionar los *BBox* de la imagen. La cuarta sección posee dos botones, el botón "Guardar" que permite almacenar el Orden de Lectura en un archivo con extensión *"mat"* y el botón "Salir" que permite

[5] https://la.mathworks.com/discovery/matlab-gui.html

cerrar la interfaz gráfica. La quinta sección es en la que se muestra la imagen del documento con sus *BBox* y permite seleccionarlos en bloque para generar un orden lógico de lectura.

Figura 80

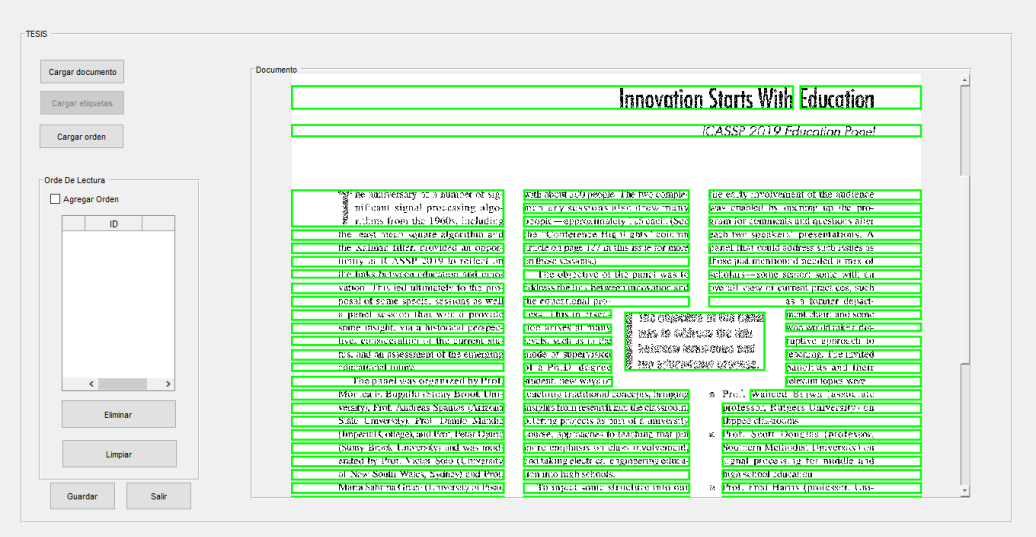
Diagrama de la interfaz gráfica para creación de base de datos.



La interfaz fue realizada mediante el entorno gráfico de GUIDE de Matlab®, esta herramienta permite el diseño y organización de la interfaz, ver Figura 81.

Figura 81

Interfaz gráfica para la creación de la base de datos.

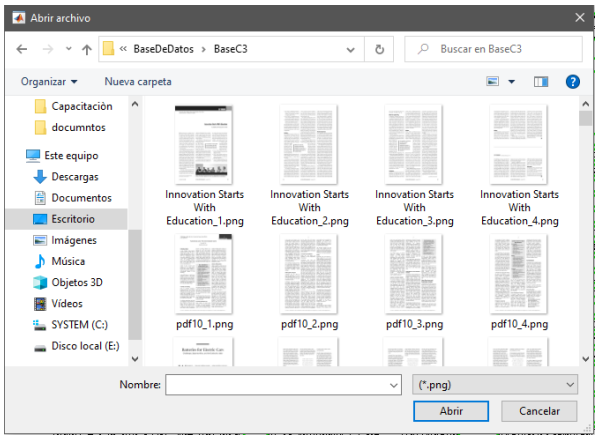


El uso de esta interfaz se detalla a continuación.

Para cargar una imagen se da click en “Cargar Documento” y se abrirá la pestaña que se muestra en la Figura 82.

Figura 82

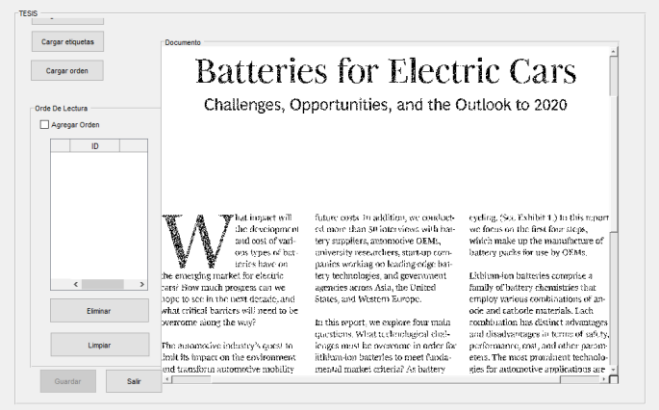
Cargar imagen a la interfaz gráfica.



Se debe seleccionar una imagen PNG dentro de la base de datos digitalizada y se oprime el botón “Abrir”, de esta forma se cargará la imagen.

Figura 83

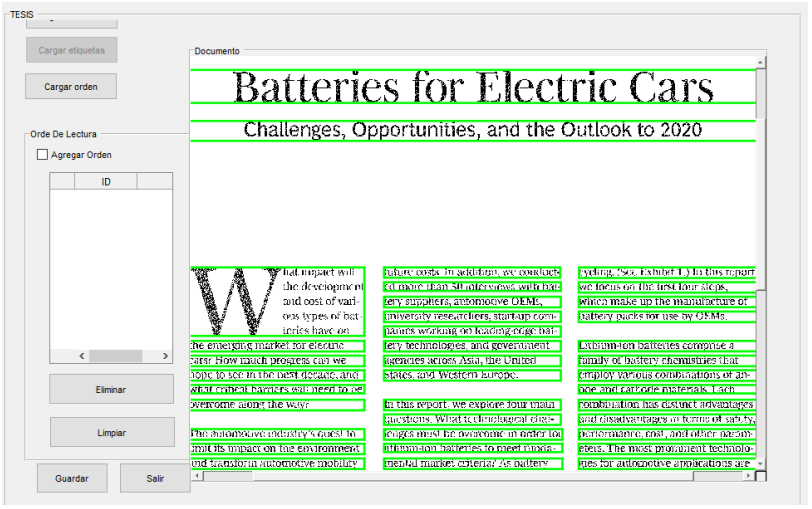
Visualización de la imagen en la interfaz gráfica.



Posterior a esto se deben cargar los *BBox* para esta imagen, para realizar esta acción se debe oprimir el botón “Cargar etiquetas”. El programa identifica la ubicación de los *BBox* y los carga sobre la imagen que se visualiza en la interfaz (Figura 84).

Figura 84

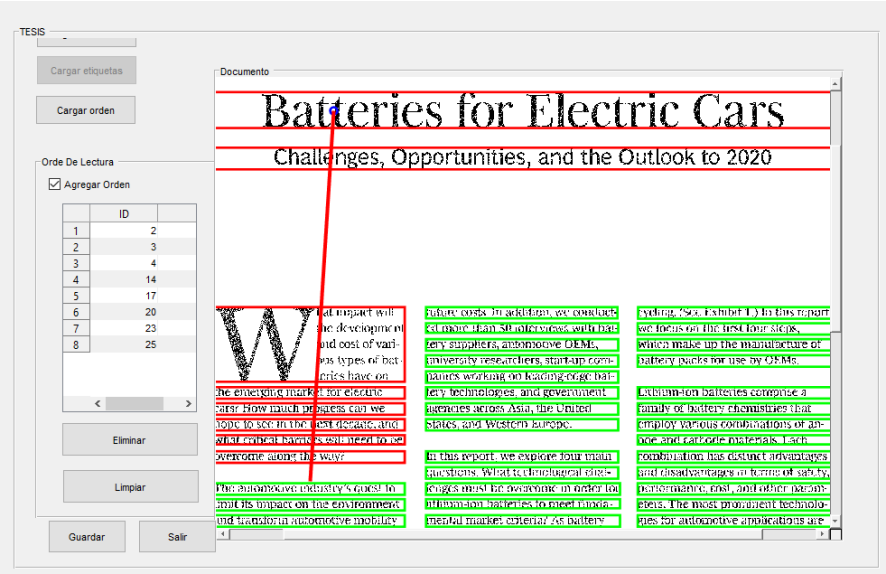
Carga de BBox de la imagen



Luego de esto se debe seleccionar el orden de lectura de la imagen, para esto se debe marcar la casilla “Agregar Orden”. La forma de seleccionar los *BBox* es mediante la tecla “d”; una pulsación y se selecciona el punto de inicio y otra pulsación para seleccionar los elementos que van a ser leídos como un solo bloque de información. Por defecto, el programa ordena a los *BBox* de arriba hacia abajo.

Figura 85

Agregar orden de lectura.



Como se observa en la Figura 85, el orden de lectura generado se muestra en el panel en la parte izquierda de la interfaz según los *BBox* que se señalen. Adicionalmente, se agregaron dos botones, el primero es “Eliminar” el cual permite que se elimine de uno en uno los elementos que se agregaron al orden de lectura. Y el segundo botón, “Limpiar” que limpia totalmente el Orden de Lectura.

Además, se agregaron dos botones que permiten guardar en un archivo en formato “mat” el Orden de Lectura generado o salir del programa.

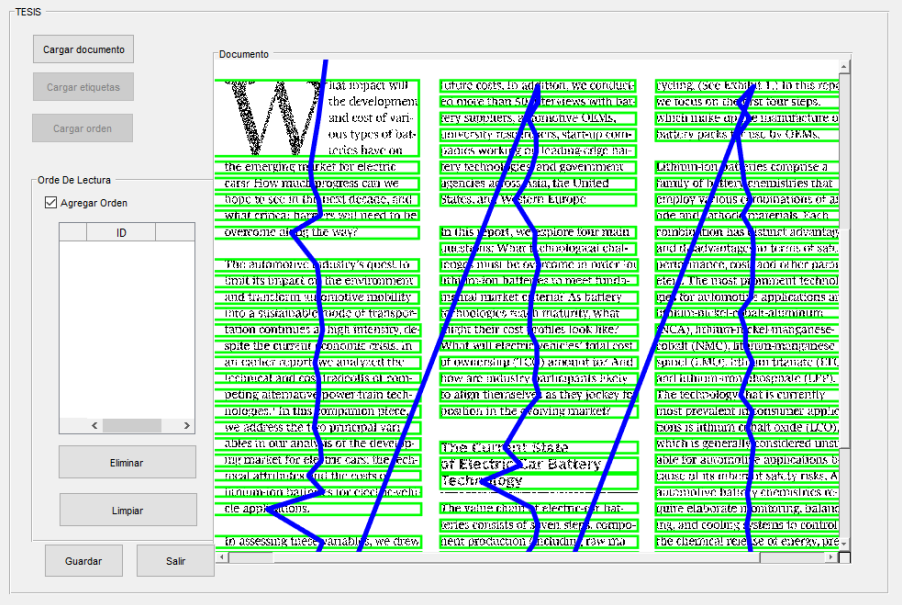
El formato *“mat”* es un tipo de archivo manejado por Matlab® para almacenar valores de variables fuera del entorno del programa, de esta forma se puede exportar la información. En el presente proyecto se generó archivos *“mat”*

con el orden de lectura de cada una de las imágenes que conforma la base de datos. (MathWorks, 2021)

La interfaz permite revisar el Orden de Lectura generado, para esto se debe pulsar el botón “Cargar orden” posteriormente de haber cargado la imagen y cargados los *BBox* de la imagen.

Figura 86

Ejemplo de orden de lectura generado mediante la interfaz gráfica.



Se solicitó a un grupo de 30 estudiantes a los cuales se les repartió las 530 imágenes para la extracción manual del orden de lectura. Con esto se generó la base de datos para la comparación entre el orden generado por los estudiantes y el Orden de Lectura generado por el algoritmo de este proyecto.

Métricas para la evaluación del desempeño del algoritmo implementado

Para comparar el desempeño del algoritmo desarrollado con otros métodos de detección de Orden de Lectura es necesario tener acceso a las bases de datos con la que se experimentó y comparar el resultado de estos códigos con el resultado obtenido por el algoritmo del presente documento. Acceder a esta base de datos resulta una tarea muy complicada, por lo cual se descartó esta posibilidad.

Como se dijo anteriormente, es necesario tener una base de datos para la comparación y evaluación del presente algoritmo, esta base de datos se creó manualmente con la ayuda de 30 estudiantes de la Universidad De Las Fuerzas Armadas – ESPE.

Se plantearon dos métricas para la evaluación de los resultados basadas en las coincidencias del Orden de Lectura generado manualmente y el Orden de Lectura generado por el algoritmo creado en este proyecto:

- Porcentaje de aciertos mediante la comparación estricta del Orden de Lectura.
- Porcentaje de aciertos mediante la comparación de elementos conectados en el Orden de Lectura.

Porcentaje de aciertos mediante la comparación estricta del Orden de Lectura

El Orden de Lectura generado por el algoritmo y el Orden de Lectura generado por los estudiantes de la Universidad de las Fuerzas Armadas – ESPE poseen el mismo formato, y el mismo número de elementos para cada una de las imágenes.

Para un mejor entendimiento de las métricas consideremos cambiar de nombre a las base de datos. La base de datos de Orden de Lectura generada por el algoritmo se denominará “Base de datos de prueba” y la Base de datos de Orden de Lectura generada por los estudiantes será la “Base de datos de referencia”.

Tabla 9

Renombramiento de base de datos.

Renombramiento de nombre de las bases de datos	
Base de datos de Orden de Lectura generada por el algoritmo	Base de datos de prueba
Base de datos de Orden de Lectura generada por los estudiantes	Base de datos de referencia

Asumiendo que la base de referencia contiene información acertada del orden de lectura de las imágenes, la primera métrica será comparar si cada

elemento de la Base de datos de prueba coincide con cada elemento de la Base de referencia

Para explicarlo propongamos un ejemplo:

- “Dato 1” un archivo de la Base de datos de prueba.
- “Dato 2” un archivo de la Base de datos de referencia.
- Tanto Dato 1 como Dato 2 corresponden al orden de lectura de la misma imagen.

Donde, El contenido de Dato 1 es:

Tabla 10

Orden de lectura contenido en Dato 1 para el ejemplo.

Contenido en Dato 1				
1	2	3	4	6

Y el contenido de Dato 2 es:

Tabla 11

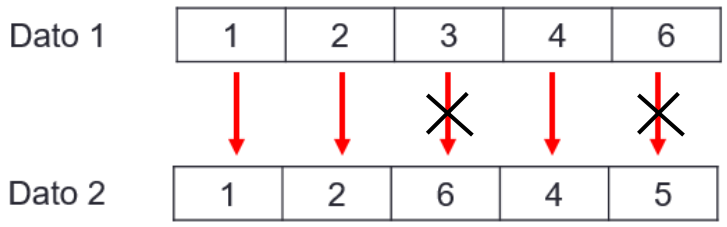
Orden de lectura contenido en Dato 2 para el ejemplo.

Contenido en Dato 2				
1	2	6	4	5

Si realizamos una comparación dato a dato tenemos:

Figura 87

Comparación entre Dato 1 y Dato2.



El objetivo de la presente métrica es cuantificar el número de coincidencias entre el Orden de Lectura de Dato 1 y Dato 2. Siendo:

- $n = número\ de\ elementos\ del\ archivo\ que\ contiene\ el\ Orden\ de\ Lectura$
- $j = el\ número\ de\ aciertos\ entre\ los\ dos\ elementos$

Se obtiene el porcentaje de acierto para cada elemento de la siguiente manera:

$$\%acierto_{m1} = \frac{j}{n} * 100\%$$

En la Figura 87 se observa que de 5 elementos que poseen los datos, 4 de ellos coinciden, es decir

- $n = 5$
- $j = 3$

Aplicando la fórmula se tiene:

$$\%acierto_{m1} = \frac{3}{5} * 100\% = 60\%$$

Este procedimiento se lo realizó para cada elemento, tanto de la Base de datos de prueba y la Base de datos de referencia correspondientes a una misma imagen y se promedió el resultado para documentos de una, dos y tres columnas.

Porcentaje de aciertos mediante la comparación de secuencias de 2 elementos del Orden de Lectura

Utilizando las mismas bases de datos y teniendo las mismas consideraciones que la métrica anterior, esta métrica pretende cuantificar el número de aciertos en la progresión del Orden de Lectura. Para una mejor comprensión tomemos en cuenta el siguiente ejemplo:

Considerando que:

- “Dato 3” correspondiente a la Base de datos de Prueba

- “Dato 4” correspondiente a la Base de datos de Referencia
- Tanto Dato 3 como Dato 4 corresponden al orden de lectura de la misma imagen.

Donde, El contenido de Dato 3 es:

Tabla 12

Orden de lectura contenido en Dato 3 para el ejemplo.

Contenido en Dato 3				
1	6	7	2	3

Y el contenido de Dato 4 es:

Tabla 13

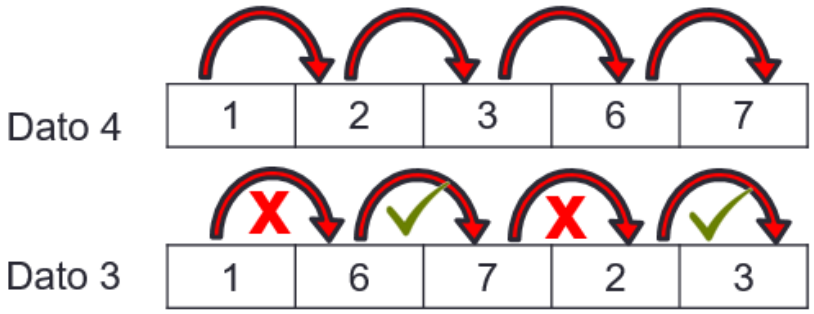
Orden de lectura contenido en Dato 4 para el ejemplo.

Contenido en Dato 4				
1	2	3	6	7

Si realizamos una comparación de la progresión de los datos tenemos:

Figura 88

Comparación entre Dato 3 y Dato 4.



Para comprender la métrica tomemos el valor de la segunda posición de Dato 3, el valor de la etiqueta es 6 y la etiqueta que se lee después es 7, si esta secuencia de dos valores se repite en Dato 4 se considera un acierto sin importar a ubicación de esta secuencia, caso contrario sería un error.

Como se observa en la Figura 88 después de la etiqueta 7 se debe leer la etiqueta 2 según el Orden de Lectura de Dato 3 pero esta secuencia no se repite en Dato 4 por lo que se considera un error.

En este caso:

- $r = número\ de\ posibles\ secuencias\ de\ dos\ valores$
- $k = el\ número\ de\ aciertos\ de\ las\ secuencias$

Se obtiene el porcentaje de acierto para cada elemento de la siguiente manera:

$$\%acierto_{m2} = \frac{k}{r} * 100\%$$

En la Figura 83 se observa que de 4 posibles secuencias, únicamente coinciden 2

- $n = 4$
- $k = 2$

Aplicando la fórmula se tiene:

$$\%acierto_{m2} = \frac{2}{4} * 100\% = 50\%$$

Este procedimiento se lo realizó para cada elemento, tanto de la Base de datos de prueba y la Base de datos de referencia correspondientes a una misma imagen y se promedió el resultado para documentos de una, dos y tres columnas.

Análisis de resultados

En la presente sección se detallan los resultados que se obtuvieron al aplicar las métricas establecidas.

Porcentaje de aciertos mediante la comparación estricta del Orden de Lectura

En la Tabla 14 se observa los resultados de aplicar la comparación estricta del Orden de Lectura a los 3 grupos de documentos.

Tabla 14

Resultados de aplicar la comparación de orden estricto.

Número de columnas	Porcentaje de acierto
Una columna	99.1%
Dos columnas	78.7%
Tres columnas	92.25%
Promedio	**90.3%**

Analizando los resultados generales, se observa que el algoritmo desarrollado funciona correctamente al tener más del 90% de aciertos en el Orden de Lectura de los documentos de la base de datos. Pero se puede apreciar que tiene deficiencias en documentos de dos columnas. Estos errores se deben en gran medida a la forma de extracción de los *BBox* de las líneas de texto cuando el documento contiene expresiones matemáticas.

A continuación se muestra el *BBox* de las líneas de texto de un documento que únicamente está compuesto por caracteres no matemáticos (Figura 89 a)) y la extracción de su orden de lectura (Figura 89 b)).

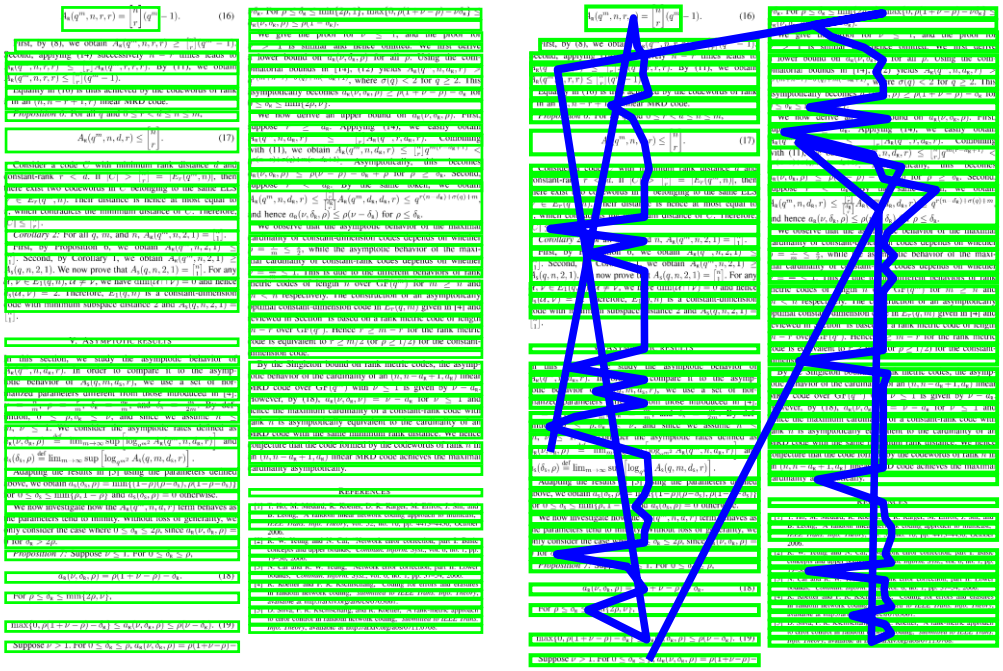
Figura 89

Ejemplo de documento sin fórmulas matemáticas. a) BBox del Documento. b) Orden de Lectura.

En la Figura 90 literal a se muestra el *BBox* de las líneas de texto de un documento que contiene caracteres matemáticos, y su orden de lectura (Figura 90 literal b).

Figura 90

Ejemplo de documento con fórmulas matemáticas. a) BBox del documento b) Orden de lectura generado.

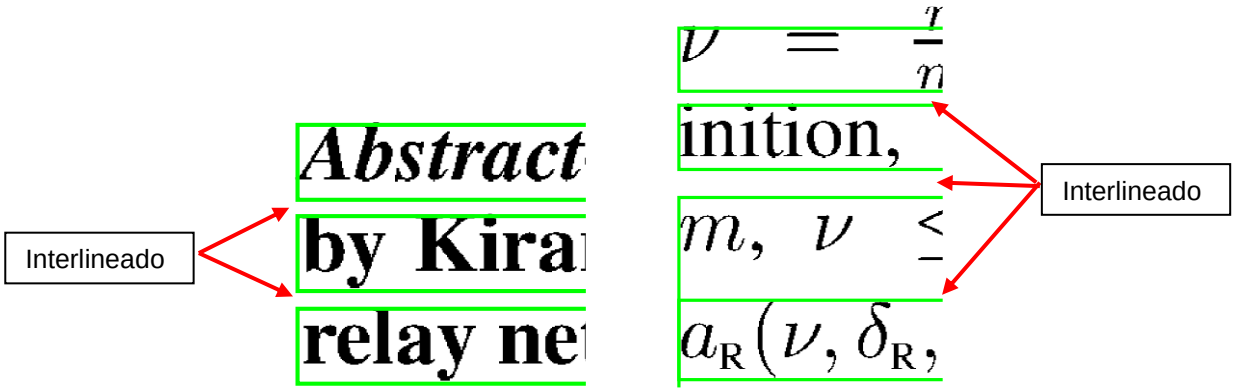


a) b)

La principal diferencia entre los documentos de la Figura 89 y la Figura 90 es el espacio de interlineado, en la Figura 91 literal a, se muestra el interlineado del documento cuya información no contiene caracteres matemáticos, el interlineado se mantiene casi constante para cada línea de texto, lo que no sucede en la Figura 91 literal b donde se muestra que el interlineado del documento con fórmulas matemáticas es variante y los *BBox* de las líneas de texto se sobreponen, esta característica hace que el algoritmo de detección de lectura obtenga menos rendimiento ante documentos que contienen expresiones matemáticas.

Figura 91

Espacio interlineado. a) Documento sin caracteres matemáticos b) Documento con caracteres matemáticos.



a) b)

La solución a este problema es implementar un sistema de segmentación de información más robusto lo cual se deja planteado para trabajos futuros, ya que un análisis más profundo tomaría más tiempo y esfuerzo por la cantidad de variantes que se puede encontrar en los documentos como tipo de letra, tamaño de letra, interlineado, expresiones matemáticas, justificación del documento, etc. Diseñar un sistema más robusto llevaría mucho más tiempo y se quitaría importancia al principal objetivo del presente proyecto que es determinar el Orden de Lectura de documentos.

Porcentaje de aciertos mediante la comparación de secuencia de 2 elementos del Orden de Lectura

Un problema de la anterior métrica es que solo basta un elemento al inicio del Orden de Lectura para que el porcentaje de acierto sea cero, esto no

es del todo correcto debido a las imágenes o cualquier otro elemento que pueden existir en el diseño del documento. En el caso particular de las imágenes, estas no tienen definido su Orden de Lectura por lo que pueden alterar el resultado si se compara estrictamente el Orden de Lectura.

Esta herramienta de evaluación permite medir el grado de acierto comparando la secuencia entre pares de elementos, esto ya se lo explicó en la sección anterior.

Los resultados de esta métrica son los mostrados en la Tabla 15.

Tabla 15

Resultado al aplicar la comparación entre pares de elementos.

Número de columnas	**Porcentaje de acierto**
Una columna	99.6%
Dos columnas	92.9%
Tres columnas	97%
Promedio	**95%**

Como era previsto, el porcentaje de acierto con esta métrica aumento considerablemente, pero el porcentaje de aciertos para los archivos de dos columnas sigue siendo inferior a los documentos de 1 y 3 columnas.

CAPITULO 5

Conclusiones y Recomendaciones

Conclusiones

- Luego de realizar el estudio del estado del arte se evidenció que el tema de segmentación en bloques de información no es un tema nuevo de estudio, en la bibliografía se pueden encontrar varios métodos basados en distintos principios para la extracción y clasificación de la información contenida en un documento. En el presente proyecto se trabajó con un algoritmo de segmentación basado en la conversión LRSA debido a su fácil implementación y grandes resultados. Sin embargo, al realizar el la segmentación se vio la necesidad de analizar los Componentes Conectados para ejecutar un suavizado horizontal de los caracteres del documento y así cubrir las necesidades que demanda el proyecto.

- En la bibliografía se encontraron varios estudios dedicados a la extracción del Orden de Lectura de un documento, sin embargo, no se encontró ningún estudio que ataque al problema mediante la Lógica Difusa. Un área en común que tienen todos estos estudios es que se basan directa o indirectamente en proposiciones gramaticales para establecer las reglas que definen el sistema de detección de Orden de Lectura, pero, muchas de estas reglas no son claras o no están correctamente delimitadas para la implementación en un sistema de control que permita identificar el Orden de Lectura. El principio de la Lógica Difusa es trabajar precisamente con

estas proposiciones por lo que en el método planteado se desarrolló un algoritmo de Detección de Orden de Lectura basado en reglas lingüísticas que fueron utilizadas tanto para la creación de Reglas de Lógica Difusa como para la segmentación y extracción de información parcial del documento.

- Se tuvo gran dificultad para encontrar un método de evaluación y comparación de resultados con otros estudios de detección de Orden de Lectura. Para comparar el presente proyecto con otros trabajos realizados es necesario trabajar con la misma base de datos, sin esta base resulta imposible generar una comparativa confiable que permita evaluar el desempeño del algoritmo implementado, por esta razón se creó una propia base de datos basado en orden de lectura determinado manualmente por 30 Estudiantes de la Universidad las Fuerzas Armada – ESPE para implementar métricas que permitan la evaluación del desempeño del presente proyecto, esta extracción se lo realizó mediante una interfaz gráfica desarrollada en Matlab®.

- El principal problema que se encontró al momento de digitalizar las imágenes y extraer el *BBox* de las líneas de texto fue las expresiones matemáticas de algunos los documentos, las expresiones matemáticas generan variaciones en la línea media de texto lo que genera que el *BBox* de líneas de texto vecinas se sobrepongan generando errores al momento de determinar el orden de lectura. Otro problema que generan las

expresiones matemáticas es la gran separación entre los caracteres de la fórmula lo que genera caracteres aislados los cuales no fueron tomados en cuenta para la detección de Orden de Lectura.

- En el presente proyecto se manejó páginas de documentos con formatos de una, dos y tres columnas siendo los documentos de dos columnas los que generaron menos aciertos con un 92.94% de coincidencias al evaluar las secuencias entre dos elementos y apenas un 78.7% de aciertos cuando se evaluó el orden estricto de lectura. Este bajo desempeño se debe a que muchos de las páginas de los documentos a dos columnas poseen expresiones matemáticas, esto generó errores al momento de la segmentación de la información ocasionando que la efectividad del algoritmo de detección de Orden de Lectura se vea reducido.

- Los mejores resultados que se obtuvo fue con los documentos de una columna, en los cuales se obtuvo un 99.1% al comparar el orden estricto de lectura y un 99.6% de aciertos cuando se comparó secuencias de dos elementos, esto se debe en gran medida a que los documentos de una columna poseen un interlineado más grande que los documentos de dos columnas, haciendo que la segmentación sea más eficiente y mejore la efectividad de la detección de Orden de Lectura.

- Se obtuvo un gran rendimiento con las páginas de documentos con tres columnas en su estructura al obtener un 92.25% de aciertos cuando se evaluó con el orden estricto de lectura y un 92,9% de coincidencias

cuando se comparó las secuencias de dos elementos. En estos documentos no se encontraron gran número de expresiones matemáticas lo que facilitó la segmentación y, de igual forma con los documentos de una columna, mejoró el desempeño de la detección del Orden de Lectura.

- En la presente investigación se creó una base de datos que alberga 530 hojas de documentos PDF dentro de las cuales se encuentra 200 páginas con diseño a una sola columna, 150 páginas de dos columnas y 180 páginas de 3 columnas. Esta base de datos contiene la información del *BBox* de las líneas de texto, el Orden de Lectura generado por Estudiantes de la Universidad de las Fuerzas Armadas – ESPE y el Orden de Lectura generado por el algoritmo desarrollado.

- Se publicará de manera gratuita la base de datos desarrollada en el presente proyecto para su acceso con fines educativos e investigativos de terceras personas.

Recomendaciones y Trabajos futuros

- Al momento de implementar métricas de evaluación se debe utilizar, en lo posible, las mismas bases de datos utilizadas por otros estudios similares, en el caso de no tener acceso a estas base de datos, la mejor opción es crear una propia y analizar los resultados únicamente de sus elementos y no comparar los resultados generales con otros proyectos debido a que los

resultados pueden variar dependiendo de los elementos de la base de datos que se esté utilizando.

- El algoritmo de segmentación, si bien, se adapta a las necesidades del proyecto, se recomienda para trabajos posteriores implementar un algoritmo de segmentación más robusto, esto mejorará el desempeño de algoritmo de detección de Orden de Lectura generado en este proyecto. En especial se recomienda mejorar la segmentación para páginas que contengan lenguaje matemático.

- Como trabajo futuro se recomienda ampliar la base de datos agregando páginas de archivos PDF con distintos diseños en su estructura para evaluar el desempeño del algoritmo y, de ser necesario, complementar el algoritmo para mejorar su desempeño.

- A futuro se puede utilizar este trabajo para crear aplicaciones para personas con deficiencias visuales que necesiten acceso a la información de cualquier documento en formato PDF.

Referencias

ADOBE ACROBAT. (2021). *Referencia de PDF*. Obtenido de https://acrobat.adobe.com/es/es/acrobat/about-adobe-pdf.html

Artifex. (s.f.). *Ghostscript*. Obtenido de Ghostscript: https://www.ghostscript.com/index.html

Asamblea Constituyente. (Octubre de 2008). Constitución de la República del Ecuador. *Registro Oficial No.449*.

Asamblea Nacional del Ecuador. (25 de Septiembre de 2012). Ley Orgánica de Discapacidades.

Baética. (15 de Mayo de 2019). *¡Formatos gráficos para imágenes! Ventajas y desventajas*. Obtenido de https://baetica.com/formatos-graficos-imagenes/

Carlos, M. (s.f.). En *Lógica Difusa una introducción práctica* (págs. 5-10).

Cattoni, R., Coianiz, T., Messelodi, S., & Modena, C. M. (1998). Geometric layout analysis techniques for document image understanding: a review. Trento, Italia: IRST.

Corbelli, A., Baraldi, L., Grana, C., & Cucchiara, R. (12 de 2016). Historical Document Digitization through Layout Analysis and Deep Content Classification. *International Conference on Pattern Recognition (ICPR)*, 4077–4082.

Corina, F., & Aldo, C. (8 de Marzo de 2016). *Principios de lógica difusa.*

Duarte, O. G. (Abril de 1999). Sistemas de lógica difusa. Fundamentos. *Revista Ingeniería e Investigación*(42), 22-30.

Esposito, F., Malerba, D., & Semeraro, G. (1994). MULTISTRATEGY LEARNING FOR DOCUMENT RECOGNITION. *Applied Arti¯cial Intelligence,, 8*(1), 33-84.

Ferilli, S., Basile, T. M., & Esposito, F. (2010). A histogram-based Technique for Automatic Threshold Assessment in a Run Length Smoothing-based Algorithm. *International Workshop on Document Analysis Systems*, 349–356.

Gordillo, R. (19 de Septiembre de 2019). *Control Inteligente.*

HASNAT, M. A. (Martes de Junio de 2007). *Run Length Smoothing Algorithm (RLSA)*. Obtenido de http://crblpocr.blogspot.com/2007/06/run-length-smoothing-algorithm-rlsa.html

Jean-Luc, M. (2005). Optimized XY-Cut for Determining a Page Reading Order. Lyon, Francia: IEEE Xplore. doi:10.1109/ICDAR.2005.182

Kacprzyk, J. (2008). Machine learning for reading order detection in document. En *Machine Learning in Document Analysis and Recognition* (págs. 45-69). Springer.

Kopec, G., & Chou, P. (s.f.). document Image Decoding Using Markov Source Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 16*(6), 602-617.

Laiphangbam, M., Raghu, G., & Chakravarthy, B. (2017). Document Layout Analysis using Multigaussian. *IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 747-752. doi:10.1109/ICDAR.2017.127

Marco, A., & Arnold M.W., S. (Octubre de 2003). Bidimensional Relations for Reading Order Detection.

MathWorks. (2021). *Centro de ayuda*. Obtenido de https://es.mathworks.com/help/index.html

Michelangelo, C., Annalisa, A., Donato, & Malerba. (s.f.). Relational Learning of Preference Relations for Reading Order Detection.

Michelangelo, C., Margherita, B., Giuseppe, A. P., & Donato, M. (2007). A Data Mining Approach to Reading Order Detection. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 924-928. doi:10.1109/ICDAR.2007.4377050

Ministerio de Salud Pública. (s.f.). *MSP*. Recuperado el 7 de Septiembre de 2017, de http://www.salud.gob.ec/calificacion-o-recalificacion-de-personas-con-discapacidad-2/

Nagy, G. (1992). A Prototype Document Image Analysis System for Technical Journals. *Computer, 25*(7), 10-22.

Naik, S., & Ramegowda, D. (Mayo de 2017). Segmentation of Unstructured Newspaper Documents. *International Journal of Advanced Engineering Research and Science, 4*, 79-83. doi:10.22161/ijaers.4.5.13

OpenCV. (2021). *OpenCV*. Obtenido de https://opencv.org/releases/

Pen-Shu, Y., Antoy, S., Litcher, A., & Rosenfeld, A. (1987). ADDRESS LOCATION ON ENVELOPES*. *Pattern Recognition, 20*(2), 213-227.

Pérez, J. M. (2010). *Inteligencia Computación Inspirada en la Vida.* España: SERVICIO DE PUBLICACIONES DE LA UNIVERSIDAD DE MÁLAGA.

Pratikakis, I., Gatos, B., Danatsas, D., & Perantonis, S. (agosto de 2005). Automatic Table Detection in Document Images. Athenas, Grecia.

Salud, O. M. (8 de Octubre de 2019). *https://www.who.int/es/news-room/detail/08-10-2019-who-launches-first-world-report-on-vision*. Obtenido de https://www.who.int/es/news-room/detail/08-10-2019-who-launches-first-world-report-on-vision

Sanahuja, S. D. (2017). *Sistemas de Control con Lógica Difusa: Métodos de Mamdani y de Takagi-Sugeno-Kang (TSK).*

Shinyama, Y. (2013). *pdfminer-docs*. Obtenido de pdfminer-docs: https://pdfminer-docs.readthedocs.io/pdfminer_index.html

Shuichi, T., & Haruo, A. (1992). Major Components of a Complete Text Reading System. *Proceedings of the IEEE,*, 80(7):1133-1149.

Social, M. d. (Julio de 2019). *Consejo Nacional para la Igualdad de Discapacidades*. Obtenido de https://www.consejodiscapacidades.gob.ec/wp-content/uploads/downloads/2019/08/Resumen-estad%C3%ADstico-de-discapacidad.pdf

Specsavers Ópticas. (s.f.). Obtenido de https://www.specsavers.es/ayuda-y-preguntas/%C2%BFqu%C3%A9-es-deficiencia-visual

Srihari, S., Wang, C.-H., Palumbo, P., & Hull, J. (1987). Recognizing Address Blocks on Mail Pieces: Specialized Tools and Problem-Solving Architecture . *AI Magazine, 8*(4), 25-40.

Stefano, F., Domenico, G., Domenico, R., & Floriana, E. (Noviembre de 2014). Abstract Argumentation for Reading Order Detection. Bari, Italia. doi:10.1145/2644866.2644883

Thomas, B. (Mayo de 2003). High Performance Document Layout Analysis. USA.

Tuan, A. T., Khuong, N.-A., & Nhat, Q. V. (Abril de 2018). Document Layout Analysis: A Maximum Homogeneous Region Approach. *1st International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, 1-5.

Vincent, N. (Mayo de 2007). Document image analysis for active reading. París, Francia.

wikiHow. (s.f.). *Cómo calcular datos atípicos*. Obtenido de https://es.wikihow.com/calcular-datos-at%C3%ADpicos

Yadav, V., & Ragot, N. (2016). Text extraction in document images: highlight on using corner points. *Document Analysis Systems (DAS)*, 281–286.

ANEXOS

ANEXO A: Segmentación y extracción de características

```
% Autor: Daniel Fraga
% 14-12-2020
% esta función extrae la información de los BBox de las imágenes

clc, clear;
dirIn = 'C:\Users\BAYRON
FRAGA\Desktop\Matlab\tesis\Parte2_orden\ProDF\Programas\BaseDeDatos\Ba
seC2';
% dirIn='C:\Users\BAYRON
FRAGA\Desktop\Matlab\tesis\Parte2_orden\ProDF\Programas\BaseC1';
% dirIn='C:\Users\BAYRON
FRAGA\Desktop\Matlab\tesis\Parte2_orden\ProDF\Programas\BaseC3';
listF =dir(dirIn);
tamx= size(listF,1);
for i =5:tamx
   filename = listF(i).name
   clipboard('copy', filename(1:end-4))
   strx = strcat(dirIn,'\',filename);
   vabsm=progprin(dirIn,filename,2.5);
    figure(11)
    imshow(ones(1000));
    for n=1:size(vabsm,1)
      rectangle('Position',vabsm.BoundingBox(n,:),'EdgeColor','g','LineWidth',2)
    end
   vabsm=elmF(vabsm);
    figure(12)
    imshow(ones(1000));
    for n=1:size(vabsm,1)
        rectangle('Position',(vabsm.BoundingBox(n,:)),'EdgeColor','g','LineWidth',2)
    end
   vabsm=elmUD(vabsm);
    figure(13)
    imshow(ones(1000));
    for n=1:size(vabsm,1)
        rectangle('Position',(vabsm.BoundingBox(n,:)),'EdgeColor','g','LineWidth',2)
    end
   vabsm=elmRS(vabsm);
    figure(14)
    imshow(ones(1000));
    for n=1:size(vabsm,1)
        rectangle('Position',(vabsm.BoundingBox(n,:)),'EdgeColor','g','LineWidth',2)
    end
```

```
   vabsm =alinear(vabsm);
   namedatos=strcat(dirIn,'_matf3\',filename(1:end-3),'mat');
    figure(15)
    imshow(ones(1000));
    for n=1:size(vabsm,1)
        rectangle('Position',(vabsm.BoundingBox(n,:)),'EdgeColor','g','LineWidth',2)
    end
    save(namedatos,'vabsm')
 end


% Esta función extrae las características de los bbox y las devuelve
% normalizadas en un plano de 1000x1000 pixeles
function [psob]=progprin(dim,filename,ch) %psob
%% Carga imagen
dirim = strcat(dim,'\',filename);
imOr=imread(dirim);
imOr=im2bw(imOr,0.9); %para imágenes a color
SE=strel('rectangle',[2 1]);   % atenuación de rasgos de letras
imOr = imclose (imOr, SE);
im=~imOr;
%% Ecuentra el Bbox  de los elementos
ax=findbox(im);
vv=mode(ax.BoundingBox(:,4))% moda valor vertical bbox
vh=mode(ax.BoundingBox(:,3));
%% dividir la matriz en 3 para el análisis de el mtld
tamim=size(im);
div=round(tamim(2)/3);
im1=im(:,1:div);
im2=im(:,div:div*2);
im3=im(:,div*2+1:end);
%% encontrar el mtld (valor medio de línea de texto)
mtld1=findmtld(im1,vv);
mtld2=findmtld(im2,vv);
mtld3=findmtld(im3,vv);
mtld=[mtld1 mtld2 mtld3];
mtld=mode(mtld);
%% LRSA
% Horizontal
imconh=RLSAh(~im,vh*2);%%300
 %Vertical
imconv=RLSAv(~im,mtld*1); %%300
 %AND
mrlsa=or(imconv,imconh);
SE=strel('rectangle',[round(vv*0.3) round(vh*0.45)]);   % atenuación de rasgos de
letras
closeBW = imclose (imconh, SE);
a1=findmtld(~closeBW(:,1:div),vv);
```

```
a2=findmtld(~closeBW(:,div:div*2),vv);
a3=findmtld(~closeBW(:,div*2+1:end),vv);
mtldb=mode([a1 a2 a3]);% con la imagen de rasgos atenuados
SE=strel('line',round(ch*vh),0); %ch:2.5
mdil=imopen(mrlsa,SE); %dilatación
% Calcular propiedades de los objetos de la imagen
propied= regionprops(~mdil);
propied(end,:)=[];
propied=struct2table(propied);
pmone=find([propied.BoundingBox(:,3)]>15);
propied=propied(pmone,:);
pmone=find([propied.BoundingBox(:,4)]>25);
propied=propied(pmone,:);
nrec=size(propied,1);
pAbs=propied.BoundingBox; % se almacena solo los valores de los recuadros
de información
pAbs(:,3)=pAbs(:,1)+pAbs(:,3);
pAbs(:,4)=pAbs(:,2)+pAbs(:,4);
aux=ones(size(imOr));
for n=1:size(propied,1)
   x=round(pAbs(n,1));
   if(x==0)
      x=1;
   end
   y=round(pAbs(n,2));
   if(y==0)
      y=1;
   end
   dx=round(pAbs(n,3));
   dy=round(pAbs(n,4));
   aux(y:dy,x:dx)=0;
end
aux=imfill(~aux,'holes');
psob=regionprops(aux);
psob=struct2table(psob);
%% normalizar valores con respecto al tamaño de la página
psob.Area=(psob.Area/(tamim(1)*tamim(2)))*1000;
psob.Centroid(:,1)=(psob.Centroid(:,1)/tamim(2))*1000;
psob.Centroid(:,2)=(psob.Centroid(:,2)/tamim(1))*1000;
psob.BoundingBox(:,1)=(psob.BoundingBox(:,1)/tamim(2))*1000;
psob.BoundingBox(:,2)=(psob.BoundingBox(:,2)/tamim(1))*1000;
psob.BoundingBox(:,3)=(psob.BoundingBox(:,3)/tamim(2))*1000;
psob.BoundingBox(:,4)=(psob.BoundingBox(:,4)/tamim(1))*1000;
vv=(vv/tamim(1))*1000;
vh=(vh/tamim(2))*1000;
psob(psob.BoundingBox(:,3)>550&psob.BoundingBox(:,4)>550,:)=[];
psob(psob.BoundingBox(:,4)>850,:)=[];
nrec=size(psob,1)
```

```
dats=table(vv*ones(nrec,1),vh*ones(nrec,1),(1:nrec)',tamim(2)*ones(nrec,1),tami
m(1)*ones(nrec,1),'VariableNames',{'vv','vh','id','Torx','Tory'});
psob=[psob dats];
end


% esta función devuelve la línea media de texto.
function [mcl]=findmtld(im,vv)
t=size(im,1);
f=im;
h = zeros(1,t);
x=1:t;
for j = 1:t
    h(j) = sum(f(j,:));
end
poitm=max(h);
h(h>poitm*0.5)=poitm*0.5;
pmax=islocalmax(h,'MinSeparation',1.3*vv);%puntos mínimos
x1=x(pmax);%localización de maximos
i1=size(x1,2);%escoge el segundo valor
dx1=0;
for j=2:i1-1
        dx1(j-1)=x1(j)-x1(j-1);
end
mcl=dx1;



% Esta función elimina los valores atípicos de los BBox
function [ax]=findbox(im)
aux=regionprops(im);
aux=struct2table(aux);
pmone=find([aux.BoundingBox(:,3)]>15);% posicion de los valores mayores a "1"
OOOOJOOO
ax=aux(pmone,:);
%% determinar valores atípicos
q2=median(ax.Area);%mediana
q1=prctile(ax.Area,25);%primer cuartil
q3=prctile(ax.Area,75);%tercer cuartil
rango=q3-q1;
%% Límites internos
tq3=q3+rango*1.5;
tq1=q1-rango*1.5;
if (tq1<=0)
    tq1=0;
end
%% Límites externos
txq3=q3+rango*3;
```

```
txq1=q1-rango*3;
if (txq1<=0)
   txq1=0;
end
%% menores al límite inf
pmin=find([ax.Area]<txq1);% posicion
%% mayores al Límite sup
pmax=find([ax.Area]>txq3);% posicion
%% propiedades sin unos y sin valores atípicos
ax([pmax; pmin],:)=[];
end


%LRSA horizontal
function result=RLSAh(image,hor_thresh)
      ones_count=0;
      hor_image=image;
      [m,n]=size(image);
      for i=1:m
         for j=1:n
            if(image(i,j)==0)
                  if(ones_count<=hor_thresh)
                     hor_image(i,j-ones_count:j-1)=0;
                  end
                  ones_count=0;
            else
                  ones_count=ones_count+1;
            end
         end
         if(ones_count>=1)
            hor_image(i,n)=0;
            ones_count=0;
         end
      end
      result= hor_image;
end
%LRSA vertical
function result=RLSAv(image,v_thresh)
      ones_count=0;
      v_image=image;
      [m,n]=size(image);
      for j=1:n
         for i=1:m
             if(image(i,j)==0)
                  if(ones_count<=v_thresh)
                     v_image(i-ones_count:i-1,j)=0;
                  end
                  ones_count=0;
```

```
            else
                ones_count=ones_count+1;
            end
        end
        if(ones_count>=1)
            v_image(m,j)=0;
            ones_count=0;
        end
    end
    result= v_image;
end


% Elimina los BBox aislados
function [vabsm] =elmF(vabsm)
 dxm=median(vabsm.BoundingBox(:,3));
 dym=median(vabsm.BoundingBox(:,4));
 dx=(vabsm.BoundingBox(:,3)*100)/dxm;
 dy=(vabsm.BoundingBox(:,4)*100)/dym;
 [cs,cb,ci,cd]=vecindad(vabsm);
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)+vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)+vabsm.BoundingBox(:,2);
addt=table(0,[0 0],[0 0 0 0] ,0 ,0 ,0
,0,0,'VariableNames',{'Area','Centroid','BoundingBox','vv','vh','id','Torx','Tory'});
vabsm=[vabsm;addt];
cs=[cs;zeros(1,size(cs,2))];
cb=[cb;zeros(1,size(cb,2))];
ci=[ci;zeros(1,size(ci,2))];
cd=[cd;zeros(1,size(cd,2))];
cs(cs==0)=size(vabsm,1);
cb(cb==0)=size(vabsm,1);
ci(ci==0)=size(vabsm,1);
cd(cd==0)=size(vabsm,1);
xi = vabsm.BoundingBox(:,1) ;
xf = vabsm.BoundingBox(:,3) ;
yi = vabsm.BoundingBox(:,2) ;
yf = vabsm.BoundingBox(:,4) ;
dci=xi-xf(ci(:,1)) ;  %ci
dci(xf(ci(:,1))==0)=0;
fdci=ones(size(dci,1),1);
fdci(xf(ci(:,1))==0)=0;
dcd=xi(cd(:,1))-xf;  %cd
dcd(xi(cd(:,1))==0)=0;
fdcd=ones(size(dcd,1),1);
fdcd(xi(cd(:,1))==0)=0;
dcs=yi-yf(cs(:,1));  %cs
dcs(yf(cs(:,1))==0)=0;
fdcs=ones(size(dcs,1),1);
```

```
fdcs(yf(cs(:,1))==0)=0;
dcb=yi(cb(:,1))-yf;  %cb
dcb(yi(cb(:,1))==0)=0;
fdcb=ones(size(dcb,1),1);
fdcb(yi(cb(:,1))==0)=0;
cs(cs==size(vabsm,1))=0;
cs(end,:)=[];
cb(cb==size(vabsm,1))=0;
cb(end,:)=[];
ci(ci==size(vabsm,1))=0;
ci(end,:)=[];
cd(cd==size(vabsm,1))=0;
cd(end,:)=[];
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)-vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)-vabsm.BoundingBox(:,2);
vabsm(end,:)=[];
dats=[dci  dcd  dcs dcb ]; % distancia con respecto a la vecindad
dats(end,:)=[];
dats=[dx dy dats ];% distancia relativa en x, y
fis=readfis('ClasificadorV2');%cargo fuzzy
salida=round(evalfis(dats(:,1:2),fis));%evaluo solo con las distancias
vabsm(find(salida==0),:)=[];% elimina recuadros etiquetados con cero
vabsm.id= (1:size(vabsm,1))'; % reajuste del id de cada recuadro
% %  ------------------------------------
%  figure(2)
%  imshow(ones(1000));
%   for n1=1:size(vabsm,1)
%
rectangle('Position',vabsm.BoundingBox(n1,:),'EdgeColor','g','LineWidth',2)
%   end
% %    ------------------------------------
end


function [vabsm] =elmUD(vabsm)
[~, s] = sort(vabsm.BoundingBox(:, 2));%ordenar con respecto a y
vabsm=vabsm(s, :);
vabsm.id=(1:size(vabsm,1))';
 aux=vabsm;%para graficar
[cs,cb,ci,cd]=vecindad(vabsm);
 %% eliminación de recuadros continuos
 vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)+vabsm.BoundingBox(:,1);
 vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)+vabsm.BoundingBox(:,2);
 pb=0;
 ps=0;
 cofrep=0;
a=cs(:,1);
[~, ind_unicos] = unique(a,'stable');
```

```
ind_repetidos = setdiff(1:length(a), ind_unicos);
valores_repetidos = a(ind_repetidos, 1);
vrcs=unique(valores_repetidos,'stable');
vrcs(vrcs==0)=[];
a=cb(:,1);
[~, ind_unicos] = unique(a,'stable');
ind_repetidos = setdiff(1:length(a), ind_unicos);
valores_repetidos = a(ind_repetidos, 1);
vrcb=unique(valores_repetidos,'stable');
vrcb(vrcb==0)=[];
for i=1:size(vrcs,1)
    if isempty(vrcs)==0
        ps(i,1:size(find(cs(:,1)==vrcs(i)),1))=find(cs(:,1)==vrcs(i))';
    end
end
for i=1:size(vrcb,1)
    if isempty(vrcb)==0
        pb(i,1:size(find(cb(:,1)==vrcb(i)),1))=find(cb(:,1)==vrcb(i))' ;
    end
end
%% reestructuración de matrices con elementos repetidos
npmax=max(size(pb),size(ps));
matp=zeros(npmax);
matp(1:size(pb,1),1:size(pb,2))=pb;
pb=matp;
matp=zeros(npmax);
matp(1:size(ps,1),1:size(ps,2))=ps;
ps=matp;
for i=1:size(matp,1)
    for j=1:size(matp,1)
        flag=ps(i,:)==pb(j,:);
        if sum(flag)==size(matp,2); % busca coincidencias entre las matrices
(repetidos)
            cofrep(i,:)=1;
        end
    end
end
cofrep=find(cofrep==1);
cofrep=ps(cofrep,:);
if sum(sum(cofrep))~=0
for i=1:size(cofrep,1)
    temp2=zeros(1,size(cofrep,2));
    temp=cofrep(i,:);
    temp(temp==0)=[];
        if sum(size(temp)==size(cd(temp)))==0
            temcd=cd(temp)';
        end
        if sum(size(temp)==size(ci(temp)))==0
```

```
            temci=ci(temp)';
        end
        if sum(size(temp)~=size(cd(temp)))==0
            temcd=cd(temp);
            temcd=unique(temcd,'stable');
        end
        if sum(size(temp)~=size(ci(temp)))==0
            temci=ci(temp);
            temci=unique(temci,'stable');
        end
        a=[temp temci temcd]';
        [~, ind_unicos] = unique(a,'stable');
        ind_repetidos = setdiff(1:length(a), ind_unicos);
        valores_repetidos = a(ind_repetidos, 1);
        temp=unique(valores_repetidos,'stable')';
        temp(temp==0)=[];
         temp2(1:size(temp,2))=temp;
        cofrep(i,:)=temp2;
    if sum(temp)~=0
    vabsm.BoundingBox(temp(1),3)=max(vabsm.BoundingBox(temp,3));
%reestructuración de bloques en la misma línea de texto
    vabsm.BoundingBox(temp(1),1)=min(vabsm.BoundingBox(temp,1));
%reestructuración de bloques en la misma línea de texto
    vabsm.BoundingBox(temp(1),4)=max(vabsm.BoundingBox(temp,4));
%reestructuración de bloques en la misma línea de texto
    vabsm.BoundingBox(temp(1),2)=min(vabsm.BoundingBox(temp,2));
%reestructuración de bloques en la misma línea de texto
        for x1=1:size(temp,2)

rectangle('Position',(aux.BoundingBox(temp(x1),:)),'EdgeColor','r','LineWidth',2);
        end
    end
end
%% eliminar datos repetidos
j=1+size(cofrep,1); %para eliminar desde atrás para que no altere la matriz
for i=1:size(cofrep,1)
        k=j-i;
        temp=cofrep(k,:);
        temp(temp==0)=[];
        vabsm.Area(temp(2:end),:)=0;
        cs(temp(2:end),:)=0;
        cb(temp(2:end),:)=0;
        ci(temp(2:end),:)=0;
        cd(temp(2:end),:)=0;
end
end
vabsm((find(vabsm.Area==0)),:)=[];
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)-vabsm.BoundingBox(:,1);
```

```
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)-vabsm.BoundingBox(:,2);
vabsm.id=(1:size(vabsm,1))'; %asigna id a los recuadros ordenados por el valor
de y
end


% Une los BBox vecinos
function [vabsm] =elmRS(vabsm)
cjv=1;
while ~isempty(cjv)
dxm=median(vabsm.BoundingBox(:,3));% determina la mediana de los valores
en x,y
dym=median(vabsm.BoundingBox(:,4));
[~,~,ci,cd]=vecindad(vabsm);
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)+vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)+vabsm.BoundingBox(:,2);
addt=table(0,[0 0],[0 0 0 0] ,0 ,0 ,0
,0,0,'VariableNames',{'Area','Centroid','BoundingBox','vv','vh','id','Torx','Tory'});
vabsm=[vabsm;addt];
ci=[ci;zeros(1,size(ci,2))];
cd=[cd;zeros(1,size(cd,2))];% aumenta una fila de ceros
ci(ci==0)=size(vabsm,1); %cambia se es 0 a el último valor de vabsm para evitar
errores
cd(cd==0)=size(vabsm,1);
xi = vabsm.BoundingBox(:,1) ;
xf = vabsm.BoundingBox(:,3) ;
dcd=xf(cd(:,1))-xi;  %cd
cj=find(dcd<(dxm*0.05+dxm)&dcd>0);%cajas juntas
cjv=vabsm(cj,:);
[~, s] = sort(cjv.BoundingBox(:, 1));%ordenado según x para reestructurar
cjv=cjv(s, :);
cjd=cd(cjv.id);
for n1=1:size(cjv,1)
    nr=find(cjd(n1)==cjv.id);%números repetidos
    if nr~=0
        cjv(nr,:)=[];          %elimino los valores repetidos
    end
end
aux=cd(cjv.id);
for n1=1:size(cjv,1)
      vabsm.BoundingBox(cjv.id(n1),3)=vabsm.BoundingBox(aux(n1),3);
end
vabsm(aux,:)=[];
vabsm.id=(1:size(vabsm,1))';
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)-vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)-vabsm.BoundingBox(:,2);
end
vabsm(end,:)=[];
```

```
vabsm.id= (1:size(vabsm,1))';
% %----------------------------------
%  figure(5)
%  imshow(ones(1000));
%   for n1=1:size(vabsm,1)
%
rectangle('Position',vabsm.BoundingBox(n1,:),'EdgeColor','g','LineWidth',2)
%   end
% %----------------------------------
end



% Esta función redimensiona los BBox
function [vabsm] =alinear(vabsm)
vabsm(vabsm.Area==0,:)=[];
[cs,cb,ci,cd]=vecindad(vabsm);
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)+vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,2)+vabsm.BoundingBox(:,4);
for i=1:size(vabsm,1)
   if cs(i)~=0
      xti=vabsm.BoundingBox(cs(i),1);
      xtf=vabsm.BoundingBox(cs(i),3);
      xif=ci(i);
      if xif==0
         xif=0;
      else
         xif=vabsm.BoundingBox(ci(i),3);
      end
      xdi=cd(i);
      if xdi==0
         xdi=20000;
      else
         xdi=vabsm.BoundingBox(cd(i),1);
      end
      if xtf>vabsm.BoundingBox(i,3)&&xtf<xdi
         vabsm.BoundingBox(i,3)=xtf;
      end
      if (xti)<vabsm.BoundingBox(i,1)&&xti>xif
         vabsm.BoundingBox(i,1)=xti;
      end
   end
end
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)-vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)-vabsm.BoundingBox(:,2);
% %--------------------
% m=ones(1000);
% figure(6)
```

```
% imshow(m);
%     for n1=1:size(vabsm,1)
%
rectangle('Position',vabsm.BoundingBox(n1,:),'EdgeColor','g','LineWidth',2)
%     end
% %--------------------
end
```

ANEXO B: Identificación de Orden de Lectura

```
% Autor: Daniel Fraga
% 14-12-2020
% Esta función extrae el Orden de lectura de los BBox de las imágenes
clear, clc
dirc='C:\Users\BAYRON
FRAGA\Desktop\Matlab\tesis\Parte2_orden\ProDF\Programas\BaseDeDatos\Ba
seC2_matf3\';
dirc2='C:\Users\BAYRON
FRAGA\Desktop\Matlab\tesis\Parte2_orden\ProDF\Programas\BaseDeDatos\Ba
seC2\';
listF =dir(dirc);
listF([1 2])=[];
for x1=62: size(listF,1)
filename = listF(x1).name;
load([dirc filename]);
vabsm(vabsm.BoundingBox(:,4)>8*median(vabsm.BoundingBox(:,4)),:)=[];
vabsm(vabsm.Area==0,:)=[];
vabsm=redtam(vabsm);
vabsm.id=(1:size(vabsm,1))';
aux=vabsm;
xm=mode(vabsm.BoundingBox(:,3));
%% pie de pag
aux.BoundingBox(:,3)=aux.BoundingBox(:,3)+aux.BoundingBox(:,1);
aux.BoundingBox(:,4)=aux.BoundingBox(:,4)+aux.BoundingBox(:,2);
xi = aux.BoundingBox(:,1);
xf = aux.BoundingBox(:,3);
yi = aux.BoundingBox(:,2);
yf = aux.BoundingBox(:,4);
aux.BoundingBox(:,3)=aux.BoundingBox(:,3)-aux.BoundingBox(:,1);
aux.BoundingBox(:,4)=aux.BoundingBox(:,4)-aux.BoundingBox(:,2);
[cs,cb,ci,cd]=vecindad(vabsm);
vv=vabsm.vv(1);
pie= vabsm(end,:);
resp=yi(pie.id)-yf(cs(pie.id));
auxi=pie.id;
```

```
auxd=pie.id;
cpied=[];
cpiei=[];
cpie=[];
num=1;
if resp>vv*2
   while ci(auxi)~=0
      cpiei(num)=ci(auxi);
      num=num+1;
      auxi=ci(auxi);
   end
   cpiei=flip(cpiei);
   num=1;
   while  cd(auxd)~=0
      cpied(num)=cd(auxd);
      num=num+1;
      auxd=cd(auxd);
   end
   cpie=[cpiei pie.id cpied];
    aux(cpie',:)=[];
end
    divp=find(aux.BoundingBox(:,3)>800);
    divp=aux.id(divp);
    cad=[];
    cadena=[];
for i=1:size(divp,1) %divide la página
    x=1:find(aux.id==divp(i));
    mtemp=aux(x,:);
    if size(mtemp,1)>2
       [cadena{i},ncadena]=ordenLec(mtemp(1:end-1,:),xm) ;
       cadena{i}=aux.id(cadena{i});
       cadena{i}=[cadena{i};mtemp.id(end)];
       aux(x,:)=[];
    else
       cadena{i}=aux.id(x);
       aux(x,:)=[];
    end
end
if size(aux,1)>1
    [cadena{end+1},ncadena]=ordenLec(aux,xm);
    cadena{end}=aux.id(cadena{end});
else
     cadena{end+1}=aux.id;
end
for i=1:size(cadena,2)
    cad=[cad;cadena{i}];
end
cad=cad';
```

```
if ~isempty(cpie)
cad=[cad cpie];
end
filename(end-3:end)=".PNG";
strx= strcat(dirc2,filename);
imOr=imread(strx);
vabsm.Area=(vabsm.Area*(vabsm.Tory(1)*vabsm.Torx(1)))*1000;
vabsm.Centroid(:,1)=(vabsm.Centroid(:,1)*vabsm.Torx(1))/1000;
vabsm.Centroid(:,2)=(vabsm.Centroid(:,2)*vabsm.Tory(1))/1000;
vabsm.BoundingBox(:,1)=(vabsm.BoundingBox(:,1)*vabsm.Torx(1))/1000;
vabsm.BoundingBox(:,2)=(vabsm.BoundingBox(:,2)*vabsm.Tory(1))/1000;
vabsm.BoundingBox(:,3)=(vabsm.BoundingBox(:,3)*vabsm.Torx(1))/1000;
vabsm.BoundingBox(:,4)=(vabsm.BoundingBox(:,4)*vabsm.Tory(1))/1000;
figure(10)
imshow(imOr,[90 100]);
hold on
for n1=1:size(vabsm,1)
       rectangle('Position',vabsm.BoundingBox(n1,:),'EdgeColor','g','LineWidth',2)
end
     graf=vabsm.Centroid(cad,:);
     ordf=plot(graf(:,1),graf(:,2),'b','LineWidth',4);
     hold off
     %Almacena los datos
     dirc1='C:\Users\BAYRON
FRAGA\Desktop\Matlab\tesis\Parte2_orden\ProDF\Programas\BaseDeDatos\Ba
seC3_ord';
     filename(end-3:end)=".mat"
     load([dirc1 'p\' filename] );
     cad(end)=[];
     graf=vabsm.Centroid(cad,:);
     ordf1=plot(graf(:,1),graf(:,2),'r','LineWidth',4);
        hold off
    namec=strcat(dirc(1:end-7),'_ordp4\',filename(1:end-3),'mat');
    save(namec,'cad')
end



% Redimensiona los valores de BBox de la imagen
function [vabsm] =redtam(aux)
[~, s] = sort(aux.BoundingBox(:, 2));%ordenar con respecto a y
aux=aux(s, :);
aux.id=(1:size(aux,1))';
vabs=aux;
vabs.BoundingBox(:,3)=vabs.BoundingBox(:,3)+vabs.BoundingBox(:,1);
vabs.BoundingBox(:,4)=vabs.BoundingBox(:,2)+vabs.BoundingBox(:,4);
yi = vabs.BoundingBox(:,2);
yf = vabs.BoundingBox(:,4);
```

```
[tcs,tcb,tci,tcd]=vecindad(aux);
x=(1:size(tci,1))';
 %% selección de recuadros sin vecinos en la izquierda y derecha
vzi=[sum((tci>0.5),2) x];
vzd=[sum((tcd>0.5),2) x];
vzi(vzi(:,1)>0,:)=[];
vzd(vzd(:,1)>0,:)=[];
a=[vzi(:,2);vzd(:,2)];
[~, ind_unicos] = unique(a,'stable');
% ind_unicos(ind_unicos==0)=[]
ind_repetidos = setdiff(1:length(a), ind_unicos);
valores_repetidos = a(ind_repetidos, 1);
vzt=unique(valores_repetidos,'stable');
vzt(vzt==0)=[];
 %% reestructuración de valores para recuadros sin vencinos
vabsm=vabs;
vv=aux.vv(1);
auxSup=tcs(vzt);
auxBaj=tcb(vzt);
restSup=zeros(size(vzt,1),1);
restInf=zeros(size(vzt,1),1);
for n1=1:size(vzt,1)
    if(auxSup(n1)~=0)
       restSup(n1)=yi(vzt(n1))-yf(auxSup(n1));
    end
    if(auxBaj(n1)~=0)
       restInf(n1)=yi(auxBaj(n1))-yf(vzt(n1));
    end
end
inexpi=((restSup>vv*4.5))|((restInf>vv*4.5));%1.5
vzt=vzt.*inexpi;
vzt(vzt==0)=[];
restSup=restSup.*inexpi;
restSup(inexpi==0)=[];
restInf=restInf.*inexpi;
restInf(inexpi==0)=[];
if ~isempty(vzt)
   for n1=1:size(vzt,1)
          vabsm.BoundingBox(vzt(n1),1)=0.5;%%%%%%%%%%%% descomentar!!!
          vabsm.BoundingBox(vzt(n1),3)=1000;
   end
end
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)-vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)-vabsm.BoundingBox(:,2);
vabsm.id= (1:size(vabsm,1))';
[tcs,tcb,tci,tcd]=vecindad(vabsm);
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)+vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,2)+vabsm.BoundingBox(:,4);
```

```
yi = vabs.BoundingBox(:,2);
yf = vabs.BoundingBox(:,4);
auxSup=tcs(vzt);
auxBaj=tcb(vzt);
restSup2=zeros(size(vzt,1),1);
restInf2=zeros(size(vzt,1),1);
if ~isempty(vzt)
   for n1=1:size(vzt,1)
      if(auxSup(n1)~=0)
         restSup2(n1)=yi(vzt(n1))-yf(auxSup(n1));
      end
      if(auxBaj(n1)~=0)
         restInf2(n1)=yi(auxBaj(n1))-yf(vzt(n1));
      end
   end
end
restInf(restInf>vv*3.5)=vv*3.5;
restSup(restSup>vv*3.5)=vv*3.5;
restInf2(restInf2>vv*3.5)=vv*3.5;
restSup2(restSup2>vv*3.5)=vv*3.5;
inexpi=(restSup>restSup2)|(restInf>restInf2);
vzt(inexpi==0)=[];
vabsm.BoundingBox(:,3)=vabsm.BoundingBox(:,3)-vabsm.BoundingBox(:,1);
vabsm.BoundingBox(:,4)=vabsm.BoundingBox(:,4)-vabsm.BoundingBox(:,2);
if ~isempty(vzt)
   for n1=1:size(vzt,1)
         vabsm.BoundingBox(vzt(n1),:)=aux.BoundingBox(vzt(n1),:);
   end
end
%%------------------------
%     figure(3)
%    imshow(ones(1000));
%    for n=1:size(vabsm,1)
%      rectangle('Position',vabsm.BoundingBox(n,:),'EdgeColor','g','LineWidth',2)
%    end
%%------------------------
end


function [tcs,tcb,tci,tcd]=vecindad(vabs)
aux=vabs;
vabs.BoundingBox(:,3)=vabs.BoundingBox(:,3)+vabs.BoundingBox(:,1);
vabs.BoundingBox(:,4)=vabs.BoundingBox(:,2)+vabs.BoundingBox(:,4);
bbox=vabs.BoundingBox;
tame = size(bbox,1);
tame=(1:tame)';
xi = [bbox(:,1) ];
xf = [bbox(:,3) ];
```

```
yi = [bbox(:,2) ];
yf = [bbox(:,4) ];
 for n=1:size(bbox,1)
      cns1=(xi(n)<xf);%coeficientes no seleccionados
      cns2=(xf(n)>xi);
      cns1=tame(cns1);
      cns2=tame(cns2);
      cc=zeros(size(cns1,1),1);
      if (size(cc,1)>=1)
      for n1=1:size(cns2,1)
           caux=find(cns1==cns2(n1));
           if(caux>=1)
           cc(n1)=cns1(caux);
           end
      end
      cc(cc==0)=[];
      end
    cc(find(cc==n))=[];
    a(:,1)=yi(n)*ones(size(cc,1),1)-yf(cc);
    a(:,2)=(cc)';
    a=(a(:,1)>0).*a;
    a1=a(:,1);
    a2=a(:,2);
    a1(a1==0)=[];
    a2(a2==0)=[];
    a=[a1 a2];
    if(isempty(a)==1)
     a=[0 0];
    end
    cs=find(a(:,1)==min(a(:,1)));% Cs: cuadros superiores
    cs=a(cs,2);
    b(:,1)=yf(n)-yi(cc);
    b(:,2)=(cc)';
    b=(b(:,1)<0).*b;
    b1=b(:,1);
    b2=b(:,2);
    b1(b1==0)=[];
    b2(b2==0)=[];
    b=[b1 b2];
    if(isempty(b)==1)
     b=[0 0];
    end
    cb=find(b(:,1)==max(b(:,1)));
    cb=b(cb,2);
    %% Derecha e izquierda
      cns3=(yi(n)<yf);%coeficientes no seleccionados
      cns4=(yf(n)>yi);
      cns3=tame(cns3);
```

```
    cns4=tame(cns4);
    cc2=zeros(size(cns3,1),1);
    if (size(cc2,1)>=1)
    for n1=1:size(cns4,1)
        caux=find(cns3==cns4(n1));
        if(caux>=1)
        cc2(n1)=cns3(caux);
        end
    end
    cc2(cc2==0)=[];
    end
  cc2(find(cc2==n))=[];
  c(:,1)=xi(n)*ones(size(cc2,1),1)-xf(cc2);
  c(:,2)=(cc2)';
  c=(c(:,1)>0).*c;
  c1=c(:,1);
  c2=c(:,2);
  c1(c1==0)=[];
  c2(c2==0)=[];
  c=[c1 c2];
  if(isempty(c)==1)
   c=[0 0];
  end
  ci=find(c(:,1)==min(c(:,1)));
  ci=c(ci,2);
  d(:,1)=xf(n)-xi(cc2);
  d(:,2)=(cc2)';
  d=(d(:,1)<0).*d;
  d1=d(:,1);
  d2=d(:,2);
  d1(d1==0)=[];
  d2(d2==0)=[];
  d=[d1 d2];
  if(isempty(d)==1)
   d=[0 0];
  end
  cd=find(d(:,1)==max(d(:,1)));
  cd=d(cd,2);
  vr(n,1)=n;
  tcs(n,1:size(cs,1))=cs';
  tcb(n,1:size(cb,1))=cb';
  tci(n,1:size(ci,1))=ci';
  tcd(n,1:size(cd,1))=cd';
  clear a b ci cd c d cs csi
 end
end
```

```
% extrae las cadenas de texto y evalua con el Clasificador difuso, devuelve
% el orden de lectura de la página segmentada
function [cadena,ncadena] =ordenLec(vabsm,xm)
vabsm.id=(1:size(vabsm,1))';
cadena=[];
ncadena=[];
n=1;
vord=0;
temporal=vabsm;
temporal.BoundingBox(:,3)=vabsm.BoundingBox(:,3)+vabsm.BoundingBox(:,1);
temporal.BoundingBox(:,4)=vabsm.BoundingBox(:,2)+vabsm.BoundingBox(:,4);
aux=vabsm;
[~,cb,ci,cd]=vecindad(aux);
num=1;
orden=1;
while 1
% continúa desde el superior
   cadena(orden)=aux.id(num);
   num2=num;
   aux(num,:)=[];
   orden=orden+1;
   num=find(aux.id==cb(cadena(end)));
   tama=vabsm.BoundingBox(cadena(end),3);
   if cb(cadena(end))~=0
       tamcb=vabsm.BoundingBox(cb(cadena(end)),3);
   else
        tamcb=0;
   end
    if isempty(num)||num==0||tamcb>tama*2
       if size(ncadena,2)==0
          ncadena{n}=cadena;
          cadena=[];
       else
          cadena(cadena==0)=[];
          ncadena{n}=cadena;
          cadena=[];
       end
       n=n+1;
        if  isempty(aux)
            break
        end
        num=1;
    end
end
for i=1:size(ncadena,2)
%        vord(i)=mean(vabsm.BoundingBox(ncadena{i},1))
        x(i)=min(temporal.BoundingBox(ncadena{i},1));
        xmx(i)=max(temporal.BoundingBox(ncadena{i},3));
```

```
        y(i)=min(temporal.BoundingBox(ncadena{i},2));
        ymx(i)=max(temporal.BoundingBox(ncadena{i},4));
end
x=xmx;
y=ymx;
teta=atan(y./x)';
dP=sqrt((x).^2+y.^2)';
fis=readfis('OrdCadEscrito');%cargo fuzzy OrdCadEscrito2
vord=evalfis([teta dP],fis);
[~, s] = sort(vord,'descend');
ncadena=ncadena(:,s);
cadena=[];
    for i=1:size(ncadena,2)
        cadena=[cadena ncadena{i}];
    end
end
```