INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELÉCTRICA

“DISEÑO DE UN INTÉRPRETE BRAILLE-ESPAÑOL / ESPAÑOL-BRAILLE UTILIZANDO LA BAM ALFA-BETA”

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTAN

JESSICA FLORES HERNÁNDEZ

UZIEL GALLEGOS ESPINOSA

ASESORES

DRA. MARÍA ELENA ACEVEDO MOSQUEDA

M. EN C. JAFETH ASCENCIÓN ALONSO CARREÓN

M. EN C. GENARO ZAVALA MEJÍA

MÉXICO, D.F., DICIEMBRE DE 2013

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

TEMA DE TESIS

QUE PARA OBTENER EL TITULO DE INGENIERO EN COMUNICACIONES Y ELECTRÓNICA
POR LA OPCIÓN DE TITULACIÓN TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL
DEBERA (N) DESARROLLAR

C. JESSICA FLORES HERNANDEZ
C. UZIEL GALLEGOS ESPINOSA

"DISEÑO DE UN INTÉRPRETE BRAILLE-ESPAÑOL/ESPAÑOL-BRAILLE UTILIZANDO LA BAM ALFA-BETA"

DISEÑAR E IMPLEMENTAR UN INTÉRPRETE BRAILLE-ESPAÑOL/ESPAÑOL-BRAILLE UTILIZANDO EL MODELO DE LAS MEMORIAS ASOCIATIVAS BIDIRECCIONALES ALFA-BETA.

- INTRODUCCIÓN
- ANTECEDENTES
- MARCO TEÓRICO
- FUNDAMENTOS PARA EL PROCESAMIENTO DIGITAL DE IMÁGENES
- DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE
- TRABAJOS FUTUROS
- CONCLUSIONES

MÉXICO D.F. A 12 DE AGOSTO DE 2014

ASESORES

M. EN C. MARIA ELENA ACEVEDO MOSQUEDA

ING. JAFETH ASCENCIÓN ALONSO CARREÓN

ING. GENARO ZAVALA MEJIA

ING. PATRICIA LORENA RAMIREZ RANGEL
JEFE DEL DEPARTAMENTO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRONICA

Índice General

Índice de Figuras

Índice de Tablas

CAPÍTULO 1

Introducción

1.1. Contexto

La discapacidad visual es hoy en día uno de los temas que más preocupan al gobierno, al sector salud, educativo y empresarial y a la sociedad en general, debido a que por un lado, cada vez son más las personas con ceguera o debilidad visual que demandan a estos sectores una mayor oferta de oportunidades y por el otro, las organizaciones creadas a favor de los discapacitados visuales subrayan la importancia de su integración social, laboral y educativa a través de la rehabilitación, alfabetización en el sistema Braille y capacitación laboral para estas personas.

En este proceso, los profesores en educación especial e instructores del sistema escritura y lectura Braille juegan un papel importante, pues son ellos, quienes de forma deliberada y sistemática, preparan a los futuros usuarios del sistema Braille.

Existen actualmente aplicaciones de software orientadas a brindar ayuda a las personas en el proceso de aprendizaje del sistema Braille. En esta tesis se pretende desarrollar un software innovador que cumpla con esta misma función orientada a la transcripción y traducción de documentos en Braille al español y viceversa, implementando Memorias Asociativas Bidireccionales Alfa-Beta y análisis de imágenes; con lo cual se asegura una mayor confiabilidad en dicho proceso.

1.2. Planteamiento del Problema

El sistema Braille se ha convertido en una forma eficaz de escritura y lectura para personas invidentes desde su creación en 1825. Es un sistema de “celdas” las cuales se conforman por grupos

de seis relieves en forma de punto, los cuales pueden ser percibidos por medio del tacto. Este sistema es enseñado a personas con deficiencia visual así como a tutores, docentes o personas que deban interactuar con individuos con esta discapacidad. Hoy en día resulta muy práctico contar con un método que permita realizar la traducción de un lenguaje a otro para que personas, que no dominan dicho lenguaje puedan usarlo y entenderlo de manera sencilla. Es así como surge la idea de crear un software que relacione al sistema Braille y al idioma Español, siendo éste capaz de efectuar la conversión entre ambos, con el propósito de que se pueda comprender fácilmente el sistema Braille, mediante su interpretación usando símbolos del alfabeto latino, ya que su lectura táctil puede resultar compleja para personas que desconocen el Braille o bien para aquellas que comienzan a aprenderlo y practicarlo.

1.3. Objetivo General

Diseñar e implementar un Intérprete Braille-Español/Español-Braille utilizando el modelo de las Memorias Asociativas Bidireccionales Alfa-Beta.

1.3.1. Objetivos Particulares

- Elaborar y digitalizar un documento en sistema Braille.
- Procesar la imagen obtenida usando diversas técnicas del Procesamiento Digital de Imágenes (PDI).
- Implementar el modelo matemático de las Memoria Asociativas Bidireccionales (BAM) Alfa-Beta en un algoritmo utilizando el lenguaje de programación C#.
- Diseñar e implementar el Interprete Braille-Español/Español-Braille en base a los resultados obtenidos de los procesos de PDI y BAM Alfa-Beta.
- Obtener el porcentaje de recuperación del software en base a los resultados experimentales.

1.4. Justificación

El sentido de la vista es una de las funciones primordiales del ser humano. Según la Organización Mundial de la Salud (OMS) se estima que hay en el mundo 45 millones de personas ciegas y 135 millones que padecen alguna discapacidad visual. La agudeza visual se expresa como una fracción, el número superior de ésta se refiere a la distancia entre la persona y una tabla, la cual es

generalmente de 6 metros (20 pies); el número inferior indica la distancia a la que una persona con vista normal podría leer correctamente la línea con las letras más pequeñas. Por ejemplo, 20/20 se considera normal, 20/40 indica que la línea que el paciente leyó correctamente a los 20 pies pudo ser leída. De acuerdo con los resultados obtenidos en el XII Censo General de Población y Vivienda 2000, en México existían casi cinco personas con discapacidad visual por cada 1000 habitantes en el país, es decir, poco más de 467 mil personas, de las cuales 32.2% residían en el medio rural.

El estado con mayor prevalencia de discapacidad visual fue Yucatán, con 10.7 personas por cada mil habitantes, le siguen Tabasco y Campeche con 8.9 y 8.6, respectivamente. En contraste, Baja California (2.3), Tlaxcala (3.4) y México (3.4) fueron las entidades donde la prevalencia de discapacidad visual fue menor en el país. Ver Figura 1.4.1.

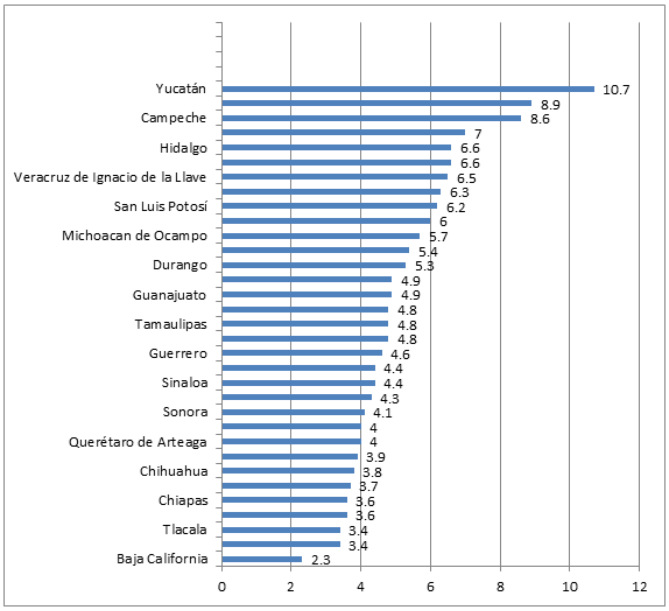


Figura 1.4.1. Prevalencia de discapacidad visual por entidad federativa.

A su vez, de las estadísiticas mostradas se tiene un censo en el que se calcula que aproximadamente el diez por ciento de las personas que se consideran ciegas o con deficiencia visual utilizan el sistema Braille y un porcentaje menor para aquellos que escriben empleando este sistema. Los principales factores que determinan estas estadísticas son el posible pobre desarrollo del sentido del tacto, además de la ausencia de textos más interesantes y a su pequeño número de ediciones.

CAPÍTULO 2

Antecedentes

2.1. El Sistema Braille

El sistema Braille es un sistema de lectura y escritura táctil, el cual presenta un conjunto de características específicas que hoy en día lo han hecho posicionarse como un método universal que utilizan las personas invidentes para poder comunicarse [1].

El sistema Braille se expresa mediante la utilización de puntos en relieve dispuestos ordenadamente en grupos de dos columnas con tres filas cada una, a los cuales se les llama celdas. Para facilitar la descripción de cada uno de los signos dispuestos en la celda, se han enumerado los puntos convencionalmente de la siguiente manera: los de la columna izquierda se numeran 1-2-3, de arriba abajo, y los puntos 4-5-6 corresponden a los de la derecha. La letra "A" se representa con el punto 1; "B" es la agrupación de los puntos 1-2, "C" los puntos 1-4, y así sucesivamente [2]. Las diez primeras letras están formadas por los cuatro puntos de arriba, las diez siguientes comprenden las diez primeras repetidas agregando el punto 3, hasta llegar a formar los diferentes grupos de signos (Ver Anexo 1).

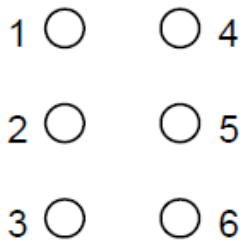


Figura 2.1.1 Configuración de una celda Braille.

En el Braille latino, el alfabeto abarca veintiséis de los signos, diez de los cuales sirven como signos de puntuación, mientras que los veintisiete restantes se usan diversamente, para satisfacer las

necesidades especiales de cada lengua determinada, o para las abreviaturas. Los números se hayan representados por las diez primeras letras precedidas de un signo numeral [1,2].

Para cierto número de lenguas se han establecido dos grados de Braille. En el primer grado, todas las palabras se escriben letra por letra, como en la escritura visual. El segundo grado corresponde a la forma cotidiana para fines más generales: revistas en Braille, libros y escritura de cartas. Comprende un número considerable de signos abreviados, para la expresión de preposiciones, conjunciones, pronombres, sufijos, repitiéndose con frecuencia grupos de letras y palabras. Su propósito consiste en reducir el volumen de los libros en Braille y al mismo tiempo ahorra al Braillista algunos esfuerzos en la lectura y la escritura [3].

Cabe mencionar que algunas lenguas se han permitido sistemas aún más abreviados, los cuales suelen considerarse como Braille grado tres [1], en el cual los textos originales completos son apenas irreconocibles, constituyendo casi una especie de taquigrafía, demasiado complicada para lectores que no reúnan las siguientes tres condiciones: amplio dominio del idioma, buena memoria y un sentido del tacto muy desarrollado.

El tacto es un factor sumamente importante para determinar la amplitud del uso del Braille. Los lectores se dividen, según sus capacidades, en grupos muy determinados, como aquellos que nacieron con discapacidad visual en donde el Braille viene a convertirse en algo totalmente natural para ellos, así como también a los que perdieron la vista en edad adulta, los cuales tienen que pasar de una lectura visual a una táctil. Estos últimos son generalmente lectores lentos, siendo los de más edad los que enfrentan mayores dificultades para dominar el Braille [1,3].

La lectura se realiza deslizando las yemas de los dedos de izquierda a derecha a lo largo de las líneas que constituyen el texto. La escritura puede realizarse de dos maneras, de forma manual, o utilizando una máquina de escribir llamada "Máquina de Perkins" [1].

La escritura manual se realiza introduciendo la hoja de papel en un bastidor cuyos lados verticales presentan unos orificios en los cuales se inserta la "pauta". Este instrumento consiste en una regleta en la que aparecen perforadas líneas con orificios rectangulares que se corresponden a las "celdas" (Figura 2.1.2). La persona escribe presionando con un punzón en las posiciones correspondientes de cada celda que identifican el signo que se pretende escribir. Este procedimiento exige que la

escritura se realice de derecha a izquierda, y que cada signo deba escribirse rotando 180 grados respecto a su posición cuando se lee. Es decir, los signos aparecen "en espejo" en la lectura respecto a su posición en la escritura [4].

Figura 2.1.2. Regleta y punzón para escritura manual en Braille.

La máquina de Perkins es una máquina de escribir que dispone de una tecla para cada uno de los seis puntos que constituyen cada "celda" y de otra que actúa como espaciador. Los puntos aparecen de abajo hacia arriba en el papel sobre el que se escribe, pero la persona para poder palparlos precisa desplazar su mano desde el teclado hacia la parte posterior del "carro" de la máquina. Cada celda se escribe presionando simultáneamente las teclas correspondientes al signo que se quiere escribir (Figura 2.1.3) [3,4].

Figura 2.1.3. Máquina de Perkins para escritura Braille.

El procedimiento de impresión del Braille resulta costoso, encargándose casi totalmente de su ejecución organizaciones privadas, a menudo con ayuda de subsidios del Gobierno. Los textos braille se estereotipan sobre planchas de un metal blando, a mano o por medio de máquinas. Estas planchas se colocan sobre una prensa plana o rotativa que repuja los puntos en un papel grueso y fuerte, generalmente humedecido, para facilitar la impresión de los puntos y evitar que se taladre el papel [1].

Por último, podemos expresar que el propósito principal del Braille consiste en ofrecer una forma de lectura y escritura que sirva a las personas ciegas para su educación, su trabajo, su correspondencia y goce literario. Más concretamente, pone en forma escrita completa toda clase de literatura de cualquier lengua, o, si se quiere, la expresa en una forma abreviada, representativa del texto completo. El objeto de la unificación consiste en ofrecer a las personas invidentes el medio más fácil de escritura para aprender, leer y comunicarse con otras lenguas además de la propia.

En México, como en otros países del mundo, se han implementado programas de integración e inclusión educativa con la finalidad de contribuir con el proceso educativo y social de las personas con discapacidad visual. En un principio, retomando el modelo europeo, se puso en marcha un programa educativo, el cual consistía básicamente en diagnosticar causas y síntomas de la enfermedad, más que en atender las necesidades educativas. Tiempo después llegó a México un nuevo modelo que planteaba que la educación de las personas con discapacidad no debía ser atendida únicamente desde el enfoque médico, sino también desde la orientación educativa (Programa Nacional de Fortalecimiento de la Educación Especial y de la Integración Educativa).

La inclusión educativa se refiere al proceso en el que las instituciones educativas, con la intención de impartir una educación más homogénea reconocen las necesidades de cada niño y se adaptan a ellas. Si bien este proceso no ha sido sencillo, lo cierto es que es un paso más hacia la integración socioeducativa para quienes viven con este tipo de discapacidad. Todos debemos estar conscientes de que este tipo de personas presentan ciertas necesidades educativas especiales, tales como el aprendizaje y el empleo del sistema Braille para el desarrollo de sus habilidades de lectura y escritura.

2.2. Memorias Asociativas

Una memoria asociativa es un proceso de interrelacionar información con otra, partiendo de conocimiento previo. Esta relación se puede ejemplificar de manera sencilla en la forma en que el ser humano puede identificar a una persona partiendo de sus facciones, tono de voz, estatura, entre otras características. Esto lo logra asociando un conocimiento adquirido previamente, al conocer por primera vez al otro individuo, con las características antes mencionadas.

En el área de la computación las memorias han adquirido un lugar importante, fundamentalmente en la teoría y aplicación de reconocimiento y clasificación de patrones.

Una memoria asociativa puede modelarse como un sistema de entrada y salida, donde la entrada corresponde a un estímulo o patrón representado por un vector x y un patrón de salida correspondiente a la respuesta del sistema representada por un vector y.

Las memorias asociativas poseen una fase de aprendizaje y una fase de recuperación.

a) Fase de aprendizaje: en esta fase se asocia a cada estimulo x^n una respuesta y^n.

$$\left.\begin{array}{l} x^1 \rightarrow y^1 \\ x^2 \rightarrow y^2 \\ \vdots \\ x^n \rightarrow y^n \end{array}\right\} \text{Memoria Asociativa}$$

b) Fase de recuperación: consiste en encontrar el vector y^n correspondiente a un patrón de entrada x^n mediante la utilización de la memoria asociativa, formada previamente.

$$\begin{array}{c} x^1 \rightarrow Memoria\ Asociativa \rightarrow y^1 \\ x^2 \rightarrow Memoria\ Asociativa \rightarrow y^2 \\ \vdots \\ x^n \rightarrow Memoria\ Asociativa \rightarrow y^n \end{array}$$

La importancia del uso de memorias asociativas recae en la implementación que puede tener en cuanto al reconocimiento de patrones, pues al tener una fase de aprendizaje y una fase de recuperación su uso es muy accesible. La confiabilidad que pueda tenerse al recuperar la información en este tipo de sistemas depende mucho del tipo de Memoria Asociativa que se implemente. A lo largo del tiempo las memorias asociativas han ido evolucionando, partiendo de modelos matemáticos como la morfología.

2.3. Estado del Arte

En este apartado están descritos algunos de los softwares más recientes que son similares al Interprete Braille-Español/Español-Braille que se propone en esta tesis, ya sea en cuanto a la

funcionalidad de traducir o bien a la serie de algoritmos que se ocupan. Se han hecho intentos para reconocer ópticamente Braille en relieve utilizando diversos métodos.

En 1988, Dubus y su equipo diseñaron un algoritmo llamado Lectobraille [5] en el que se traducen caracteres con relieves Braille en una versión impresa en papel. A partir de ello, se han realizado investigaciones basadas en el conocimiento de las técnicas de procesamiento de imágenes con el objetivo de traducir Braille a texto.

En 1993, Mennens, Tichelen, Francois y Engelen [6] diseñaron un sistema de reconocimiento óptico que reconoce la escritura Braille con el uso de un escáner especial disponible en el mercado. El resultado fue satisfactorio utilizando relieves en Braille razonablemente bien formados. Sin embargo, el sistema no puede manejar la deformación en la cuadrícula de alineación de puntos, es decir al haber una variante en la alineación de las filas o las columnas, el sistema comete considerables errores.

En 1999, los chinos Ng y Lau [29] abordaron el problema utilizando las técnicas de detección de contorno para traducir Braille en el idioma inglés o chino. Las tasas de reconocimiento eran buenas, sin embargo, no se hizo mención de la posible entrada de una rejilla deforme, ni su eficacia.

En 2001, Murray y Días [7] diseñaron un dispositivo de mano que se encarga de la exploración, así como la traducción. En este prototipo es el usuario quien tiene el control de la orientación de exploración, y por ser portátil sólo un pequeño segmento se escanea en cada caso, la deformación rejilla no es una preocupación importante, y un algoritmo más simple se utilizó para producir eficiente, traducción en tiempo real de caracteres Braille.

En 2003, en Italia, Morgavi y Morando [8] publicaron un artículo en el que describen el uso de un sistema híbrido utilizando una red neuronal para resolver el problema de reconocimiento. El documento también proporciona un medio de medición de precisión en el reconocimiento de Braille, y los resultados muestran que el sistema puede manejar un mayor grado de degradación de la imagen en comparación con los algoritmos que utilizan técnicas más convencionales y rígidas de procesamiento de imágenes.

En el año 2004, Wong, Abdulla y Hussmann [10], investigadores de la Escuela de Ingeniería de la Univeristy of Auckland en Nueva Zelanda propusieron un prototipo de solución de software para reconocer ópticamente el relieve de una cara de documentos en Braille con un sencillo algoritmo de procesamiento de imágenes utilizando una red neuronal probabilística. Teniendo como resultado un archivo de texto con formato Braille para conservar el diseño del documento original que se puede enviar posteriormente a una impresora electrónica para su reproducción.

Los experimentos preliminares se realizaron con una tasa de reconocimiento excelente, donde la precisión de transcripción es al 99%. Sin embargo el resultado final para este prototipo es únicamente la digitalización de un documento Braille, es decir, se limita a la creación de un documento digital que posteriormente puede ser re-impreso, pero no hace una conversión de cada casilla en una letra arábiga. A pesar de utilizar redes neuronales, su función es para interrelacionar una plantilla, o imagen de una columna y tres filas con puntos en diferente posición, con la imagen de entrada.

En la actualidad, hay disponibles en internet una serie de traductores online que permiten hacer la conversión de español a Braille como los siguientes:

- *Software Traductor a Braille Duxbury*

  Sistemas de Duxbury es líder mundial en software para Braille. El Duxbury Braille Translator (DBT) y MegaDots, son utilizados por prácticamente todos los principales editores del mundo en Braille. Nadie soporta más idiomas que Duxbury Systems - DBT apoya grado 1 y grado 2 de traducción en Inglés, Español, Francés, Alemán, Portugués, Árabe, Malayo, Sueco y otros idiomas. Existe una versión del programa para Windows, de documentos hechos en Word o Código ASCII a Código Braille sin la intervención del usuario. Automáticamente agrega todos los caracteres de control necesarios, así como las convenciones utilizadas en el Braille [11].

- *Alfabeto Braille en Línea*

  Es una aplicación desarrollada usando tecnología JAVA que permite a los usuarios escribir el texto deseado y convertirlo a braille, no usa un algoritmo muy complejo ya solamente se basa en hacer comparaciones [12]. La interfaz gráfica se muestra en la Figura 2.3.1.

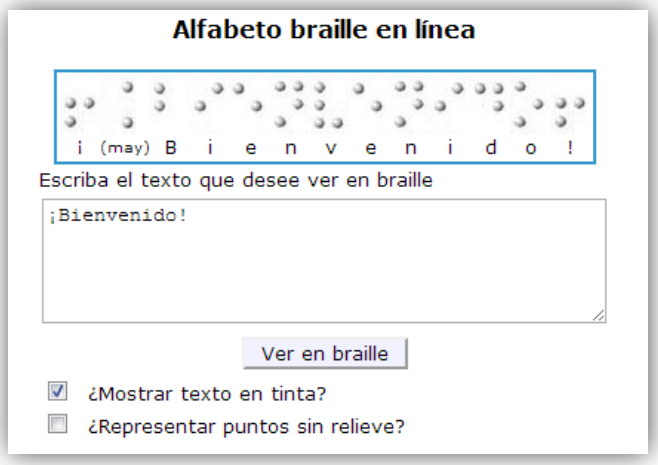


Figura 2.3.1 Interfaz principal del software "Alfabeto Braille en línea".

Poniendo en funcionamiento la aplicación, se introduce en la caja de texto la palabra "HOLA" y se selecciona cada una de las opciones que tiene. Los resultados se muestran en la Figura 2.3.2.

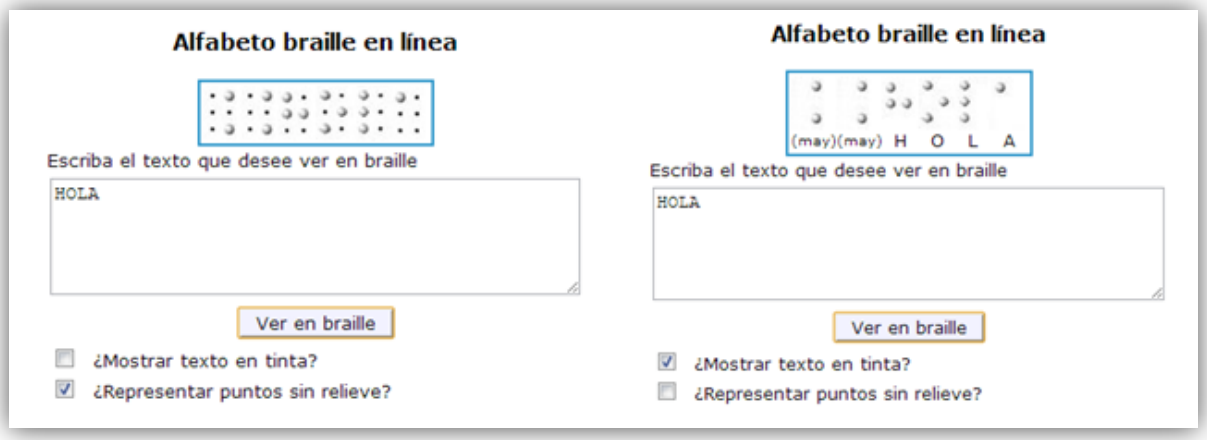


Figura 2.3.2 Funcionamiento del software "Alfabeto Braille en línea".

- *Braille Translator*

  Es una aplicación que permite a los usuarios escribir el texto deseado y convertirlo a Braille directamente o bien cargar un texto contenido en un archivo en formato PDF, DOC, DOCX, TXT, RTF o HTML [13]. La página principal se muestra en la Figura 2.3.3.

Figura 2.3.3 Vista principal del "BrailleTranslator".

También en esta aplicación, el usuario es el que ingresa el texto, pero con ciertas mejoras. Se pueden cargar archivos que contengan el texto en español, estos archivos pueden tener formato PDF, DOC, DOCX, TXT, RTF o HTML. Ver Figura 2.3.4.

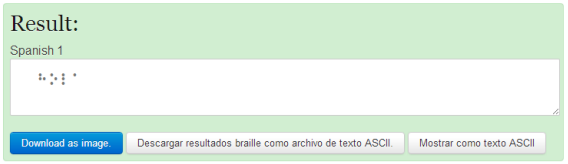


Figura 2.3.4 Resultados obtenidos del "BrailleTranslator".

- *Traductor Braille Simple*

  Es una aplicación que permite a los usuarios escribir el texto deseado y convertirlo a braille, posee una interfaz gráfica muy simple [14]. Ver Figura 2.3.5.

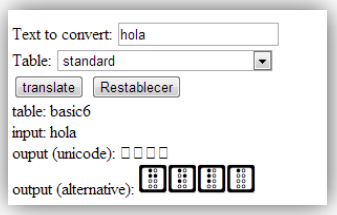


Figura 2.3.5 Interfaz principal del "Traductor Braille".

A pesar de que estos softwares son de los más populares y que se encuentran disponibles para los usuarios de Internet, se puede observar que solamente se ocupan de traducir texto en español a Braille, sin la necesidad de ocupar ningún algoritmo especial para hacer la conversión entre sistemas más que comparaciones simples. También se han recopilado algunos artículos que presentan algunos proyectos relacionados con la interpretación del Braille al español y se describen a continuación.

En 2007, Raoul Parienti de origen francés inventó el "Top Braille" [15], un aparato que permite traducir instantáneamente en Braille los textos impresos. El aparato es un poco más grande que el ratón de un ordenador, pesa 120 gramos, cabe en la palma de la mano o en el fondo de un bolsillo. El aparato se desplaza por una línea de texto, una microcámara digital escanea cada carácter, transmite las imágenes a un procesador que dirige una célula Braille bajo el índice del usuario. Los pequeños picos de esta célula bajan o suben para componer una traducción instantánea a Braille de cada letra, ya esté impresa sobre papel u otro soporte. Asimismo, el texto puede escucharse por medio de un auricular. Ver Figura 2.3.6.

Figura 2.3.6 El ciego Claude Garrandes utiliza el nuevo invento.

En 2010, el Consorcio Fernando de los Ríos una empresa pública del Gobierno de Andalucía ha desarrollado "Dots" [16], un traductor de Braille para GNOME que permite "traducir" un documento (ODT, PDF o HTML) en una representación Braille en el ordenador de forma que podamos enviar ese código Braille a una impresora Braille. Ver Figura 2.3.7.

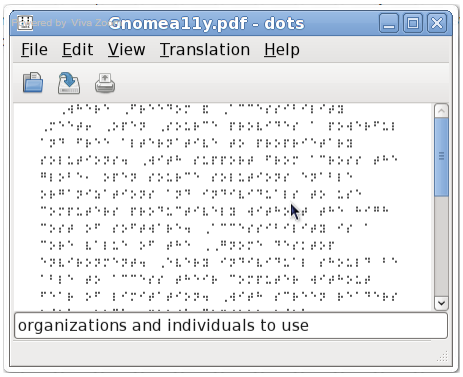


Figura 2.3.7 Dots: un traductor de Braille para GNOME.

CAPÍTULO 3

Marco Teórico

3.1. Conceptos Básicos

El propósito fundamental de una memoria asociativa es recuperar correctamente patrones completos a partir de patrones de entrada, los cuales pueden estar alterados con ruido aditivo, sustractivo o combinado.

Una Memoria Asociativa [9,17, 18] puede formularse como un sistema de entrada y salida, idea que se esquematiza a continuación:

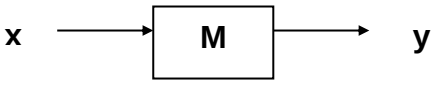


Figura 3.1.1. Memoria asociativa modelada como un sistema.

En este esquema, los patrones de entrada y salida están representados por vectores columna denotados por **x** y **y**, respectivamente. Cada uno de los patrones de entrada forma una asociación con el correspondiente patrón de salida, la cual es similar a la una pareja ordenada; por ejemplo, los patrones **x** y **y** del esquema anterior forman la asociación (**x**,**y**).

No obstante que a lo largo de las dos secciones restantes del presente capítulo se respetarán las notaciones originales de los autores de los modelos presentados aquí, a continuación se propone una notación que se usará en la descripción de los conceptos básicos sobre memorias asociativas, y en el resto de los capítulos de esta tesis.

Los patrones de entrada y salida se denotarán con las letras negrillas, **x** y **y**, agregándoles números naturales como superíndices para efectos de discriminación simbólica. Por ejemplo, a un patrón de entrada $\mathbf{x}^1$ le corresponderá el patrón de salida $\mathbf{y}^1$, y ambos formarán la asociación $(\mathbf{x}^1,\mathbf{y}^1)$; del mismo modo, para un número entero positivo k específico, la asociación correspondiente será $(\mathbf{x}^k,\mathbf{y}^k)$.

La memoria asociativa M se representa mediante una matriz, la cual se genera a partir de un conjunto finito de asociaciones conocidas de antemano: este es el conjunto fundamental de aprendizaje, o simplemente conjunto fundamental.

El conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$$

donde p es un número entero positivo que representa la cardinalidad del conjunto fundamental.

A los patrones que conforman las asociaciones del conjunto fundamental se les llama patrones fundamentales. La naturaleza del conjunto fundamental proporciona un importante criterio para clasificar las memorias asociativas:

Una memoria es Autoasociativa si se cumple que $x^{\mu} = y^{\mu} \ \forall \mu \in \{1, 2, ..., p\}$, por lo que uno de los requisitos que se debe de cumplir es que $n = m$.

Una memoria Heteroasociativa es aquella en donde $\exists \mu \in \{1, 2, ..., p\}$ para el que se cumple que $x^{\mu} \neq y^{\mu}$. Nótese que puede haber memorias heteroasociativas con $n = m$.

En los problemas donde intervienen las memorias asociativas, se consideran dos fases importantes:

La fase de aprendizaje, que es donde se genera la memoria asociativa a partir de las p asociaciones del conjunto fundamental, y la fase de recuperación que es donde la memoria asociativa opera sobre un patrón de entrada, a la manera del esquema que aparece al inicio de esta sección.
A fin de especificar las componentes de los patrones, se requiere la notación para dos conjuntos a los que llamaremos arbitrariamente A y B. Las componentes de los vectores columna que representan a los patrones, tanto de entrada como de salida, serán elementos del conjunto A, y las entradas de la matriz $\mathbf{M}$ serán elementos del conjunto B.

No hay requisitos previos ni limitaciones respecto de la elección de estos dos conjuntos, por lo que no necesariamente deben ser diferentes o poseer características especiales. Esto significa que el número de posibilidades para escoger *A* y *B* es infinito.

Por convención, cada vector columna que representa a un patrón de entrada tendrá *n* componentes cuyos valores pertenecen al conjunto *A*, y cada vector columna que representa a un patrón de salida tendrá *m* componentes cuyos valores pertenecen también al conjunto *A*. Es decir:

$$\mathbf{x}^{\mu} \in A^n \text{ y } \mathbf{y}^{\mu} \in A^m \ \forall \mu \in \{1, 2, ..., p\}$$

La *j*-ésima componente de un vector columna se indicará con la misma letra del vector, pero sin negrilla, colocando a *j* como subíndice ($j \in \{1, 2, ..., n\}$ o $j \in \{1, 2, ..., m\}$ según corresponda). La *j*-ésima componente del vector columna $\mathbf{x}^{\mu}$ se representa por: x_j^{μ}

Con los conceptos básicos ya descritos y con la notación anterior, es posible expresar las dos fases de una memoria asociativa:

1. **Fase de Aprendizaje** (Generación de la memoria asociativa). Encontrar los operadores adecuados y una manera de generar una matriz **M** que almacene las *p* asociaciones del conjunto fundamental $\{(\mathbf{x}^1,\mathbf{y}^1), (\mathbf{x}^2,\mathbf{y}^2), ..., (\mathbf{x}^p,\mathbf{y}^p)\}$, donde $\mathbf{x}^{\mu} \in A^n$ y $\mathbf{y}^{\mu} \in A^m \ \forall \mu \in \{1, 2, ..., p\}$. Si $\exists \mu \in \{1, 2, ..., p\}$ tal que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$, la memoria será heteroasociativa; si $m = n$ y $\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \ \forall \mu \in \{1, 2, ..., p\}$, la memoria será autoasociativa.

2. **Fase de Recuperación** (Operación de la memoria asociativa). Hallar los operadores adecuados y las condiciones suficientes para obtener el patrón fundamental de salida $\mathbf{y}^{\mu}$, cuando se opera la memoria **M** con el patrón fundamental de entrada $\mathbf{x}^{\mu}$; lo anterior para todos los elementos del conjunto fundamental y para ambos modos: autoasociativo y heteroasociativo.

Se dice que una memoria asociativa **M** exhibe **recuperación correcta** si al presentarle como entrada, en la fase de recuperación, un patrón $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$, ésta responde con el correspondiente patrón fundamental de salida $\mathbf{y}^{\omega}$.

Una memoria asociativa bidireccional también es un sistema de entrada y salida, solamente que el proceso es bidireccional. La dirección hacia adelante se describe de la misma forma que una memoria asociativa común: al presentarle una entrada **x**, el sistema entrega una salida **y**. La dirección hacia atrás se lleva a cabo presentándole al sistema una entrada **y** para recibir una salida **x**.

3.2. Memorias Asociativas

A continuación, en esta sección haremos un breve recorrido por los modelos de memorias asociativas, con objeto de establecer el marco de referencia en el que surgieron las memorias asociativas bidireccionales. Las memorias asociativas que se presentarán en esta sección, son los modelos más representativos que sirvieron de base para la creación de modelos matemáticos que sustentan el diseño y operación de memorias asociativas más complejas. Para cada modelo se describe su fase de aprendizaje y su fase de recuperación.

Se incluyen cuatro modelos clásicos basados en el anillo de los números racionales con las operaciones de multiplicación y adición: *Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Hopfield, además de tres modelos basados en paradigmas diferentes a la suma de productos, a saber: memorias asociativas Morfológicas, memorias asociativas Alfa-Beta y memorias asociativas Media.

3.2.1 *Lernmatrix* de Steinbuch

Karl Steinbuch fue uno de los primeros investigadores en desarrollar un método para codificar información en arreglos cuadriculados conocidos como *crossbar*. La importancia de la *Lernmatrix* se evidencia en una afirmación que hace Kohonen en su artículo de 1972, donde apunta que las matrices de correlación, base fundamental de su innovador trabajo, vinieron a sustituir a la *Lernmatrix* de Steinbuch.

La *Lernmatrix* [18] es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $\mathbf{x}^{\mu} \in A^{n}, \quad A = \{0,1\}$ y produce como salida la clase $\mathbf{y}^{\mu} \in A^{p}$ que le corresponde (de entre *p* clases diferentes), codificada ésta con un

método que en la literatura se le ha llamado *one-hot*. El método funciona así: para representar la clase $k \in \{1, 2, ..., p\}$, se asignan a las componentes del vector de salida $\mathbf{y}^{\mu}$ los siguientes valores: $y_k^{\mu} = 1$, y $y_j^{\mu} = 0$ para $j = 1, 2, ..., k\text{-}1, k\text{+}1, ..., p$.

Algoritmo de la *Lernmatrix*

Fase de Aprendizaje

Se genera el esquema (*crossbar*) al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \in A^n \times A^p$. Cada uno de los componentes m_{ij} de $\mathbf{M}$, la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon & \text{si } x_j^{\mu} = 1 = y_i^{\mu} \\ -\varepsilon & \text{si } x_j^{\mu} = 0 \text{ y } y_i^{\mu} = 1 \\ 0 & \text{en otro caso} \end{cases}$$

donde ε una constante positiva escogida previamente: es usual que ε es igual a 1.

Fase de Recuperación

La i-ésima coordenada y_i^{ω} del vector de clase $\mathbf{y}^{\omega} \in A^p$ se obtiene como lo indica la siguiente expresión, donde $\vee$ es el operador *máximo*:

$$y_i^{\omega} = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij} . x_j^{\omega} = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} m_{hj} . x_j^{\omega} \right] \\ 0 & \text{en otro caso} \end{cases}$$

3.2.2 *Correlograph* de Willshaw, Buneman y Longuet-Higgins

El *correlograph* [18] es un dispositivo óptico elemental capaz de funcionar como una memoria asociativa. En palabras de los autores “el sistema es tan simple, que podría ser construido en cualquier laboratorio escolar de física elemental”.

Algoritmo del *Correlograph*

Fase de Aprendizaje

La *red asociativa* se genera al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \in A^n \times A^m$. Cada uno de los componentes m_{ij} de la *red asociativa* **M** tiene valor cero al inicio, y se actualiza de acuerdo con la regla:

$$m_{ij} = \begin{cases} 1 & \text{si } y_i^{\mu} = 1 = x_j^{\mu} \\ valor\ anterior & \text{en otro caso} \end{cases}$$

Fase de Recuperación

Se le presenta a la *red asociativa* **M** un vector de entrada $\mathbf{x}^{\omega} \in A^n$. Se realiza el producto de la matriz **M** por el vector $\mathbf{x}^{\omega}$ y se ejecuta una operación de umbralizado, de acuerdo con la siguiente expresión:

$$y_i^{\omega} = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij}.x_j^{\omega} \geq u \\ 0 & \text{en otro caso} \end{cases}$$

donde *u* es el valor de umbral. Una estimación aproximada del valor de umbral *u* se puede lograr con la ayuda de un número indicador mencionado en el artículo de Willshaw *et al.* de 1969: $log_2 n$

3.2.3 *Linear Associator* de Anderson-Kohonen

El *Linear Associator* [18] tiene su origen en los trabajos pioneros de 1972 publicados por Anderson y Kohonen. Para presentar el *Linear Associator* consideremos de nuevo el conjunto fundamental:

$$\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\} \quad \text{con } A = \{0, 1\}, \quad \mathbf{x}^{\mu} \in A^n \quad \text{y} \quad \mathbf{y}^{\mu} \in A^m$$

Algoritmo del *Linear Associator*

Fase de Aprendizaje

1) Para cada una de las *p* asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se encuentra la matriz $\mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t$ de dimensiones $m \times n$
2) Se suman la *p* matrices para obtener la memoria

$$\mathbf{M} = \sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t = \left[m_{ij}\right]_{m \times n}$$

de manera que la *ij*-ésima componente de la memoria M se expresa así:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^{\mu} x_j^{\mu}$$

Fase de Recuperación

Esta fase consiste en presentarle a la memoria un patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., p\}$ y realizar la operación

$$\mathbf{M} \cdot \mathbf{x}^{\omega} = \left[\sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu} \right)^{t} \right] \cdot \mathbf{x}^{\omega}$$

3.2.4 La memoria asociativa Hopfield

El artículo de John J. Hopfield de 1982, publicado por la prestigiosa y respetada *National Academy of Sciences* (en sus *Proceedings*), impactó positivamente y trajo a la palestra internacional su famosa memoria asociativa.

En el modelo que originalmente propuso Hopfield [18], cada neurona x_i tiene dos posibles estados, a la manera de las neuronas de McCulloch-Pitts: $x_i = 0$ y $x_i = 1$; sin embargo, Hopfield observa que, para un nivel dado de exactitud en la recuperación de patrones, la capacidad de almacenamiento de información de la memoria se puede incrementar por un factor de 2, si se escogen como posibles estados de las neuronas los valores $x_i = -1$ y $x_i = 1$ en lugar de los valores originales $x_i = 0$ y $x_i = 1$.

Al utilizar el conjunto {-1, 1} y el valor de umbral cero, la fase de aprendizaje para la memoria Hopfield será similar, en cierta forma, a la fase de aprendizaje del *Linear Associator*. La intensidad de la fuerza de conexión de la neurona x_i a la neurona x_j se representa por el valor de m_{ij}, y se considera que hay simetría, es decir, $m_{ij} = m_{ji}$. Si x_i no está conectada con x_j entonces $m_{ij} = 0$; en particular, no hay conexiones recurrentes de una neurona a sí misma, lo cual significa que $m_{ij} = 0$. El estado instantáneo del sistema está completamente especificado por el vector columna de dimensión n cuyas coordenadas son los valores de las n neuronas.

La memoria Hopfield es autoasociativa, simétrica, con ceros en la diagonal principal. En virtud de que la memoria es autoasociativa, el conjunto fundamental para la memoria Hopfield es $\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ con $\mathbf{x}^{\mu} \in A^n$ y $A = \{-1, 1\}$

Algoritmo Hopfield

Fase de Aprendizaje

La fase de aprendizaje para la memoria Hopfield es similar a la fase de aprendizaje del *Linear Associator*, con una ligera diferencia relacionada con la diagonal principal en ceros, como se muestra en la siguiente regla para obtener la *ij*-ésima componente de la memoria Hopfield **M**:

$$m_{ij} = \begin{cases} \sum_{\mu=1}^{p} x_i^{\mu} x_j^{\mu} \text{ si } i \neq j \\ 0 \text{ si } i = j \end{cases}$$

Fase de Recuperación

Si se le presenta un patrón de entrada $\tilde{\mathbf{x}}$ a la memoria Hopfield, ésta cambiará su estado con el tiempo, de modo que cada neurona x_i ajuste su valor de acuerdo con el resultado que arroje la comparación de la cantidad $\sum_{j=1}^{n} m_{ij} x_j$ con un valor de umbral, el cual normalmente se coloca en cero. Se representa el estado de la memoria Hopfield en el tiempo t por $\mathbf{x}(t)$; entonces $x_i(t)$ representa el valor de la neurona x_i en el tiempo t y $x_i(t+1)$ el valor de x_i en el tiempo siguiente (t+1).

Dado un vector columna de entrada $\tilde{\mathbf{x}}$, la fase de recuperación consta de tres pasos:

1) Para $t = 0$, se hace $\mathbf{x}(t) = \tilde{\mathbf{x}}$; es decir, $x_i(0) = \tilde{x}_i$, $\forall i \in \{1,2,3,...,n\}$

2) $\forall i \in \{1,2,3,...,n\}$ se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

$$x_i(t+1) = \begin{cases} 1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) > 0 \\ x_i(t) \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) = 0 \\ -1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) < 0 \end{cases}$$

3) Se compara $x_i(t+1)$ con $x_i(t)$ $\forall i \in \{1, 2, 3,...,n\}$. Si $\mathbf{x}(t+1) = \mathbf{x}(t)$ el proceso termina y el vector recuperado es $\mathbf{x}(0) = \tilde{\mathbf{x}}$. De otro modo, el proceso continúa de la siguiente manera: los pasos 2 y 3 se iteran tantas veces como sea necesario hasta llegar a un valor $t = \tau$ para el cual $x_i(\tau+1) = x_i(\tau)$ $\forall i \in \{1, 2, 3,...,n\}$; el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

En el artículo original de 1982, Hopfield había estimado empíricamente que su memoria tenía una capacidad de recuperar $0.15n$ patrones, y en el trabajo de Abu-Mostafa & St. Jacques se estableció formalmente que una cota superior para el número de vectores de estado arbitrarios estables en una memoria Hopfield es n.

3.2.5 Memorias Asociativas Morfológicas

La diferencia fundamental entre las memorias asociativas clásicas (*Lernmatrix, Correlograph, Linear Associator* y Memoria Asociativa Hopfield) y las memorias asociativas morfológicas radica en los fundamentos operacionales de éstas últimas, que son las operaciones morfológicas de *dilatación* y *erosión*; el nombre de las memorias asociativas morfológicas está inspirado precisamente en estas dos operaciones básicas. Estas memorias rompieron con el esquema utilizado a través de los años en los modelos de memorias asociativas clásicas, que utilizan operaciones convencionales entre vectores y matrices para la fase de aprendizaje y suma de productos para la recuperación de patrones. Las memorias asociativas morfológicas cambian los productos por sumas y las sumas por máximos o mínimos en ambas fases, tanto de aprendizaje como de recuperación. Hay dos tipos de memorias asociativas morfológicas: las memorias *max*, simbolizadas con **M**, y las memorias *min*, cuyo símbolo es **W**; en cada uno de los dos tipos, las memorias pueden funcionar en ambos modos: heteroasociativo y autoasociativo.

Se definen dos nuevos productos matriciales:

El *producto máximo* entre **D** y **H**, denotado por $\mathbf{C} = \mathbf{D} \nabla \mathbf{H}$, es una matriz $[c_{ij}]_{mxn}$ cuya ij-ésima componente c_{ij} es

$$c_{ij} = \bigvee_{k=1}^{r} \left(d_{ik} + h_{kj} \right)$$

El *producto mínimo* de **D** y **H** denotado por $\mathbf{C} = \mathbf{D} \Delta \mathbf{H}$, es una matriz $[c_{ij}]_{mxn}$ cuya ij-ésima componente c_{ij} es:

$$c_{ij} = \bigwedge_{k=1}^{r} \left(d_{ik} + h_{kj} \right)$$

Los productos máximo y mínimo contienen a los operadores máximo y mínimo, los cuales están íntimamente ligados con los conceptos de las dos operaciones básicas de la morfología matemática: *dilatación* y *erosión*, respectivamente.

3.2.5.1 Memorias Heteroasociativas Morfológicas

Algoritmo de las memorias morfológicas *max*

Fase de Aprendizaje

1. Para cada una de las p asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se usa el producto mínimo para crear la matriz $\mathbf{y}^{\mu} \Delta (-\mathbf{x}^{\mu})^t$ de dimensiones m x n, donde el negado transpuesto del patrón de entrada $\mathbf{x}^{\mu}$ se define como $(-\mathbf{x}^{\mu})^t = \left(-x_1^{\mu}, -x_2^{\mu}, ..., x_n^{\mu}\right)$.
2. Se aplica el operador máximo $\vee$ a las p matrices para obtener la memoria **M**.

$$\mathbf{M} = \bigvee_{\mu=1}^{p} \left[\mathbf{y}^{\mu} \Delta (-\mathbf{x}^{\mu})^t\right]$$

Fase de Recuperación

Esta fase consiste en realizar el producto mínimo Δ de la memoria **M** con el patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., p\}$, para obtener un vector columna **y** de dimensión m:

$$\mathbf{y} = \mathbf{M} \Delta \mathbf{x}^{\omega}$$

Las fases de aprendizaje y de recuperación de las **memorias morfológicas *min*** se obtienen por dualidad.

3.2.5.2 Memorias Autoasociativas Morfológicas

Para este tipo de memorias se utilizan los mismos algoritmos descritos anteriormente y que son aplicados a las memorias heteroasociativas; lo único que cambia es el conjunto fundamental. Para este caso, se considera el siguiente conjunto fundamental:

$$\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mathbf{x}^{\mu} \in A^n, \text{ donde } \mu = 1, 2, ..., p\}$$

3.2.6 Memorias Asociativas Alfa-Beta

Las memorias Alfa-Beta [17, 18] utilizan máximos y mínimos, y dos operaciones binarias originales α y β de las cuales heredan el nombre.

Para la definición de las operaciones binarias α y β se deben especificar los conjuntos A y B, los cuales son:

$$A = \{0, 1\} \quad \text{y} \quad B = \{0, 1, 2\}$$

La operación binaria $\alpha: A \times A \rightarrow B$ se define en la Tabla 3.2.1.

Tabla 3.2.1 Operador Alfa.

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

La operación binaria $\beta: B \times A \rightarrow A$ se define como se muestra en la Tabla 3.2.2.

\Tabla 3.2.2 Operador Beta.

x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Los conjuntos *A* y *B*, las operaciones binarias α y β junto con los operadores $\wedge$ (mínimo) y $\vee$ (máximo) usuales conforman el sistema algebraico (A, B, α, β, $\wedge$, $\vee$) en el que están inmersas las memorias asociativas Alfa-Beta.

Las memorias asociativas Alfa-Beta tienen un conjunto fundamental denotado por $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2,..., p\}$ $\mathbf{x}^\mu \in A^n$ y $\mathbf{y}^\mu \in A^m$, con $A = \{0, 1\}$, $n \in \mathbf{Z}^+$, $p \in \mathbf{Z}^+$, $m \in \mathbf{Z}^+$ y $1 < p \leq \min(2^n, 2^m)$. Además, se cumple la condición de que todos los patrones de entrada sean diferentes; es decir $\mathbf{x}^\mu = \mathbf{x}^\xi$ si y sólo si $\mu = \xi$. Si $\forall \mu \in \{1, 2,..., p\}$ se cumple que $\mathbf{x}^\mu = \mathbf{y}^\mu$, la memoria Alfa-Beta será *autoasociativa*; y si la afirmación es falsa, es decir $\exists \mu \in \{1, 2,..., p\}$ para el que se cumple $\mathbf{x}^\mu \neq \mathbf{y}^\mu$, la memoria Alfa-Beta será *heteroasociativa*.

Antes de continuar con la descripción del proceso y el desarrollo de los fundamentos matemáticos, se definen tres tipos de vectores que se usarán intensivamente en el nuevo modelo y se definen, además, dos transformadas vectoriales originales, propias del nuevo modelo.

Vector One-Hot

Sea el conjunto $A = \{0, 1\}$ y sean $p \in \mathbf{Z}^+$, $p > 1$, $k \in \mathbf{Z}^+$, tales que $1 \leq k \leq p$. El k-ésimo vector one-hot de p bits se define como el vector $\mathbf{h}^k \in A^p$ para el cual se cumple que la k-ésima componente $h_k^k = 1$ y las demás componentes $h_j^k = 0$, $\forall j \neq k$, $1 \leq j \leq p$.

Nota : En esta definición se excluye el valor $p = 1$ *porque un vector one-hot de dimensión* 1, *por su esencia misma, no tiene razón de ser.*

Vector Zero-Hot

Sea el conjunto $A = \{0, 1\}$ y sean $p \in \mathbf{Z}^+$, $p > 1$, $k \in \mathbf{Z}^+$, tales que $1 \leq k \leq p$. El k-ésimo vector zero-hot de p bits se define como el vector $\bar{\mathbf{h}}^k \in A^p$ para el cual se cumple que la k-ésima componente $\bar{h}_k^k = 0$ y las demás componentes $\bar{h}_j^k = 1$, $\forall j \neq k$, $1 \leq j \leq p$.

Nota: *En esta definición se excluye el valor* $p = 1$ *porque un vector zero-hot de dimensión* 1, *por su esencia misma, no tiene razón de ser.*

Transformada vectorial de expansión dimensional

Sea el conjunto $A = \{0, 1\}$ y sean $n \in \mathbf{Z}^+$, y $m \in \mathbf{Z}^+$. Dados dos vectores cualesquiera $\mathbf{x} \in A^n$ y $\mathbf{e} \in$

A^m, se define la transformada vectorial de expansión de orden m, $\tau^e : A^n \rightarrow A^{n+m}$, como τ^e (**x**, **e**) = $\mathbf{X} \in A^{n+m}$, vector cuyas componentes son: $X_i = x_i$ para $1 \leq i \leq n$ y $X_i = e_i$ para $n + 1 \leq i \leq n + m$.

Transformada vectorial de contracción dimensional

Sea el conjunto A = {0, 1} y sean $n \in \mathbf{Z}^+, m \in Z^+$ tales que $1 \leq m < n$. Dado un vector cualesquiera $\mathbf{X} \in A^{n+m}$, se define la transformada vectorial de contracción de orden m, $\tau^c : A^{n+m} \rightarrow A^m$, como $\tau^c(\mathbf{X}, m) = \mathbf{c} \in A^m$, vector cuyas componentes son: $c_i = X_{i+n}$ para $1 \leq i < m$.

Vector negado

Sea el conjunto A = {0, 1} y sea un vector $\mathbf{s} \in A^n$, se define el vector negado de **s** como el vector $\bar{\mathbf{s}}$, tal que $\bar{s}_i = \neg s_i$, donde $\neg$ es el operador lógico de negación booleano.

Habiendo definido cinco conceptos importantes, se continúa con la descripción del funcionamiento de las memorias asociativas bidireccionales Alfa-Beta.

3.2.6.1 Funcionamiento del Algoritmo

El funcionamiento de los procesos requeridos por la BAM Alfa-Beta tanto en la Fase de Aprendizaje, como en la Fase de Recuperación en el sentido de $\mathbf{x} \rightarrow \mathbf{y}$, algoritmo para las Etapas 1 y 2, y en el sentido $\mathbf{y} \rightarrow \mathbf{x}$, algoritmo para las Etapas 3 y 4, se describen a continuación:

Algoritmo de las Etapas 1 y 2

El siguiente algoritmo describe los pasos requeridos por la memoria asociativa bidireccional Alfa-Beta para realizar la fase de aprendizaje y la fase de recuperación en el sentido de $x \rightarrow y$.

FASE DE APRENDIZAJE

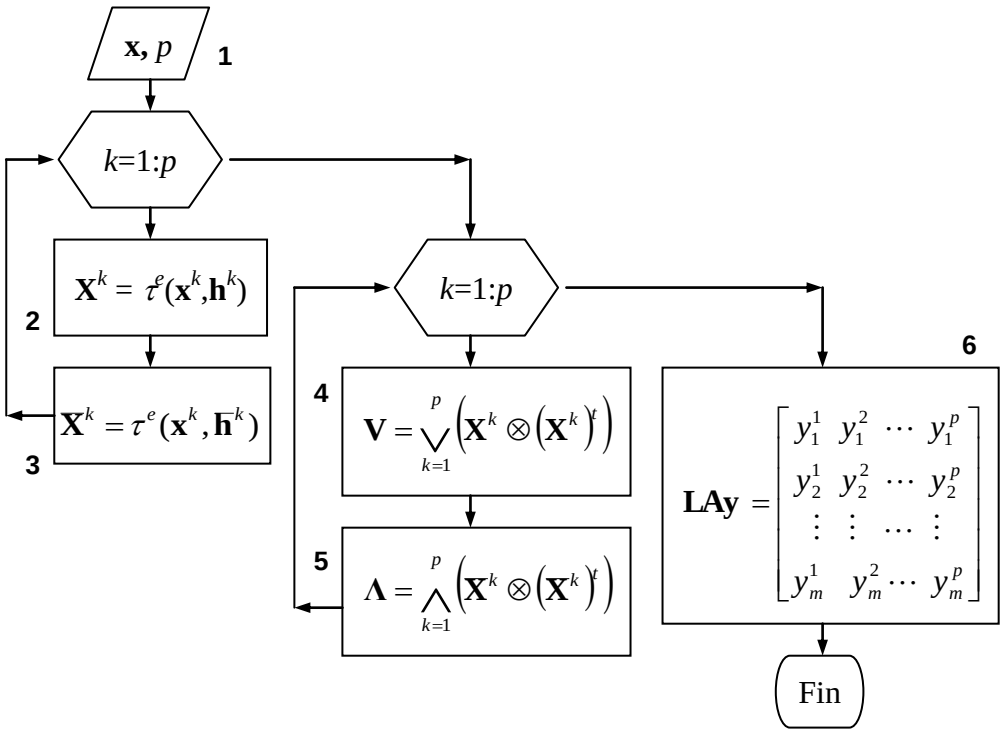


Figura 3.2.1 Algoritmo de aprendizaje de la BAM.

Paso 1. Como datos se tienen los *p* patrones de entrada **x**.

Paso 2. Se aplica la transformada de expansión vectorial del vector **x** para cada $\mathbf{X}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{x}^k$ y cada vector *one-hot* $\mathbf{h}^k$.

Paso 3. Se aplica la transformada de expansión vectorial del vector **x** para cada $\overline{\mathbf{X}}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{x}^k$ y cada vector *zero-hot* $\overline{\mathbf{h}}^k$.

Paso 4. Se crea una memoria asociativa Alfa-Beta *max* **V** con el conjunto fundamental

$$\left\{\left(\mathbf{X}^k, \mathbf{X}^k\right) \mid k = 1,..., p\right\}$$

Paso 5. Se crea una memoria asociativa Alfa-Beta *min* $\boldsymbol{\Lambda}$ con el conjunto fundamental

$$\left\{\left(\overline{\mathbf{X}}^k, \overline{\mathbf{X}}^k\right) \mid k = 1,..., p\right\}$$

Paso 6. Se crea una matriz **LAy**, que consiste de un *Linear Associator* modificado utilizando los patrones de salida **y**.

FASE DE RECUPERACIÓN

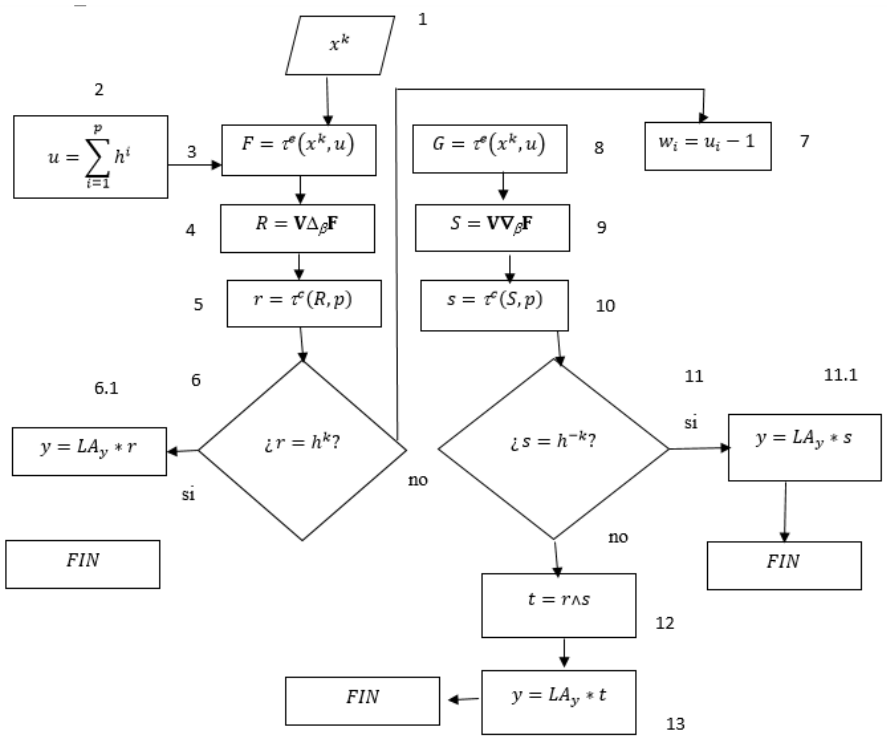


Figura 3.2.2 Algoritmo de recuperación de la BAM.

Paso 1. Presentar, a la entrada de la etapa 1, un vector del conjunto fundamental $\mathbf{x}^k \in A^n$ para algún índice $k \in \{1, ..., p\}$

Paso 2. Construir el vector $\mathbf{u} \in A^p$

Paso 3. Aplicar la transformada vectorial de expansión del vector **x** utilizando como argumentos el vector $\mathbf{x}^k$ y el vector **u**, para obtener **F**.

Paso 4. Operar la memoria autoasociativa Alfa-Beta *max* **V** con **F**, para obtener un vector **R.**

Paso 5. Aplicar la transformada de contracción del vector **R**, cuyos argumentos son el vector **R**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **r**.

Paso 6. Si r es un vector *one-hot*, entonces **r** es el *k-ésimo* vector *one-hot*, $\mathbf{h}^k$, **entonces**.

Paso 6.1 Se realiza la operación **LAy** · **r**, lo que resultará en el $\mathbf{y}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 7. Construir el vector $\mathbf{w} \in A^p$

Paso 8. Aplicar la transformada vectorial de expansión del vector **x** utilizando como argumentos el vector $\mathbf{x}^k$ y el vector **w**, para obtener **G**.

Paso 9. Operar la memoria autoasociativa Alfa-Beta *min* $\boldsymbol{\Lambda}$ con **G**, para obtener un vector **S.**

Paso 10. Aplicar la transformada de contracción del vector **S**, cuyos argumentos son el vector **S**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **s**.

Paso 11. Si s es un vector *zero-hot*, entonces **s** es el *k*-ésimo vector *zero-hot*, $\bar{\mathbf{h}}^k$, **entonces**.

Paso 11.1 Se realiza la operación $\mathbf{LAy} \cdot \bar{\mathbf{s}}$, lo que resultará en el $\mathbf{y}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 12. Realizar la operación AND lógica entre r y $\bar{\mathbf{s}}$ para obtener el vector **t**, el cual será igual al *k*-ésimo vector *one-hot*.

Paso 13. Realizar la operación **LAy · t**, para obtener el $\mathbf{y}^k$ correspondiente. **Fin**.

Algoritmo de las Etapas 3 y 4

El siguiente algoritmo describe los pasos requeridos por la memoria asociativa bidireccional Alfa-Beta para realizar la fase de aprendizaje y la fase de recuperación en el sentido de $\mathbf{y} \rightarrow \mathbf{x}$.

FASE DE APRENDIZAJE

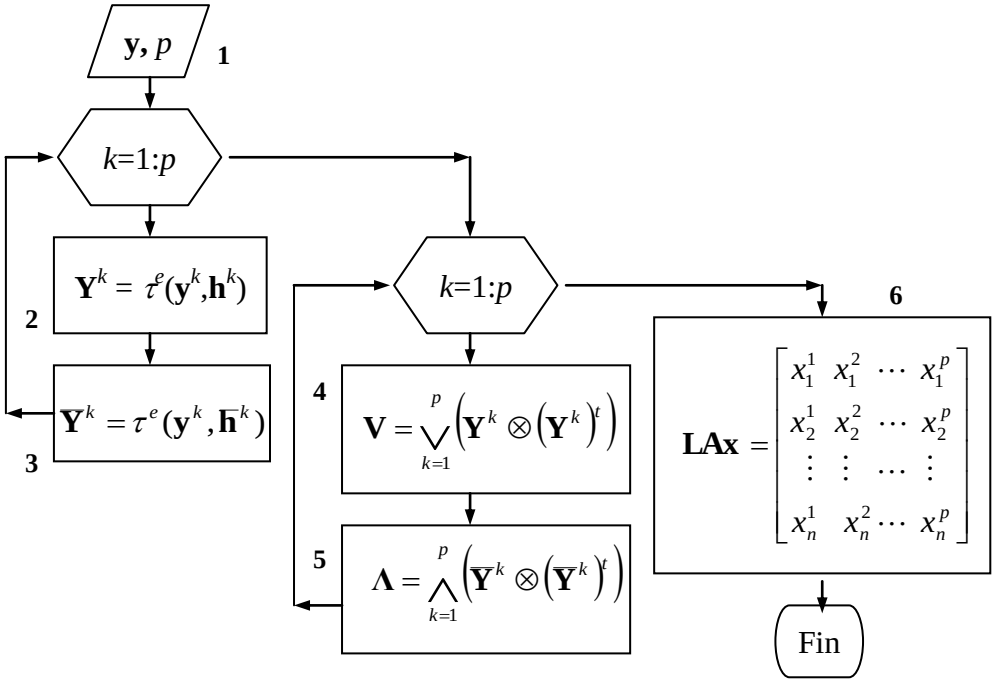


Figura 3.2.3 Algoritmo de aprendizaje de la BAM en sentido inverso.

Paso 1. Como datos se tienen los *p* patrones de salida **y**.

Paso 2. Se aplica la transformada de expansión vectorial del vector **Y** para cada $\mathbf{Y}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{y}^k$ y cada vector *one-hot* $\mathbf{h}^k$.

Paso 3. Se aplica la transformada de expansión vectorial del vector $\overline{\mathbf{Y}}$ para cada $\overline{\mathbf{Y}}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{y}^k$ y cada vector *zero-hot* $\overline{\mathbf{h}}^k$.

Paso 4. Se crea una memoria asociativa Alfa-Beta *max* **V** con el conjunto fundamental

$$\left\{\left(\mathbf{Y}^k, \mathbf{Y}^k\right) \mid k = 1,..., p\right\}$$

Paso 5. Se crea una memoria asociativa Alfa-Beta *min* $\boldsymbol{\Lambda}$ con el conjunto fundamental

$$\left\{\left(\overline{\mathbf{Y}}^k, \overline{\mathbf{Y}}^k\right) \mid k = 1,..., p\right\}$$

Paso 6. Se crea una matriz **LAx**, que consiste de un *Linear Associator* modificado utilizando los patrones de entrada **x**.

FASE DE RECUPERACIÓN

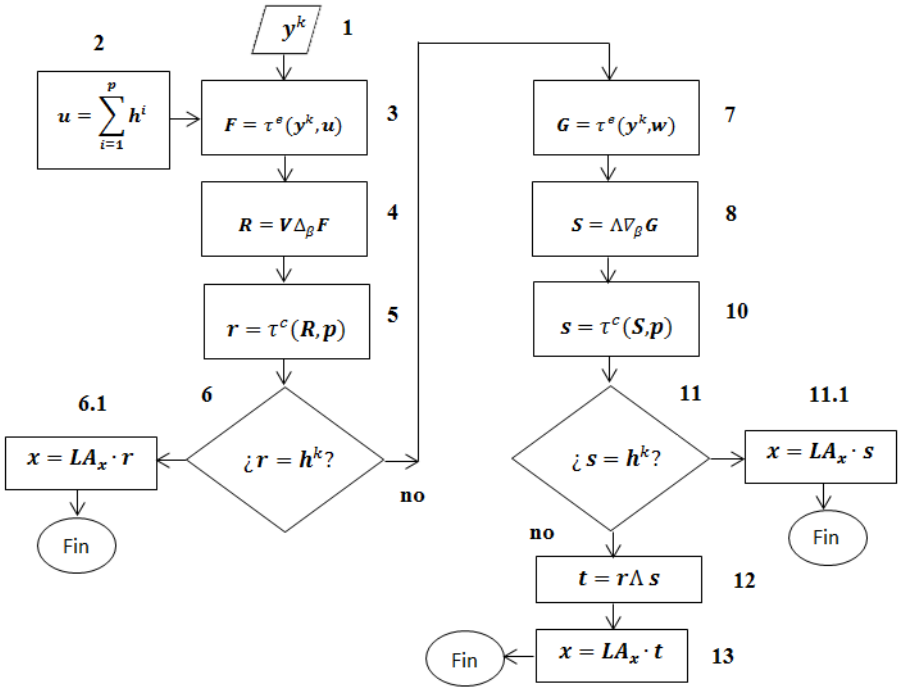


Figura 3.2.4 Algoritmo de recuperación de la BAM en sentido inverso.

Paso 1. Presentar, a la entrada de la etapa 1, un vector del conjunto fundamental $\mathbf{y}^k \in A^m$ para algún índice $k \in \{1, ..., p\}$

Paso 2. Construir el vector $\mathbf{u} \in A^p$

Paso 3. Aplicar la transformada vectorial de expansión del vector **y** utilizando como argumentos el vector $\mathbf{y}^k$ y el vector **u**, para obtener **F**.

Paso 4. Operar la memoria autoasociativa Alfa-Beta *max* **V** con **F**, para obtener un vector **R.**

Paso 5. Aplicar la transformada de contracción del vector **R**, cuyos argumentos son el vector **R**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **r**.

Paso 6. Si r es un vector *one-hot*, entonces **r** es el *k-ésimo* vector *one-hot*, $\mathbf{h}^k$, **entonces**.

Paso 6.1 Se realiza la operación $\mathbf{LAx} \cdot \mathbf{r}$, lo que resultará en el $\mathbf{x}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 7. Construir el vector $\mathbf{w} \in A^p$

Paso 8. Aplicar la transformada vectorial de expansión del vector **y** utilizando como argumentos el vector $\mathbf{y}^k$ y el vector **w**, para obtener **G**.

Paso 9. Operar la memoria autoasociativa Alfa-Beta *min* Λ con **G**, para obtener un vector **S.**

Paso 10. Aplicar la transformada de contracción del vector **S**, cuyos argumentos son el vector **S**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **s**.

Paso 11. Si s es un vector *zero-hot*, entonces **s** es el *k*-ésimo vector *zero-hot*, $\overline{\mathbf{h}}^k$, **entonces**.

Paso 11.1 Se realiza la operación $\mathbf{LAx} \cdot \overline{\mathbf{s}}$, lo que resultará en el $\mathbf{x}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 12. Realizar la operación AND lógica entre r y $\overline{\mathbf{s}}$ para obtener el vector **t**, el cual será igual al *k*-ésimo vector *one-hot*.

Paso 13. Realizar la operación $\mathbf{LAx} \cdot \mathbf{t}$, para obtener el $\mathbf{x}^k$ correspondiente. **Fin**.

Ejemplo ilustrativo de la Memoria Asociativa Bidireccional Alfa-Beta

Sean los siguientes 4 pares de patrones (*p*=4), los patrones de entrada, **x** con dimensión 4 (*n*=4) y los patrones de salida, **y** con dimensión 3 (*m*=3).

$$\mathbf{x}^1=\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ \mathbf{y}^1=\begin{bmatrix}1\\0\\1\end{bmatrix},\ \mathbf{x}^2=\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\ \mathbf{y}^2=\begin{bmatrix}1\\0\\0\end{bmatrix},\ \mathbf{x}^3=\begin{bmatrix}0\\0\\1\\0\end{bmatrix},\ \mathbf{y}^3=\begin{bmatrix}0\\1\\0\end{bmatrix},\ \mathbf{x}^4=\begin{bmatrix}1\\1\\0\\0\end{bmatrix},\ \mathbf{y}^4=\begin{bmatrix}1\\1\\0\end{bmatrix}$$

FASE DE APRENDIZAJE

- Se aplica la transformada vectorial de expansión del vector **x**, para obtener cada uno de los vectores $\mathbf{X}^k$ y $\overline{\mathbf{X}}^k$

$$\boxed{\mathbf{X}=\tau^e(\mathbf{x},\mathbf{h})}$$

$$\mathbf{X}^1=\begin{bmatrix}1\\1\\1\\1\\1\\0\\0\\0\end{bmatrix}\ \mathbf{X}^2=\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}\ \mathbf{X}^3=\begin{bmatrix}0\\0\\1\\0\\0\\0\\1\\0\end{bmatrix}\ \mathbf{X}^4=\begin{bmatrix}1\\1\\0\\0\\0\\0\\0\\1\end{bmatrix}$$

Vectores *one-hot*

$$\boxed{\overline{\mathbf{X}}=\tau^e(\mathbf{x},\overline{\mathbf{h}})}$$

$$\overline{\mathbf{X}}^1=\begin{bmatrix}1\\1\\1\\1\\0\\1\\1\\1\end{bmatrix}\ \overline{\mathbf{X}}^2=\begin{bmatrix}0\\0\\0\\0\\1\\0\\1\\1\end{bmatrix}\ \overline{\mathbf{X}}^3=\begin{bmatrix}0\\0\\1\\0\\1\\1\\0\\1\end{bmatrix}\ \overline{\mathbf{X}}^4=\begin{bmatrix}1\\1\\0\\0\\1\\1\\1\\0\end{bmatrix}$$

Vectores *zero-hot*

- Se generan las memorias autoasociativas Alfa-Beta *max* $\mathbf{V}=\bigvee_{k=1}^{4}\left(\mathbf{X}^k\otimes\left(\mathbf{X}^k\right)^t\right)$ y *min* $\mathbf{\Lambda}=\bigwedge_{k=1}^{4}\left(\mathbf{X}^k\otimes\left(\mathbf{X}^k\right)^t\right)$

$$\text{MAX}\quad \mathbf{V} = \begin{bmatrix} 1&1&2&2&2&2&2&1 \\ 1&1&2&2&2&2&2&2 \\ 2&2&1&2&2&2&2&2 \\ 1&1&1&1&1&2&2&2 \\ 1&1&1&1&1&2&2&2 \\ 2&2&2&2&2&1&2&2 \\ 2&2&1&2&2&2&1&2 \\ 1&1&2&2&2&2&2&1 \end{bmatrix} \quad \text{y} \quad \mathbf{\Lambda} = \begin{bmatrix} 1&1&0&1&0&0&0&0 \\ 1&1&0&1&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 1&1&1&1&0&1&0&0 \\ 1&1&0&1&0&0&1&0 \\ 0&0&1&1&0&0&0&1 \end{bmatrix} \quad \text{MIN}$$

- Se crea el *Linear Associator* Modificado utilizando los patrones de salida **y**

$$\mathbf{LAy} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

FASE DE RECUPERACIÓN

- Se presenta a la entrada de la BAM Alfa-Beta el patrón $\mathbf{x}^3$, se construye el vector **u** y se construye la expansión

$$\mathbf{x}^3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{u} = \sum_{i=1}^{4} \mathbf{h}^i = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \rightarrow \boxed{\mathbf{F} = \tau^e(\mathbf{x}^3, \mathbf{u})} \qquad \mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Se opera la memoria autoasociativa Alfa-Beta *max* **V** con **F**

$$\mathbf{R} = \mathbf{V}\Delta_{\beta}\mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- Se realiza la contracción: $\mathbf{r} = \tau^{c}(\mathbf{R},4) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Debido a que **r** no es un vector *one-hot*, entonces se continúa con el algoritmo

- Se construye el vector **u** y con el patrón $\mathbf{x}^3$, se construye la expansión

$$\mathbf{x}^3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \longrightarrow \boxed{\mathbf{G} = \tau^{e}(\mathbf{x}^3,} \longrightarrow \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Se opera la memoria autoasociativa Alfa-Beta *min* $\mathbf{\Lambda}$ con **G**

$$\mathbf{S} = \mathbf{\Lambda}\nabla_{\beta}\mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

- Se realiza la contracción $\mathbf{s} = \tau^{c}(\mathbf{S},4) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

El vector **s** no es un vector *zero-hot*, por lo que se continúa con el proceso.

- Se realiza la operación AND entre el vector **r** y el vector negado de **s**, $\bar{s}$, para obtener el vector **t**

$$\mathbf{t} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \Lambda \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Con esta operación se obtiene el tercer vector *one-hot*

- Se obtiene el vector $\mathbf{y}^3$ mediante

$$\mathbf{y}_{rec} = \mathbf{LAy} \cdot \mathbf{t} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \mathbf{y}^3$$

CAPÍTULO 4

Fundamentos para el Procesamiento Digital de Imágenes

4.1. Introducción

El procesamiento digital de imágenes se refiere al proceso de usar algoritmos para editar y manipular imágenes digitales. Una imagen digital es una colección finita de pequeños elementos discretos llamados pixeles, estos pixeles están organizados en una matriz bidireccional y representan la más pequeña cantidad de información gráfica disponible, esto se evidencia cuando se mira muy de cerca a una imagen, los pixeles aparecen como pequeños puntos, cabe mencionar también que a mayor número de pixeles en una imagen mayor detalle o resolución, por ejemplo, en el caso de las cámaras digitales, estas a menudo son clasificadas de acuerdo a la resolución que ofrecen, y la unidad de medida para indicarlo son los megapíxeles. Un megapíxel significa que una imagen está conformada por un millón de pixeles, entonces, una cámara de 8 megapíxeles es capaz de tomar una imagen de hasta 8 millones de pixeles [22].

El procesamiento de imágenes, se realiza por niveles, que de forma típica se hacen en la secuencia: primero bajo, luego medio y después alto.

El procesamiento a bajo nivel tiene que ver con el trabajo al nivel de imagen binario, normalmente creando una segunda "mejor" imagen a partir de una primera, cambiando la representación de la imagen, removiendo datos indeseados y mejorando los deseados.
El nivel medio de procesamiento se trata de la identificación de formas, regiones o puntos significativos de las imágenes binarias. Poco o ningún conocimiento previo es construido para este proceso de modo que mientras el trabajo puede no ser totalmente a nivel binario, los algoritmos

siguen siendo generalmente no específicos para una aplicación, en la medida que se pueden utilizar en muchos contextos.

En el nivel alto de pre-procesamiento hace un puente de la imagen con alguna base de conocimiento, esta asocia las formas descubiertas durante niveles previos de procesamiento con formas conocidas de objetos reales, el resultado de los algoritmos a este nivel son pasados a los procedimientos que no manipulan imágenes, los que toman decisiones acerca de las siguientes acciones después del análisis de la imagen.

4.2. Representación de las Imágenes

En el apartado de representación de las imágenes, se hablara en primer lugar de la percepción visual en los humanos, ya que comprender este proceso puede ser útil para la creación de sistemas de visión artificiales, luego se abordaran las representaciones del color o modelos de color y por último se describirá como se capturan, representan y almacenan las imágenes en sistemas informáticos [20, 21].

4.2.1. Percepción Visual

Cuando un observador humano procesa imágenes, es importante considerar como son convertidas en información para dicho observador. Entender la percepción visual ayuda en el desarrollo de algoritmos [21].

Los datos de imagen representan cantidades físicas como cromaticidad y luminancia.

- **Cromaticidad**: calidad del color de la luz definida por su longitud de onda. Se percibe como color.
- **Luminancia**: cantidad de luz. Se percibe como brillo.

Como percibimos la información del color de una imagen se puede clasificar en tres variables de percepción: *hue* (matiz), *saturation* (saturación) y *lightness* (luminosidad).

***Hue* o Matiz**: cuando utilizamos la palabra color, normalmente nos referimos al matiz. El matiz distingue entre colores como amarillo y verde. Los matices son las sensaciones de color reportadas por un observador expuesto a varias longitudes de onda. Se ha demostrado que la sensación predominante de longitudes de onda entre 430 y 480 nanómetros es azul, el verde caracteriza un

amplio rango de longitudes de onda de 500 a 550 nanómetros, el amarillo cubre rangos desde 570 hasta 600 nanómetros y longitudes de onda superiores a 610 nanómetros son categorizados como rojo, entre tanto el negro, gris y blanco pueden ser considerados colores pero no matices.

***Saturation* o Saturación**: es el grado al cual un color no es diluido con luz blanca. La saturación disminuye a medida que la cantidad de un color neutral añadido a un matiz puro crece. La saturación es concebida a menudo como cuan puro es un color, los colores no saturados parecen desteñidos o descoloridos, mientras que los colores saturados son vividos y brillantes, como ejemplo se puede decir que el rojo es altamente saturado y el rosado es no saturado; un color puro es 100% saturado y no contiene luz blanca, una mezcla de luz blanca y un color puro tiene una saturación entre 0 y 100 por ciento.

***Lightness* o Luminosidad**: es la intensidad percibida de un objeto reflejante, se refiere a la gama de colores desde blanco pasando por gris hasta negro, un rango frecuentemente denominada como nivel de gris.

***Contrast* o Contraste**: es el rango desde las regiones más oscuras de la imagen hasta las más iluminadas, la representación matemática es:

$$Contraste = \frac{I_{max} - I_{min}}{I_{max} + I_{min}}$$

Donde I_{max} e I_{min} son las intensidades máxima y mínima de una región o imagen.

Las imágenes con un alto contraste tienen grandes regiones de oscuro y luz, entre tanto, las imágenes con un buen contraste tienen una buena representación de todas las intensidades de luminancia.

Cuando el contraste de una imagen crece, el espectador percibe un aumento en el detalle, esto sin embargo es puramente una percepción, ya que la cantidad de información en la imagen no aumenta, solo es que nuestra percepción es sensible al contraste de luminancia, más que de las intensidades de luminancia absolutas [21].

4.2.2. Modelos de Color

Un modelo de color (o espacio de color) [21] es la forma de representar los colores y las relaciones entre ellos. Diferentes sistemas de procesamiento de imágenes utilizan diferentes modelos de color por diversas razones, por ejemplo la industria de las publicaciones a color utiliza el modelo CMY, los monitores CRT a color y la mayoría de sistemas de gráficos de computación utilizan el modelo de color RGB y sistemas que deben manipular matiz, saturación e intensidad de forma separada utilizan el modelo de color HSI, estos solo por citar algunos ejemplos

Modelo RGB (*Red, Green, Blue*)

El espacio de color RGB [21] consiste en tres primarios aditivos: *red* (rojo), *green* (verde) y *blue* (azul), los componentes espectrales de estos colores se combinan aditivamente para producir un color resultante, la cantidad de cada componente de color primario es conocida como su intensidad.

El modelo RGB es representado por un cubo tri-dimensional con rojo, verde y azul en las esquinas de cada eje como se muestra en la Figura 4.2.1, el negro queda en el origen y el blanco en la posición opuesta del cubo, la escala de gris sigue la línea de blanco al negro.

En un sistema de color de 24 bits, con 8 bits por canal, el color rojo seria (255, 0, 0), entones, en un sistema típico de 8 bits las intensidades varían de un mínimo de 0, hasta un máximo de 255 en una sistema de 8 bits por canal, y como cada uno de los tres colores tiene 256 niveles de intensidad, existen 256^3=16,777,216 diferentes combinaciones de las intensidades de rojo, verde y azul, el número de colores conforman la paleta de colores del modelo de color RGB.

El modelo RGB simplifica el diseño de sistemas gráficos de computación pero no es ideal para todas las aplicaciones, los componentes de los colores rojo, verde y azul están altamente correlacionados, lo que hace difícil ejecutar algunos algoritmos de procesamiento de imágenes, muchas técnicas de procesamiento, como los histogramas de ecualización, trabajan solamente en el componente de intensidad de una imagen, este proceso es más fácil de implementar en el espacio de color HSI, *Hue* (matiz), *Saturation* (saturacion) e *Intensity* (intensidad).

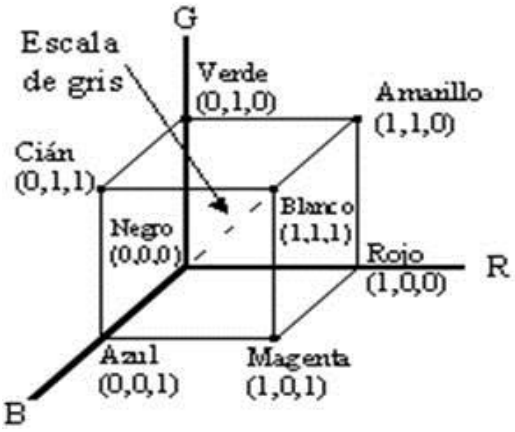


Figura 4.2.1 Cubo de color RGB.

Muchas veces es necesario convertir una imagen RGB a una imagen en escala de grises. Para convertir de color RGB a una escala de grises, se puede utilizar la siguiente ecuación:

$$Intensidad\ de\ escala\ de\ gris = 0.299R + 0.587G + 0.114B$$

La anterior ecuación viene del estándar para la luminancia de la NTSC, (*National Television System Committee*, Comisión Nacional de Sistemas de Televisión en español) es un sistema de codificación y transmisión de televisión en color analógica desarrollado en Estados Unidos en torno a 1940, que como anota Kellison, se emplea en la actualidad en la mayor parte de América, incluida Colombia.
Otra forma de conversión común de color RGB a escala de grises es un promedio simple:

$$Intensidad\ de\ escala\ de\ gris = 0.333R + 0.333G + 0.333B$$

Esta conversión es muy utilizada en muchas aplicaciones, por ejemplo este es un paso en la conversión del espacio de color RGB a HSI.

Como el verde es un gran componente de la escala de grises, muchos lo utilizan por si solo como información de la escala de grises, para reducir aún más los colores a blanco y negro se puede establecer que los valores normalizados por debajo de 0.5 se conviertan a negro y los otros a blanco, esto es simple pero no produce la mejor calidad.

4.2.3. Captura de Imágenes, Representación y Almacenamiento

Las imágenes están almacenadas en una computadora como arreglos 2 dimensiones de números (pixeles), estos números pueden representar información como color, intensidad en la escala de grises, luminancia, crominancia entre otros. La Figura 4.2.2 muestra la estructura de los datos de una imagen.

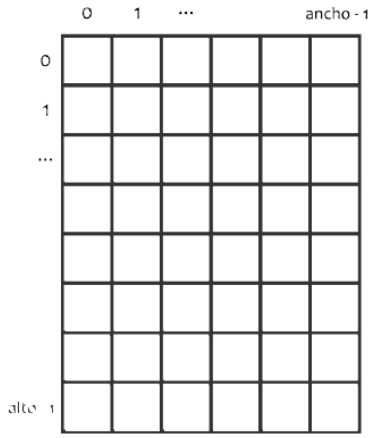


Figura 4.2.2 Estructura de los datos de una imagen.

Antes de que se pueda procesar una imagen en la computadora, se necesita que dicha imagen este en forma digital, para transformar una imagen de tono continuo a una digital se requiere de un digitalizador, los digitalizadores más comunes son los escáneres y las cámaras digitales. Un digitalizador tiene dos funciones, muestreo y cuantización, el muestreo captura puntos de datos uniformemente espaciados, y como estos puntos de datos van a ser almacenados en una computadora, deben ser convertidos a una forma binaria, en la cuantización asigna a cada punto un valor binario.

Una escena es presentada al digitalizador como una imagen continua, a medida que la escena es muestreada, el digitalizador convierte la luz a una señal que representa el brillo, un transductor hace esta conversión, un conversor analógico digital (CAD) cuantiza esta señal para producir datos que pueden ser almacenados digitalmente, estos datos representan la intensidad, normalmente el negro se represente por un 0, y el blanco por el máximo posible.

Imágenes en escala de grises

Los datos en una imagen en escala de grises se almacenan en un solo canal que representa la intensidad, brillo o densidad de una imagen. Las imágenes en escala de grises usan k=8 bits (1 byte) por pixel y la intensidad varia en el rango [0,255], donde 0 representa el brillo mínimo (negro) y 255 el máximo brillo (blanco).

Para muchas aplicaciones de impresión y fotografía digitales, como también en medicina y astronomía, 8 bits por pixel no son suficientes, por lo que se pueden encontrar imágenes de 12, 14 y 16 bits de profundidad en estos dominios.

Imágenes binarias

Las imágenes binarias son un tipo especial de imagen de intensidad en donde los pixeles solo pueden tomar uno de dos valores, blanco o negro. Estos valores son normalmente codificados utilizando un solo bit por pixel (0/1). Las imágenes binarias son usadas para archivar documentos, codificar transmisiones de fax e impresión electrónica [23].

Imágenes a color

La mayoría de las imágenes a color codifican utilizando el espacio de color RGB (Red Green Blue, Rojo Verde Azul en español), el más usado para la visualización de imágenes en la computación, aunque existen otros espacios de color usados en este ámbito, como por ejemplo el espacio YCbCr utilizado en el formato de imagen JPEG para la codificación de la información. Usualmente se emplean 8 bits por cada color de componente, en estas cada pixel requiere 3 x 8 = 24 bits para codificar los tres componentes, y el rango de cada componente es [0-255]. Como pasa con las imágenes de intensidad 30, 36 y 42 bits por pixel son usados comúnmente para aplicaciones profesionales [20, 21].

4.2.4 Operaciones individuales

Como se mencionó con anterioridad, en el procesamiento de imágenes digitales a bajo nivel se manipulan los datos de las imágenes a nivel binario, y el tipo de operaciones más elementales son aquellas que manipulan de a un pixel a la vez, estas son conocidas como operaciones individuales. Las operaciones individuales [23] implican la generación de una nueva imagen modificando el valor del pixel en una posición basándose en una regla global aplicada a cada posición de la imagen original. El proceso consiste en obtener el valor del pixel de una posición dada en la imagen,

modificándolo por una operación lineal o no lineal y colocando el valor del nuevo pixel en la correspondiente posición de la nueva imagen. El proceso se repite para todas y cada una de las localizaciones de los pixeles en la imagen original.

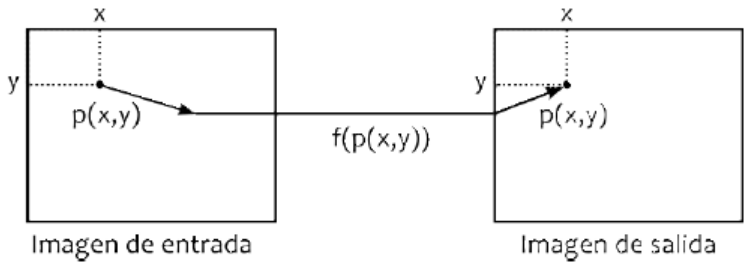


Figura 4.2.3 Operación individual.

Como se aprecia en la Figura 4.2.3, el operador individual es una transformación uno a uno. El operador f se aplica a cada pixel en la imagen o sección de la imagen y la salida depende únicamente de la magnitud del correspondiente pixel de entrada; la salida es independiente de los pixeles adyacentes. La función transforma el valor del nivel de gris de cada pixel en la imagen y el nuevo valor se obtiene a través de la siguiente ecuación:

$$S(x,y) = F(I(x,y)),$$

La función F puede ser un operador lineal o no lineal. La imagen resultante es de la misma dimensión que la imagen original.

Con este tipo de procesamiento pixel por pixel se pueden modificar las imágenes para producir efectos como imágenes negativas, en escala de grises, con efecto sepia, operaciones de umbral y en general modificaciones sobre los canales de color de las imágenes.

Las operaciones aritméticas más usadas en el procesamiento de imágenes son: suma, resta, multiplicación y división. Para que se pueda llevar a cabo una operación aritmética, ambas imágenes deberán ser del mismo tamaño.

Por ejemplo en la Figura 4.2.4 se muestra la suma de dos imágenes, la cual se realiza de la forma:

$$C(x,y) = A(x,y) + B(x,y).$$

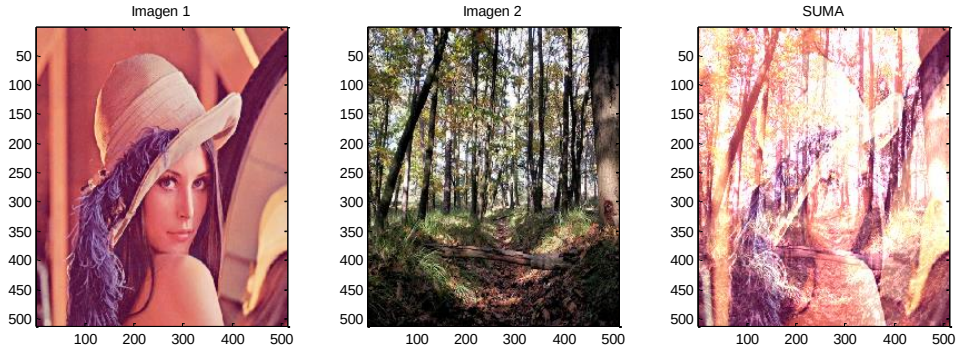


Figura 4.2.4 Suma de dos imágenes.

4.2.5 Operaciones morfológicas

El análisis de imágenes está basado en conceptos matemáticos que permiten describir cuantitativamente imágenes de distintos orígenes. La morfología matemática [19, 21] es una herramienta para extraer componentes de una imagen útiles en la representación y descripción de regiones importantes (bordes, esqueletos…). El lenguaje de la morfología matemática es el de la teoría de conjuntos. Los conjuntos representaran la forma de los objetos en una imagen. La forma de proceder será transformar una imagen en otra, mediante transformaciones en *todo o nada*, para destacar algún rasgo de la imagen primitiva que resultara sencillo de medir.

La morfología matemática se puede usar, entre otros, con los siguientes objetivos:

- Pre-procesamiento de imágenes (supresión de ruido, simplificación de formas).
- Destacar la estructura de objetos (extraer el esqueleto, marcado de objetos, envolvente convexa, ampliación, reducción).
- Descripción cualitativa de objetos (área, perímetro, diámetro, etc.).

Morfología binaria.

Las imágenes binarias son aquellas que tienen dos niveles, generalmente blanco y negro. Por lo tanto, pueden ser representadas mediante conjuntos. Por ejemplo, el conjunto de todos los pixeles blancos de una imagen blanco y negro constituyen una descripción completa de la imagen.

Un elemento estructural es un subconjunto de puntos en Z^2 (es así para nuestro uso discreto, genéricamente son puntos en el espacio Euclidiano de dimensión N: EN), cuya representación en el plano tiene cierta forma y tamaño. El elemento estructural se concibe como un simple parámetro de forma para los filtros morfológicos [19, 21].

La idea básica es probar la imagen con un elemento estructural y cuantificar el modo en que está contenido dentro de la imagen. En una determinada ubicación dentro de la imagen pueden pasar dos cosas: que el elemento estructural esté contenido o que no lo esté. Marcando las ubicaciones en que está contenido obtenemos información estructural de la imagen. Esta depende de la forma y del tamaño del elemento estructural. La característica de estar contenido depende de la relación de subconjunto.

En adelante, llamaremos A, a la imagen de entrada y B al elemento estructural. Las operaciones morfológicas básicas más utilizadas son:

- Dilatación
- Erosión

Dilatación y Erosión

Estas operaciones son fundamentales en el procesamiento morfológico. De hecho, la mayoría de los algoritmos morfológicos están basados en estas dos operaciones [19].

Elemento Estructural:

Si bien los conjuntos A y B pueden ser considerados como una imagen (u objeto), generalmente se considera que A es la imagen y B es el elemento estructural. El elemento estructural es en morfología matemática lo que la máscara (o núcleo) de convolución es en los filtros lineales.

Los elementos estructurales más comunes son los conjuntos que están 4-conectados, N_4, y 8-conectados, N_8, ilustrados en la Figura 4.2.5.

a)

0	1	0
1	1	1
0	1	0

b)

1	1	1
1	1	1
1	1	1

Figura 4.2.5 Elementos estructurales estándar. a) N_4. b) N_8.

DILATACIÓN.

Sean A y B conjuntos en Z^2. La dilatación de A por B, expresada por $A \oplus B$, se define como

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \varnothing\}$$

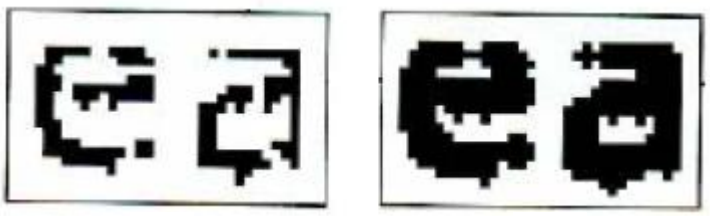
Esta ecuación consiste en obtener la reflexión de B sobre su origen y trasladar esta reflexión por z.

La dilatación de A por B es entonces el conjunto de todos los desplazamientos, z, tal que la reflexión de B y A se solapan por al menos un elemento. Teniendo en cuenta lo anterior, la dilatación de A por B también se puede expresar como

$$A \oplus B = \{z \mid ((\hat{B})_z \cap A) \subseteq A\}$$

En general, la dilatación aumenta el tamaño de un objeto. La cantidad y la forma en que aumenta el tamaño dependen de la elección del elemento estructural.

Una de las aplicaciones más simples de la dilatación es la unión de píxeles relacionados. La Figura 4.2.13 a) muestra un texto cuyos caracteres han perdido píxeles debido a un filtrado con pasabajos. Se sabe que la longitud máxima de las rupturas es de dos píxeles. Si se dilata la imagen con un elemento estructural de conectividad 4 (N_4) como el que se muestra en la Figura 4.2.12 a) se pueden unir los caracteres partidos y obtener como resultado la Figura 4.2.6 b).



a) b)

Figura 4.2.6 a) Letras con poca resolución y partidas. b) Dilatación de la imagen del inciso a) con el elemento estructural N_4.

EROSIÓN.

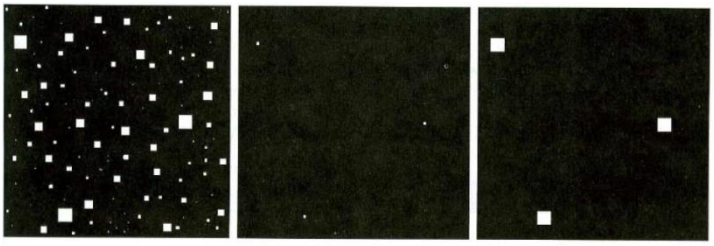
Sean A y B conjuntos en Z^2. La erosión de A por B, que se expresa como AYB, se define como

$$A\Theta B = \{z \mid (B)_z \subseteq A\}$$

Esta ecuación indica que la erosión de A por B es el conjunto de todos los puntos z tales que B, trasladado por z, está contenido en A.

Generalmente, la erosión disminuye el tamaño de los objetos. Como pasaba en la dilatación, la cantidad y la forma en que se produce esta disminución depende del elemento estructural elegido.

Uno de los usos más simples de la erosión es para la eliminación de detalles irrelevantes (en términos de tamaño) de una imagen binaria. La Figura 4.2.7 a) muestra una imagen compuesta por cuadrados cuyos lados tienen 1, 3, 5, 9, y 15 píxeles. Supongamos que queremos eliminar todos los cuadrados excepto los más grandes. Esto los podemos hacer erosionando la imagen con un elemento estructural cuyo tamaño sea un poco menor que el de los cuadrados que deseamos conservar. Por ejemplo, si elegimos un elemento estructural de 13x13 podemos obtener la imagen de la Figura 4.2.7 b). Como se observa, sólo se han mantenido las porciones de los cuadrados más grandes. Luego, podemos restituir el tamaño de estos 3 cuadrados a su tamaño original de 15x15 dilatando la imagen erosionada (Figura 4.2.7 b) con el mismo elemento estructural utilizado para la erosión. El resultado puede verse en la Figura 4.2.7 c).



a) b) c)

Figura 4.2.7 a) Imagen con cuadrados de 1,2, 3, 5, 7, 9 y 15 pixeles de lado. b) Erosión de la imagen del inciso a) con un elemento estructural de 15x15. c) Dilatación de la imagen del inciso b) con el mismo elemento estructural.

La dilatación y la erosión son operaciones duales con respecto a la complementación y a la reflexión. Esto es

$$(A\Theta B)^C = A^C \oplus \hat{B}$$

CAPÍTULO 5

Diseño e Implementación del Software

El propósito de este capítulo es presentar paso a paso el diseño, desarrollo y aplicación del software para el Intérprete Braille-Español/Español-Braille. Para poder explicar mejor como se desarrolló el software vamos a dividirlo en dos secciones, la primera sección trata del diseño y desarrollo del algoritmo de la Memoria Asociativa Bidireccional la cual tendrá como objetivo principal asociar letras en Braille con su correspondiente traducción a caracteres del alfabeto latino, mientras que la segunda sección corresponderá al diseño y desarrollo del algoritmo para el Procesamiento Digital de Imágenes que nos permitirá a partir de una imagen de una letra o texto en Braille obtener un vector binario de dimensión 6 que represente el valor y la posición de cada uno de los seis puntos que componen una letra (celda) Braille para que podamos incluirla como entrada en el algoritmo de las BAM.

El software tendrá dos modos de traducción, el primer modo es de Braille a Español y el segundo modo es de Español a Braille. Para el funcionamiento del traductor de Braille a Español se necesitará una imagen que contenga texto en Braille para poder incluirla en la aplicación y se pueda efectuar la correspondiente traducción al Español. Por otro lado el funcionamiento del traductor de Español a Braille se efectuará de forma que el usuario introduzca el texto que desea traducir mediante teclado y así la aplicación pueda enviarle la traducción en formato Braille. Finalmente lo que se obtendrá como resultado es la traducción en alfabeto latino de una imagen que contenga texto en Braille y viceversa.

5.1 Funcionamiento del algoritmo de la Memoria Asociativa Bidireccional.

El modo de funcionamiento del algoritmo de las Memorias Asociativas Bidireccionales Alfa-Beta enfocado a nuestro tema de tesis, como reconocimiento de patrones Braille, está planteado de la siguiente forma:

1. Se tendrán 26 pares de patrones (p=26), donde el número de patrones corresponde a la cantidad de letras que posee el alfabeto latino y a sus correspondientes representaciones en Braille. Definiremos los vectores de entrada x_k con dimensión de 6 (n=6) y los vectores de salida y_k con dimensión 7.

2. Las propuestas de las dimensiones tanto de los vectores de entrada como de los vectores de salida están dadas de esa forma debido a lo siguiente:

   a) Como se pretende que los vectores de entrada x_k representen cada una de las letras en Braille, se pensó que sería conveniente establecer los vectores de la misma dimensión que la cantidad de puntos que existen en una celda de Braille, así cada vez que haya un punto se establecerá que el valor en el vector de entrada en esa posición tendrá valor de 1, mientras que cuando no haya un punto el valor será 0.

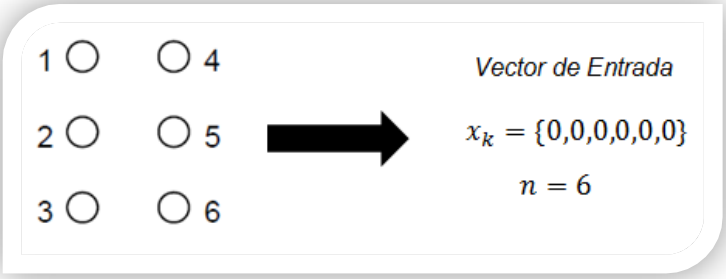


Figura 5.1.1 Dimensión y valores de los vectores de entrada.

A continuación se presenta una tabla que contiene las diferentes combinaciones que tendrán los vectores de entrada en el algoritmo de las BAM.

Tabla 5.1.1 Vectores de entrada x_k representados en forma binaria.

Símbolo Braille	Representación Binaria	Símbolo Braille	Representación Binaria
	Ab={ 1, 0, 0, 0, 0, 0 }		Jb={ 0, 1, 0, 1, 1, 0 }
	Bb={ 1, 1, 0, 0, 0, 0 }		Kb={ 1, 0, 1, 0, 0, 0 }
	Cb={ 1, 0, 0, 1, 0, 0 }		Lb={ 1, 1, 1, 0, 0, 0 }
	Db={ 1, 0, 0, 1, 1, 0 }		Mb={ 1, 0, 1, 1, 0, 0 }
	Eb={ 1, 0, 0, 0, 1, 0 }		Nb={ 1, 0, 1, 1, 1, 0 }
	Fb={ 1, 1, 0, 1, 0, 0 }		Ob={ 1, 0, 1, 0, 1, 0 }
	Gb={ 1, 1, 0, 1, 1, 0 }		Pb={ 1, 1, 1, 1, 0, 0 }
	Hb={ 1, 1, 0, 0, 1, 0 }		Qb={ 1, 1, 1, 1, 1, 0 }

	Ib={ 0, 1, 0, 1, 0, 0 }		Rb={ 1, 1, 1, 0, 1, 0 }
	Sb={ 0, 1, 1, 1, 0, 0 }		Wb={ 0, 1, 0, 1, 1, 1 }
	Tb={ 0, 1, 1, 1, 1, 0 }		Xb={ 1, 0, 1, 1, 0, 1 }
	Ub={ 1, 0, 1, 0, 0, 1 }		Yb={ 1, 0, 1, 1, 1, 1 }
	Vb={ 1, 1, 1, 0, 0, 1 }		Zb={ 1, 0, 1, 0, 1, 1 }

a) Para establecer los vectores de salida que corresponden a cada una de las letras del alfabeto latino y que cada una tenga una combinación binaria diferente se propuso utilizar la tabla del código ASCII y obtener la respectiva combinación binaria de cada una de las letras del alfabeto.

Tabla 5.1.2 Vectores de salida y_k representados en forma binaria, del código ASCII.

Letra Alfabeto	Representación Binaria (ASCII)	Letra Alfabeto	Representación Binaria(ASCII)
A	Abin={ 1, 1, 0, 0, 0, 0, 1 }	**N**	Nbin={ 1, 1, 0, 1, 1, 1, 0 }
B	Bbin={ 1, 1, 0, 0, 0, 1, 0 }	**O**	Obin={ 1, 1, 0, 1, 1, 1, 1 }
C	Cbin={ 1, 1, 0, 0, 0, 1, 1 }	**P**	Pbin={ 1, 1, 0, 0, 0, 0, 0 }

D	Dbin={ 1, 1, 0, 0, 1, 0, 0 }	**Q**	Qbin={ 1, 1, 1, 0, 0, 0, 1 }
E	Ebin={ 1, 1, 0, 0, 1, 0, 1 }	**R**	Rbin={ 1, 1, 1, 0, 0, 1, 0 }
F	Fbin={ 1, 1, 0, 0, 1, 1, 0 }	**S**	Sbin={ 1, 1, 1, 0, 0, 1, 1 }
G	Gbin={ 1, 1, 0, 0, 1, 1, 1 }	**T**	Tbin={ 1, 1, 1, 0, 1, 0, 0 }
H	Hbin={ 1, 1, 0, 1, 0, 0, 0 }	**U**	Ubin={ 1, 1, 1, 0, 1, 0, 1 }
I	Ibin={ 1, 1, 0, 1, 0, 0, 1 }	**V**	Vbin={ 1, 1, 1, 0, 1, 1, 0 }
J	Jbin={ 1, 1, 0, 1, 0, 1, 0 }	**W**	Wbin={ 1, 1, 1, 0, 1, 1, 1 }
K	Kbin={ 1, 1, 0, 1, 0, 1, 1 }	**X**	Xbin={ 1, 1, 1, 1, 0, 0, 0 }
L	Lbin={ 1, 1, 0, 1, 1, 0, 0 }	**Y**	Ybin={ 1, 1, 1, 1, 0, 0, 1 }
M	Mbin={ 1, 1, 0, 1, 1, 0, 1 }	**Z**	Zbin={ 1, 1, 1, 1, 0, 1, 0 }

Una vez construidos los vectores de entrada y de salida se procede a entrar en la fase de aprendizaje de la memoria asociativa bidireccional.

5.1.1 Fase de Aprendizaje.

Se aplica la transformada vectorial de expansión a los vectores x_k , para obtener cada uno de los vectores X^k y $\bar{X}^k$.

$$\boldsymbol{X = \tau^e(x, h)}$$

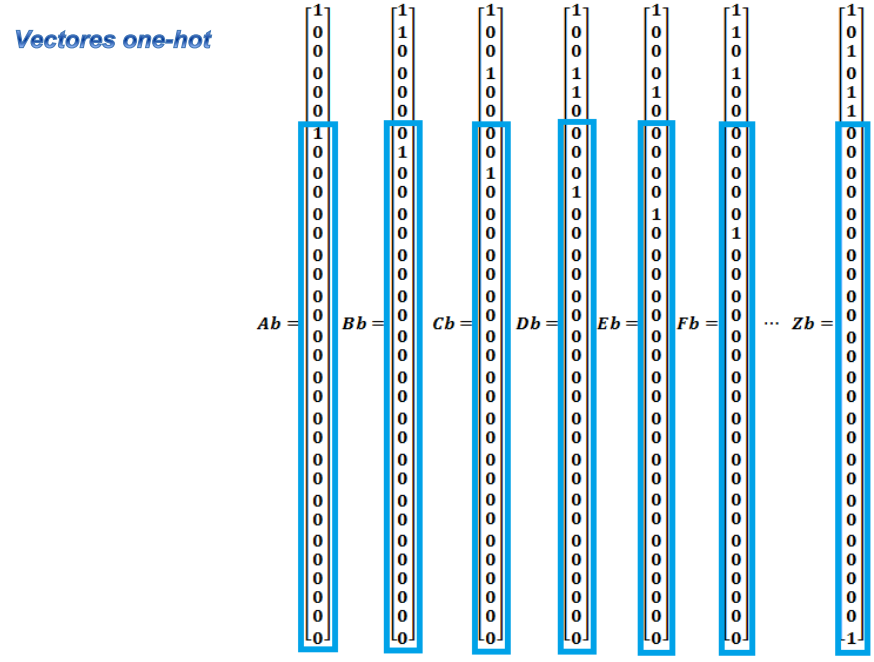


Figura 5.1.2 Creación de los vectores one-hot.

$$\bar{X} = \tau^{e}(x, \bar{h})$$

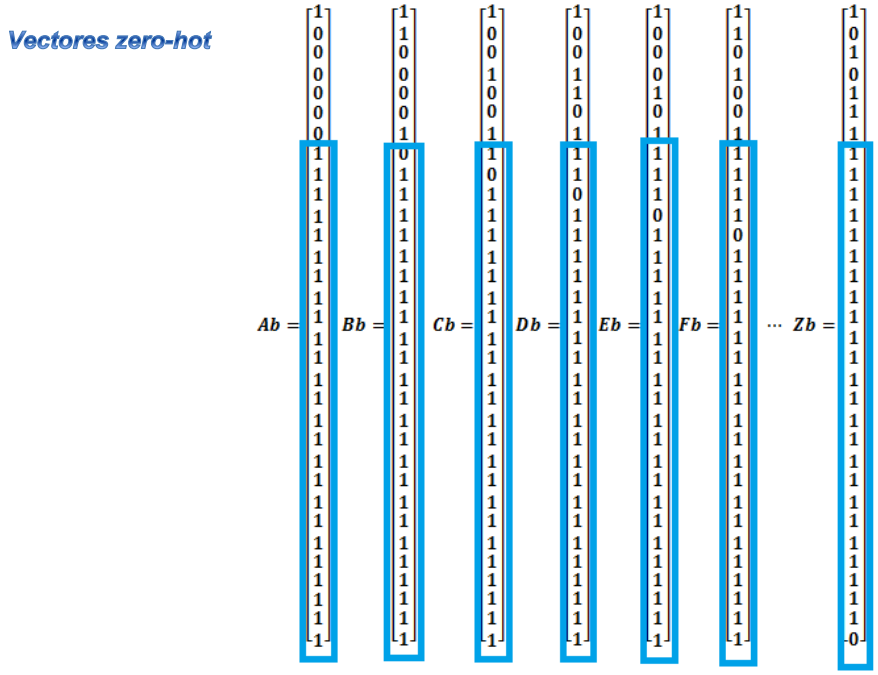


Figura 5.1.3 Creación de los vectores zero-hot.

- Se generan las memorias autoasociativas Alfa-Beta *max* $\mathbf{V} = \bigvee_{k=1}^{4} \left(\mathbf{X}^k \otimes \left(\mathbf{X}^k \right)^t \right)$ y *min* $\mathbf{\Lambda} = \bigwedge_{k=1}^{4} \left(\overline{\mathbf{X}}^k \otimes \left(\overline{\mathbf{X}}^k \right)^t \right)$

$\boldsymbol{V} =$

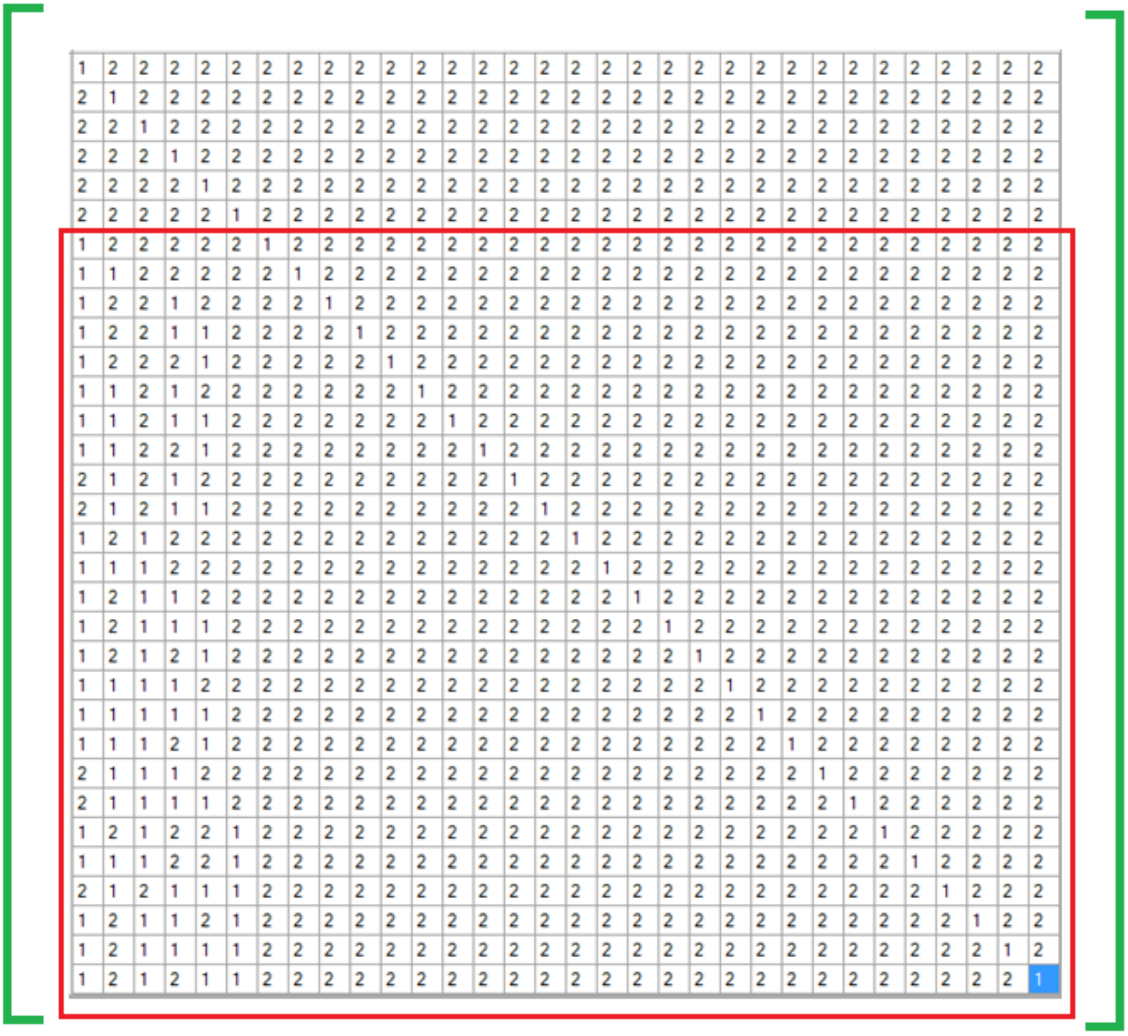


Figura 5.1.4 Matriz obtenida usando el operador MAX.

Λ=

Figura 5.1.5 Matriz obtenida usando el operador MIN.

- Se crea el *Linear Associator* Modificado utilizando los patrones de salida **y**

LAy=

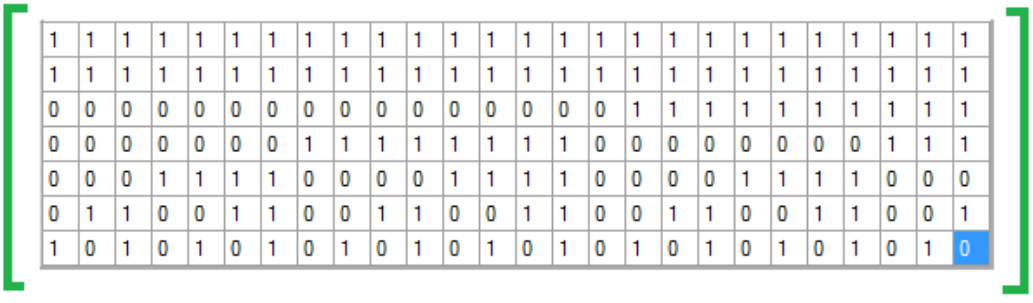
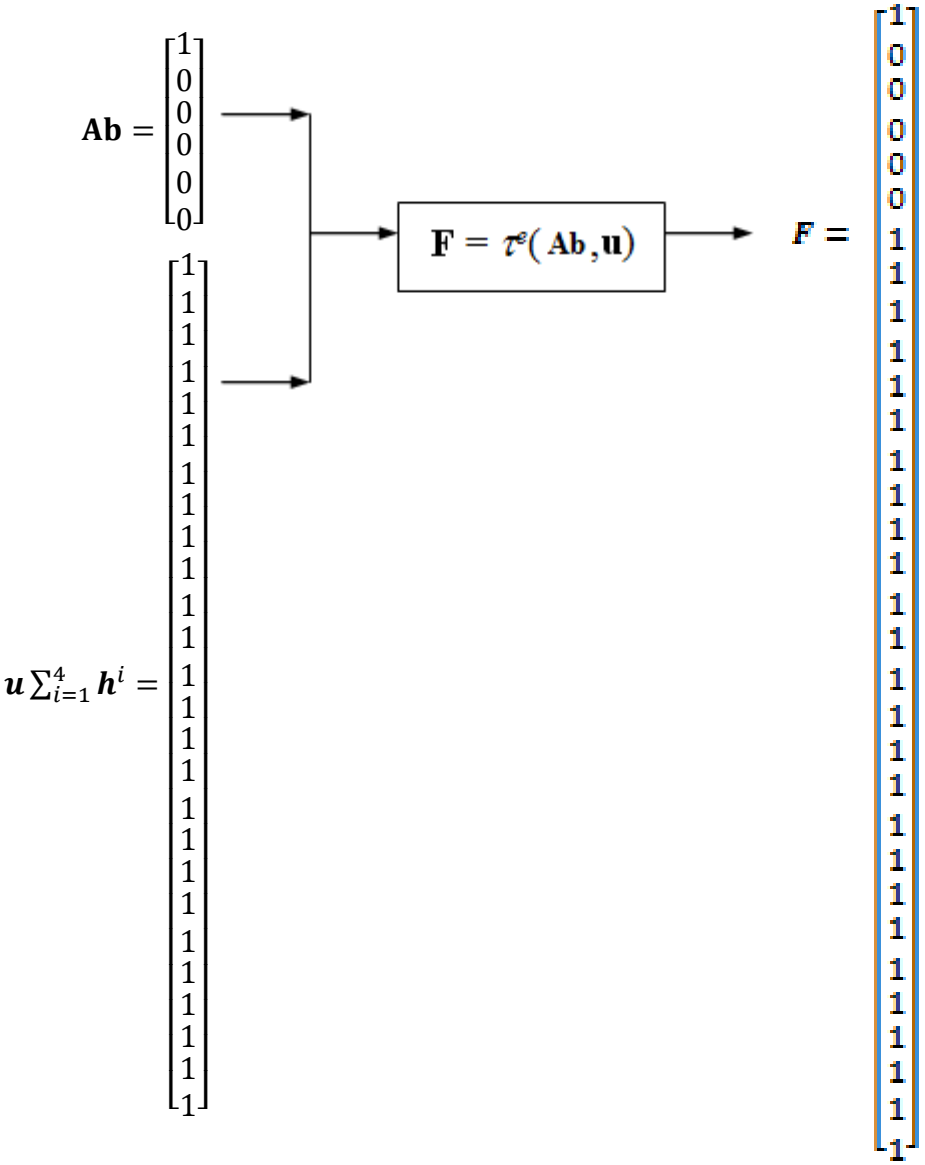


Figura 5.1.6 Linear Associator Modificado.

5.1.2 Fase de Recuperación

- Se presenta a la entrada de la BAM Alfa-Beta el patrón $\mathbf{x}^3$, se construye el vector **u** y se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *max* **V** con **F**

$$\mathbf{R} = \mathbf{V}\Delta_{\beta}\boldsymbol{F} =$$

1
0
0
0
0
0
1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

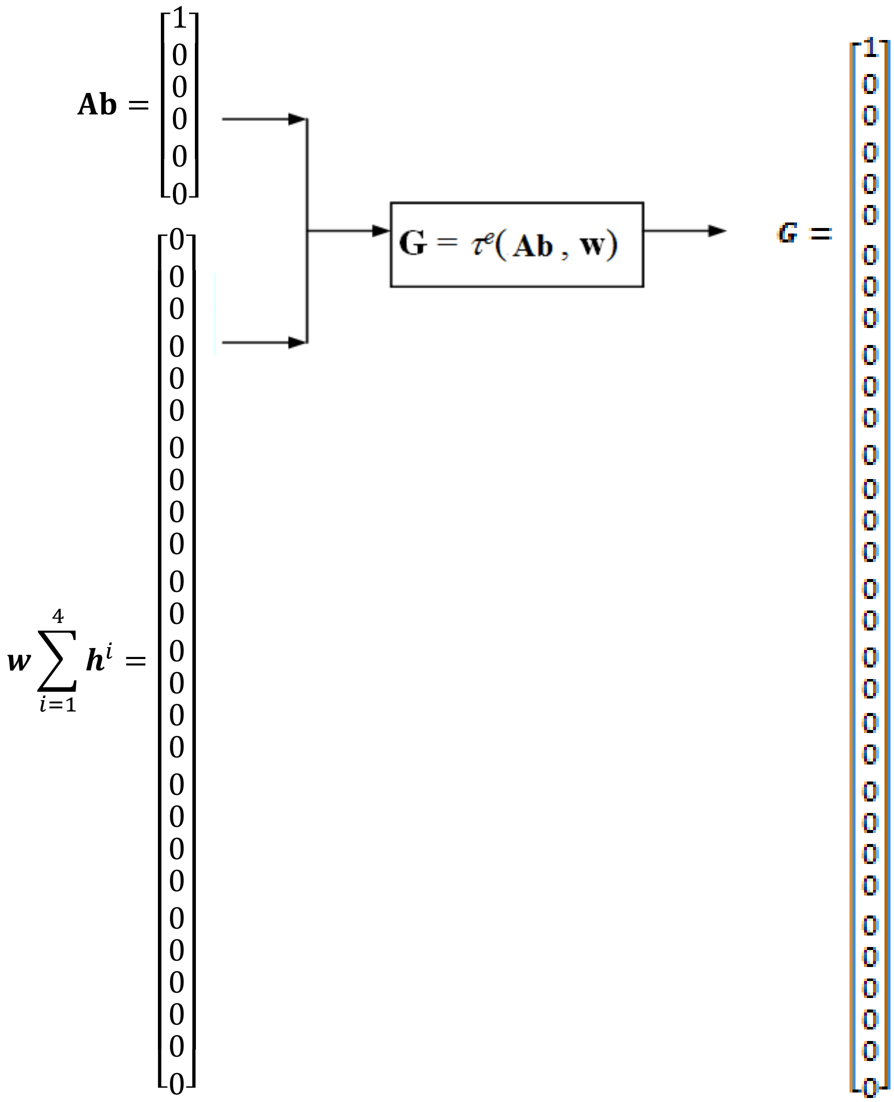
Contracción

$$\boldsymbol{r} = \tau^{c}(R, 26) =$$

1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Debido a que **r** no es un vector *one-hot*, entonces se continúa con el algoritmo.

- Se construye el vector **u** y con el patrón $\mathbf{x}^3$, se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *min* $\boldsymbol{\Lambda}$ con $\mathbf{G}$

$$\mathbf{S} = \mathbf{V}\Delta_{\beta}\boldsymbol{G} = \begin{matrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1\\1\\0\\0\\1\\0\\0\\0\end{matrix} \xrightarrow{\text{Contracción}} \boldsymbol{s} = \boldsymbol{\tau}^{c}(\boldsymbol{S}, \mathbf{26}) = \begin{matrix}0\\0\\0\\0\\0\\0\\0\\0\\1\\1\\0\\0\\0\\0\\0\\0\\0\\0\\1\\1\\0\\0\\1\\0\\0\\0\end{matrix}$$

El vector **s** no es un vector *zero-hot*, por lo que se continúa con el proceso.

- Se realiza la operación AND entre el vector **r** y el vector negado de **s**, $\bar{s}$, para obtener el vector **t**

$$\boldsymbol{t} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Con esta operación se obtiene el tercer vector *one-hot*

- Se obtiene el vector de salida **Abin** (Vector Recuperado) mediante

$$\boldsymbol{y_{rec}} = \boldsymbol{LA_y} * \boldsymbol{t}$$

Por lo tanto el vector recuperado es:

$$Abin = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Finalmente ya que se obtiene el vector de salida, que en este caso es el Vector **Abin** el cual efectivamente corresponde a la salida correcta de acuerdo a la entrada introducida **Ab,** se procede a transformar ese resultado de manera que en lugar de un vector nos devuelva un carácter que corresponda a esa combinación binaria, basándonos en el código ASCII. Es decir, que el valor final que devuelve el algoritmo para este caso en particular es la letra "**A**" que corresponde al valor binario 1100001 expresado en el vector de salida previo.

5.2 Funcionamiento del algoritmo para el Procesamiento Digital de la Imagen

El algoritmo que se encargará de efectuar el procesamiento digital de imágenes, como se mencionó previamente pretende que a partir de una imagen de un texto en Braille podamos manipularla implementando diversas técnicas del PDI con el fin de extraer solamente la información que permita identificar el total de celdas Braille existentes en la imagen, así como el valor y la posición de cada uno de los puntos por celda, para que como resultado final el algoritmo nos pueda proporcionar en forma de vectores de dimensión 6 cada una de las celdas que logre identificar y de esta manera utilizar estos resultados para poder ingresarlos a nuestra Memoria Asociativa Alfa-Beta.

5.2.1 Diseño y elaboración de un documento en Braille

Para fines de este trabajo de tesis se diseñaron y elaboraron diversos documentos en Braille mediante la escritura manual con el uso de regleta y punzón. Cabe mencionar que para redactar el documento es preciso escribir los signos invertidos, es decir, se escribe de derecha a izquierda invirtiendo el orden de la numeración de los puntos dentro de una celda, lo cual hace que su aprendizaje sea un poco más complicado, al menos las primeras veces, ya que exige a la persona que lo está usando tener bien definida su lateralidad y el concepto de reversibilidad.

Una de las ventajas al escribir los documentos en Braille es que se pudo manipular un formato estándar para poder hacer las pruebas correspondientes con el software y poder evitar de alguna manera algunas anomalías que pudieran estar presentes en otro tipo de redacciones. Al comenzar a escribir el documento debemos asegurarnos de que la hoja de papel Braille este correctamente ubicada dentro de la regleta para que las filas se encuentren lo más alineadas posible, también el punzón debe estar perfectamente perpendicular al papel con el fin de propiciar que cada uno de los puntos se marquen uniformemente. El dedo índice de la mano que no escribe precede al punzón sobre la línea de escritura (no olvidar que se escribe de derecha a izquierda), ayudando a la

localización de la celda y a calcular los espacios que quedan libres para no cortar una palabra incorrectamente. Antes de comenzar a escribir letras conviene adquirir mecánicamente precisión en el punteado, ejerciendo la misma presión en todos los puntos. Una vez obtenidos los documentos terminados se procede a buscar un modo de digitalizarlos de forma que puedan ser usados por el software.

5.2.2 Modo de digitalización de la imagen

Con base en diferentes pruebas realizadas utilizando modos de digitalización distintos se determinó que para el correcto funcionamiento del Intérprete se necesita que la imagen sea digitalizada por medio de un escáner, y con ello obtener la mayor efectividad posible en la traducción.

El escáner es un dispositivo que permite digitalizar una imagen, la imagen se rastrea mediante elementos semiconductores sensibles a la luz (LDR). La señal emitida por estos semiconductores se traduce en una secuencia de bytes que puede ser almacenada como un archivo. Un parámetro muy importante a tomar en cuenta es la resolución. La resolución es la distancia que existe entre los detectores luminosos que tiene el escáner su valor se mide en dpi (puntos por pulgada), y de ellos depende, en buena parte, su calidad.

Digitalización de hoja Braille:

Para efectos de las pruebas realizadas en este proyecto, la hoja del escrito en Braille fue convertida a formato digital a través del escáner de un multifuncional EPSON Stylus TX130. Tras realizar varias pruebas variando las características de la configuración del scanner, tales como su resolución, su color, y los formatos de imagen permitidos, se decidió establecer una resolución de 100ppp (puntos por pulgada), un formato de imagen tipo JPG y que el escaneo se realizará en escala de grises. Estas características permiten tener una imagen aceptable para poder ser procesada en los siguientes bloques. (Figura 5.2.1)

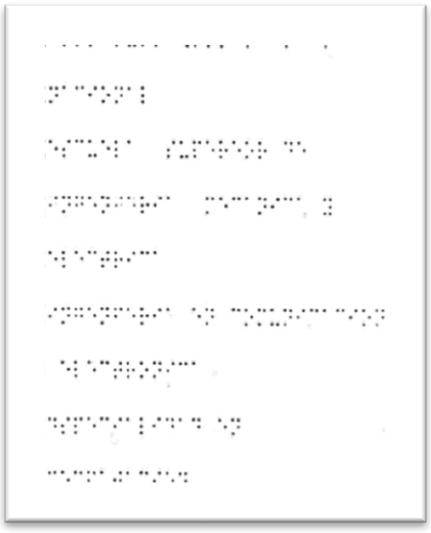


Figura 5.2.1 Hoja de escritura Braille escaneada

5.2.3 Procesamiento de la imagen usando MATLAB

MATLAB (Matrix Laboratory) es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para diversas plataformas entre ellas Windows. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas de otros lenguajes. Debido a que las herramientas de MATLAB son un gran apoyo para las imágenes generadas por un amplio rango de dispositivos incluyendo escáneres. Es posible visualizar, analizar y procesar estas imágenes. Los soportes estándares de MATLAB para datos incluyen JPEG, TIFF, PNG, HDF, HDF-EOS, FITS, Microsoft Excel, ASCCI y archivos binarios.

MATLAB es un producto con un amplio desarrollo en ambientes para la creación de algoritmos y aplicaciones científicas y de ingeniería, nos provee de herramientas poderosas para cada paso en el proceso. El alto nivel del lenguaje, el ambiente interactivo para la programación, funciones matemáticas pre-construidas, editor y depurador de herramientas. Con MATLAB no se tienen que desarrollar algoritmos desde cero o trabajar con interfaces complicadas para librerías externas como es común con C y C++.

Es por todo lo anterior que en un principio el algoritmo se había comenzado a implementar bajo el lenguaje M, gracias a las herramientas que posee MATLAB con respecto al procesamiento digital de imágenes nos fue posible plantear un método que nos permitiera hacer el reconocimiento de imágenes y aunque éste no fue lo suficientemente funcional debido a algunas desventajas que se tuvieron al hacer las pruebas experimentales, se describe brevemente debido a que más adelante nos serviría de referencia para poder desarrollar nuestras propias funciones pero ahora implementadas con el lenguaje de programación C# en Visual Studio 2012.

El proceso que se efectúo para el procesamiento digital de imágenes y los resultados obtenidos se presentan a continuación:

Paso 1. Elegir una imagen de un símbolo Braille que nos sirva como base que contenga los seis puntos seleccionados.

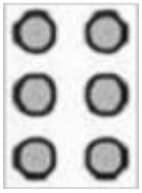


Figura 5.2.2 Celda Braille base.

Paso 2. Elegir la imagen del símbolo Braille del cual queremos identificar sus puntos representativos. En este caso elegimos para trabajar el símbolo que representa la letra **u**.

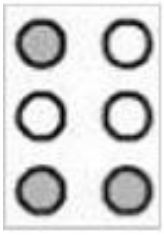


Figura 5.2.3 Símbolo que representa la letra u.

Paso 3. Convertir a escala de grises para poder binarizar cada una de las imágenes, de tal forma que cada pixel de la imagen tenga únicamente dos posibles valores, es decir 255 y 0 para blanco y negro respectivamente.

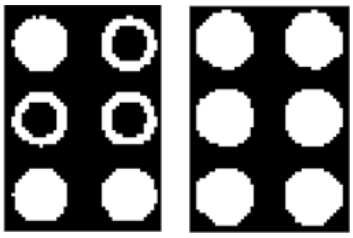


Figura 5.2.4 Proceso de binarización de la imagen.

Paso 4. Se realiza un proceso de erosión en la imagen con el objetivo de eliminar los contornos de los puntos no utilizados

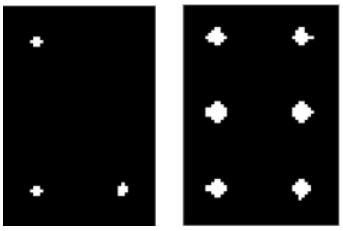


Figura 5.2.5 Proceso de erosión de la imagen.

Paso 5. Debido a que el tamaño de los objetos en la imagen es muy pequeño se hace una fase de dilatación, hasta tener un tamaño considerable.

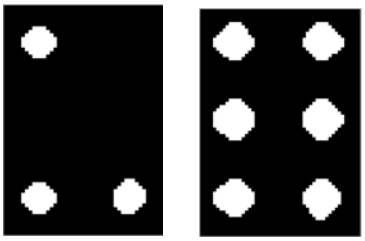


Figura 5.2.6 Proceso de dilatación de la imagen.

Paso 6. Se hace un etiquetado de cada objeto de la imagen, con color diferente se puede diferenciar un objeto de otro en cada casilla.

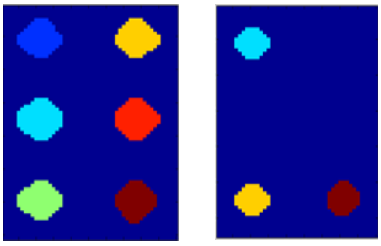


Figura 5.2.7 Proceso de segmentación de la imagen.

Paso 7. Se obtienen los centroides de cada objeto, a partir de la imagen obtenida en el paso anterior, y las coordenadas de estos son comparadas una por una.

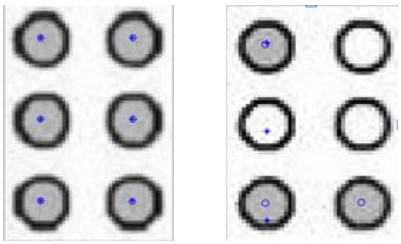


Figura 5.2.8 Identificación de objetos.

Paso 8. En el caso en que cada centroide, correspondientes a los elementos de la casilla de entrada, coincida en un rango aproximado con alguno de los centroides de la casilla base se establece que existe un punto en esa posición. Esto se traduce a un vector de 6 elementos, que corresponden respectivamente a las seis posiciones de la casilla Braille, identificando a cada punto existente como un uno y los espacios como ceros.

```
caracter =

     1
     0
     1
     0
     0
     1
```

Este vector o "código Braille" será la entrada para la memoria alfa beta previamente creada, de tal forma que al final se recupere un código binario correspondiente al carácter ASCII de la letra ingresada.

El resultado obtenido se muestra a continuación:

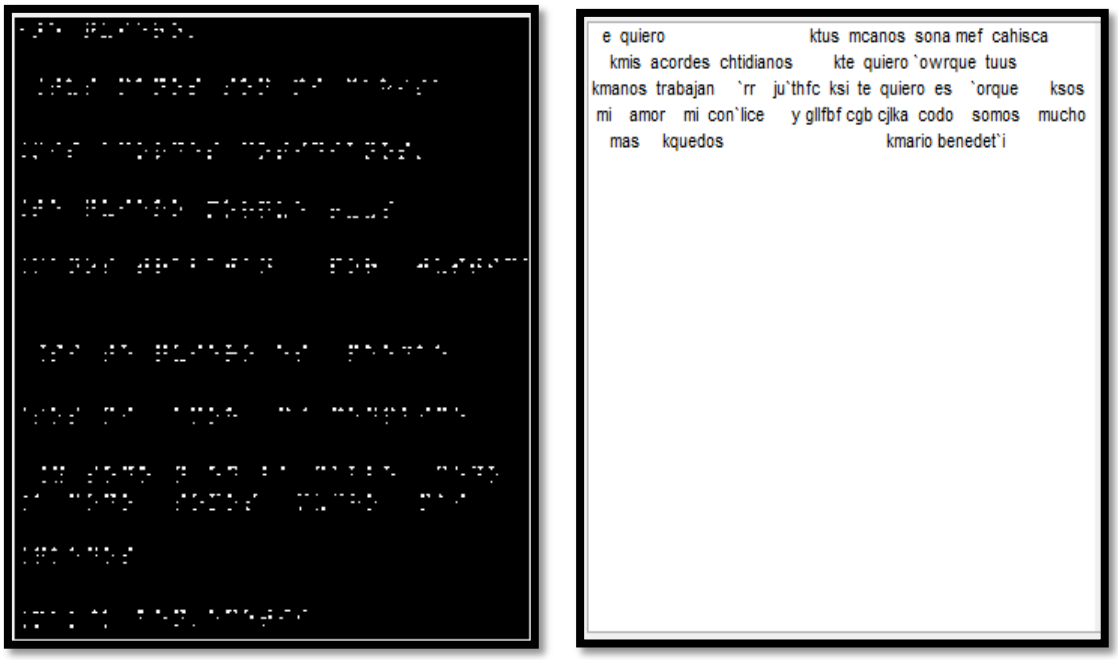


Figura 5.2.9 Resultados obtenidos del PDI usando MATLAB.

Como se puede observar, el resultado obtenido cuenta con muchas inconsistencias en la traducción de la hoja, con líneas de escritura sin sentido y traslapes de letras en la mayoría de las palabras.
Es por ello que decidió implementarse un nuevo algoritmo, que fuese más preciso, ya que en este primer intento se considera a que los relieves de la imagen digitalizada son uniformes y del mismo tamaño, por lo que al compararse con la casilla base, la localización de los centros de cada objeto es muy deficiente.

La utilización de MATLAB en los inicios del prototipo, como se mencionó al principio tuvo como objetivo principal acelerar el proceso de programación, y pruebas del mismo, sin embargo una vez planteadas las etapas necesarias para el traductor se comenzó a desarrollar todo el algoritmo utilizando lenguaje C#, aunado a la interfaz gráfica que Visual Studio provee.

5.2.4 Procesamiento de la imagen usando Visual Studio 2012

Microsoft Visual Studio es un entorno de desarrollo integrado dirigido a sistemas Windows, éste permite crear diversas aplicaciones para computadoras, dispositivos móviles, sitios web, etc.
Entre las ventajas más importantes que posee este entorno, se encuentra una interfaz la posibilidad de desarrollar algoritmos cuyos códigos sean estables y reduzcan significativamente la aparición de

errores. Además, particularmente en la versión 2012, se tiene la posibilidad de crear aplicaciones del sistema operativo Windows 8.

El lenguaje definitivo utilizado en este proyecto fue el lenguaje C#, el cual es soportado por Visual Studio y con gran versatilidad.

El procesamiento de imagen se divide en varios bloques o funciones, para su mejor comprensión se pude observar el diagrama de bloques (Figura 5.2.10)

Figura 5.2.10 Diagrama de bloques del procesamiento de la imagen en VS2012.

El objetivo del procesamiento de la imagen es obtener un vector Braille (conformado por 6 elementos correspondientes a cada posición de la casilla Braille) para cada uno de los caracteres que conforman el escrito a traducir y que ha sido digitalizado.

Binarización y filtrado

Una vez que se ha digitalizado la imagen, es necesario convertirla a un mapa de bits, éste permite la manipulación y edición de los valores de las tres capas de color de cada pixel.

Una vez hecho este proceso, la imagen como mapa de bits se binarizó, es decir, se le aplicó una conversión a blanco y negro, para ello se calculó el promedio de los valores de las capas de color, pixel por pixel. Si el valor del promedio es más cercano a cero se asigna el color negro a ese pixel, en caso contrario se asigna el color blanco.

Para determinar la "cercanía" se utilizó un valor "a" que representa el porcentaje, o bien, el punto de referencia para asignar valores blanco o negro.

$$pixel = \begin{cases} Blanco\ si & promedio > 255a \\ Negro\ si & promedio < 255a \end{cases}$$

Por ejemplo, si una imagen es binarizada con un factor del 50% (a=0.5) entonces se asignará un valor 0 (color Negro) a todo pixel cuyo promedio de sus valores rojo, verde y azul den como resultado un valor dentro del rango 0 a 127 se establecerá un valor de cero a ese pixel, en caso contrario, cuando todos los valores superen este rango, el valor asignado es un 255, representando al color blanco.

Entre mayor sea el porcentaje "a", la binarización será más selectiva, eliminando información innecesaria en la imagen y resaltando únicamente los relieves o puntos de la hoja en Braille. (Figura 5.2.11).

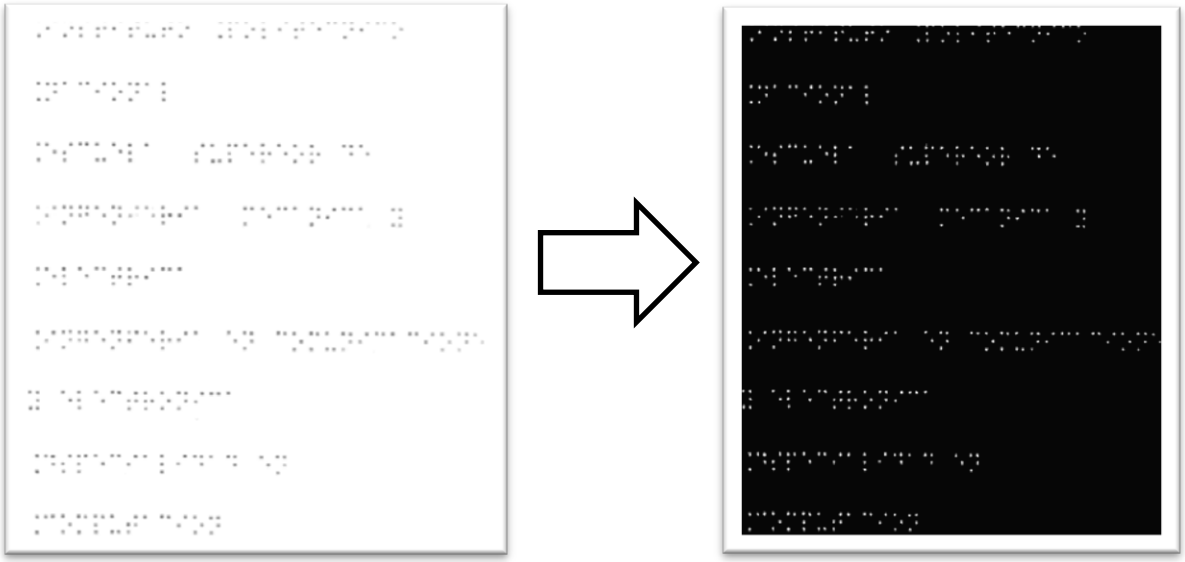


Figura 5.2.11 Hoja de escritura Braille después de un proceso de binarización.

Posteriormente, se hizo un análisis de densidad de pixeles blancos en cada renglón de la imagen, con el objetivo de descartar puntos en blanco que no representen nada, rescatando los puntos o relieves de la hoja.

El algoritmo cuenta en cada reglón el número de pixeles en blanco, si esta cantidad es muy pequeña en comparación al ancho en pixeles de un punto Braille, el renglón completo es iluminado. En caso contrario el renglón permanece intacto. (Figura 5.2.12)

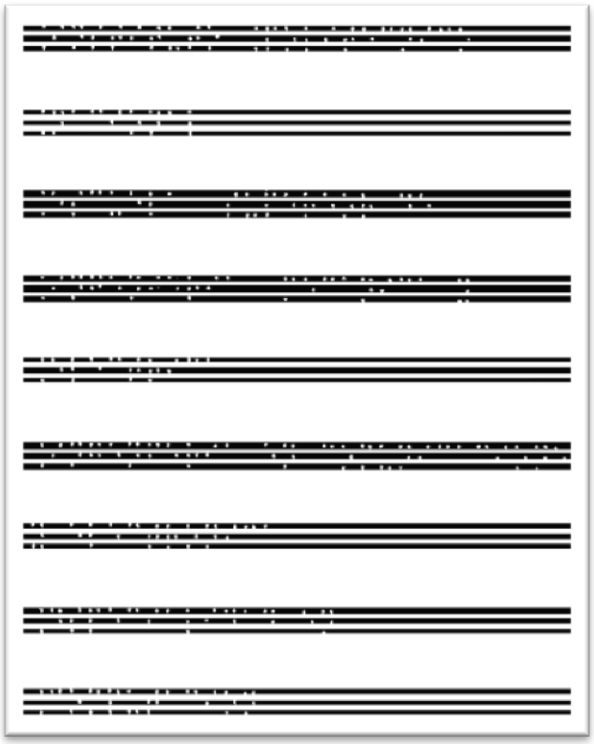


Figura 5.2.12 Hoja de escritura Braille después de eliminación de regiones con información no relevante

Detección de renglones

Como puede observarse, desde el subproceso anterior los renglones correspondientes a un ancho de punto son identificados, por lo que para la detección y selección de un renglón correspondiente a un ancho de casilla Braille es necesario conjuntar tres renglones simultáneos.

Esto arroja como resultado una imagen con una dimensión horizontal igual a la de la hoja original y una dimensión vertical correspondiente a la altura de una casilla en Braille. (Figura 5.2.13)

Figura 5.2.13 Sección de la hoja en Braille que representa a un renglón de escritura en este sistema.

Esta sección de la imagen será nuevamente analizada para detectar cada casilla, es decir, cada carácter.

Detección de casillas

Las casillas en el renglón (Figura 5.2.13) no están determinadas explícitamente en la imagen, por lo que se hace una aproximación de sus dimensiones, partiendo del tamaño de un punto en Braille.

Para conocer el ancho y simultáneamente el número de columnas en la línea de escritura seleccionada se hace una dilatación vertical de este segmento, en otras palabras se ilumina en blanco cada columna en la cual exista por lo menos un pixel en blanco. El resultado de este proceso se muestra en la Figura 5.2.14.

Figura 5.2.14 Dilatación del renglón de escritura Braille

Debido a que un carácter Braille puede o no utilizar las dos columnas de la casilla, no se puede dar por hecho que dos columnas, detectadas en este segmento de imagen, formen una casilla ya que, como puede observarse, las distancias entre una columna es aproximadamente igual en algunos pares, sin embargo existen columnas con una distancia mayor con respecto a su columna sucesiva o previa. Por tanto es necesaria una inspección de cada columna y sus alrededores para determinar si junto con las columnas de su derecha y/o izquierda conforman una celda, además de conocer qué posición ocupa en ella.

Debe mencionarse que se hizo una correspondencia de las dimensiones de la casilla Braille en función del tamaño de sus columnas, partiendo de las dimensiones estándar de los caracteres del sistema de escritura para invidentes. (Figura 5.2.15)

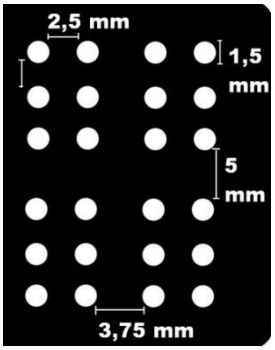


Figura 5.2.15 Medidas estándar del Sistema Braille

Ya que el ancho de un punto es de 1.5mm, este valor se toma como referencia para saber las proporciones de cada parámetro en la casilla.

Tabla 5.2.1 Proporción de las dimensiones de casilla en función del ancho de un punto Braille.

Parámetro	Medida	Proporción
Ancho de punto Ancho de columna	1.5mm	1
Distancia entre columnas	3.75mm	2.5
Ancho de casilla	5.5mm	3.66

Para establecer el ancho de un relieve o punto en la imagen dilatada, se hizo un barrido horizontal en una de sus filas, y se obtuvo el valor mayor del ancho de las columnas en blanco.
El funcionamiento del algoritmo se explica a continuación:

1. Se parte de la ubicación de la columna más cercana al lado izquierdo de la imagen. A partir de su inicio se toma una distancia equivalente a tres veces el ancho de una columna (valor mayor previamente identificado), con el fin de abarcar una longitud aproximadamente igual al ancho de una casilla.

Figura 5.2.16 a) Selección de región en la cual solo existe una columna, b) y c) Selección de región que contiene dos columnas

2. Se hace un análisis a lo largo de esta nueva subsección para detectar más columnas a la derecha de la ya seleccionada.
3. En caso de que sean dos las columnas en esta región, incluyendo la seleccionada (Figura 5.2.16.b), entonces se considera como una casilla completa, es decir, una casilla que utiliza las dos columnas
4. En otro caso, cuando sólo exista una columna en la región examinada se hace una comparación respecto a la posición de la columna posterior, si esta distancia rebasa tres veces el ancho de un punto pero es menor que cuatro veces este valor, entonces se considera que la columna se ubica

en el lado izquierdo de la casilla Braille, en caso contrario, si esta distancia es aún mayor se considera a la columna en el lado derecho. Para cualquiera de estos casos, se reubica al renglón en la región de la casilla. (Figura 5.2.17)

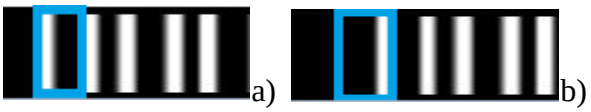


Figura 5.2.17 a) Ubicación inicial de un sólo renglón. b) Reubicación del renglón del lado derecho de la casilla

5. El proceso se repite hasta que todas las columnas de la figura sean reubicadas, partiendo de la columna inmediata a la última acomodada.(Figura 5.2.18)

Figura 5.2.18 a) Ubicación de todas las casillas respecto a la posición de cada columna dilatada

Creación de Vector Braille

Ya que se ha identificado la ubicación de cada casilla en la línea de escritura, se selecciona una a una para determinar qué carácter representa. Para ello, la casilla es seccionada como una matriz de 3x2, con el objetivo de identificar la posición de los puntos utilizados. (Figura 5.2.19)

Figura 5.2.19 Casilla seccionada.

Por otra parte, se declara un vector de seis elementos con valores iniciales de 0.

$$vector\ braille = [0,0,0,0,0,0]$$

Se hace una inspección en cada subregión de la casilla para identificar si existe un punto o no, en caso afirmativo se asigna un 1 al elemento del vector correspondiente, los demás continuarán intactos.

$$vector\ braille = [0,1,0,1,0,0]$$

Este vector será la entrada para la memoria alfa-beta creada previamente.

Cabe señalar que el sistema Braille posee prefijos que indican si los caracteres inmediatos son números mayúsculas, etc, por lo que antes de ingresarlo a la memoria es importante hacer una inspección previa. Por ejemplo, para el identificador de mayúsculas (Figura 5.2.20) se obtiene un vector:

$$vector\ braille = [0,0,0,1,0,1]$$

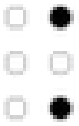


Figura 5.2.20 Carácter especial, indicador de letras en mayúscula.

Al identificar este vector, se resta un 32 binario al carácter viario devuelto por la memoria asociativa, con el fin de escribir el carácter siguiente como una letra mayúscula del código ASCII.

5.3 Interfaz Gráfica

La interacción de un usuario y el Intérprete requiere de una interfaz, que permita iniciar de manera correcta cada uno de las funciones de procesamiento de imágenes así como el uso de la asociación de las memorias, todo esto de una forma amigable y sencilla. Figura 5.3.1

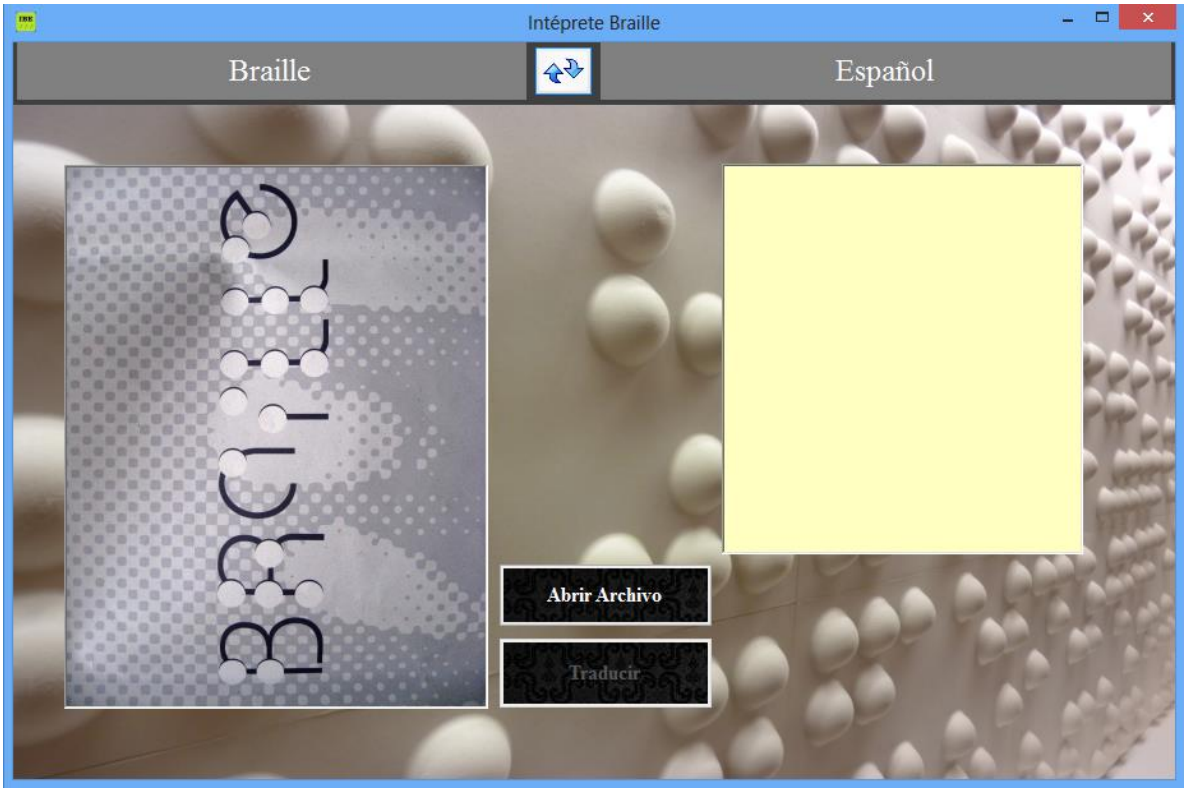
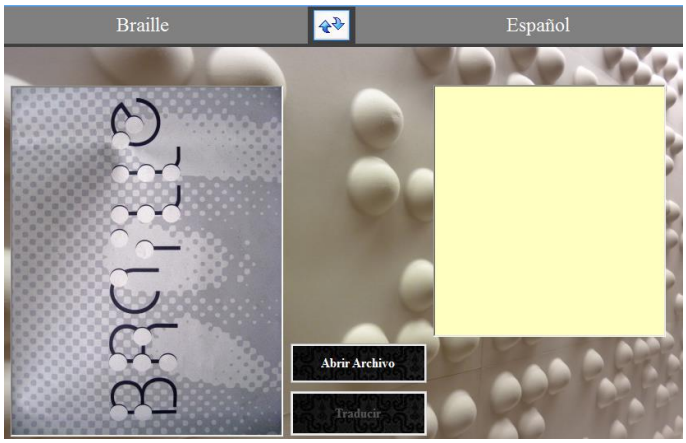
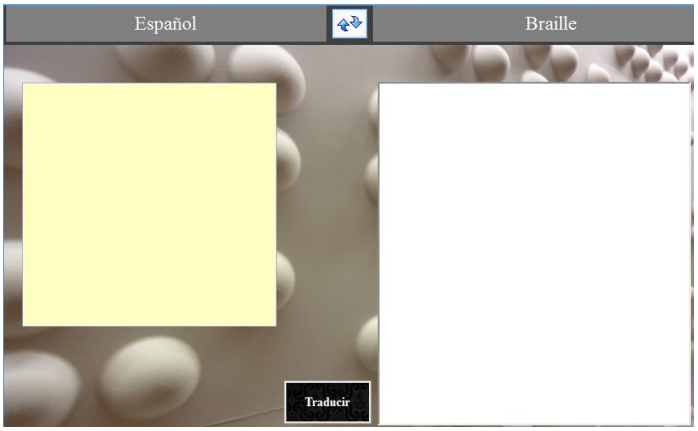


Figura 5.3.1. Interfaz gráfica del intérprete.

Ya que el software posee dos procesos distintos, traducción español a braille y traducción braille español, se colocó una carátula por cada proceso. Figura 5.3.2



a)



b)

Figura 5.3.2 Carátulas de la interfaz gráfica del intérprete, a) traducción Braille a español, b) traducción español a Braille.

La elección de cada proceso se realiza mediante el uso de un control Botón (Figura 5.3.3) que al ser presionado mediante un clic del cursor, muestre sólo una carátula correspondiente a la traducción que se desee realizar.

Figura 5.3.3 Botón de selección de traducción.

En cada carátula existen dos zonas, correspondientes a cada tipo de escritura, es decir, braille y español.

En lo que corresponde a la interfaz Braille-Español, se utiliza un control PictureBox en donde se almacenará la imagen de escritura Braille, previamente escaneada.

En la zona del lado derecho, se cuenta con un control TextBox, en el cual se mostrará la traducción realizada.

En el segundo proceso, traducción español-braille, se hace una traducción directa, es decir, el texto original es ingresado por el usuario mediante el teclado en el TextBox del lado izquierdo. El resultado de la traducción se muestra en el segundo TextBox.

Todos los controles utilizados en cada carátula, están colocados en un control PanleLayout ajustándolo a las dimensiones del formulario, de tal forma que al cambiar el tamaño del mismo, los controles se autoajusten tamaños proporcionales.

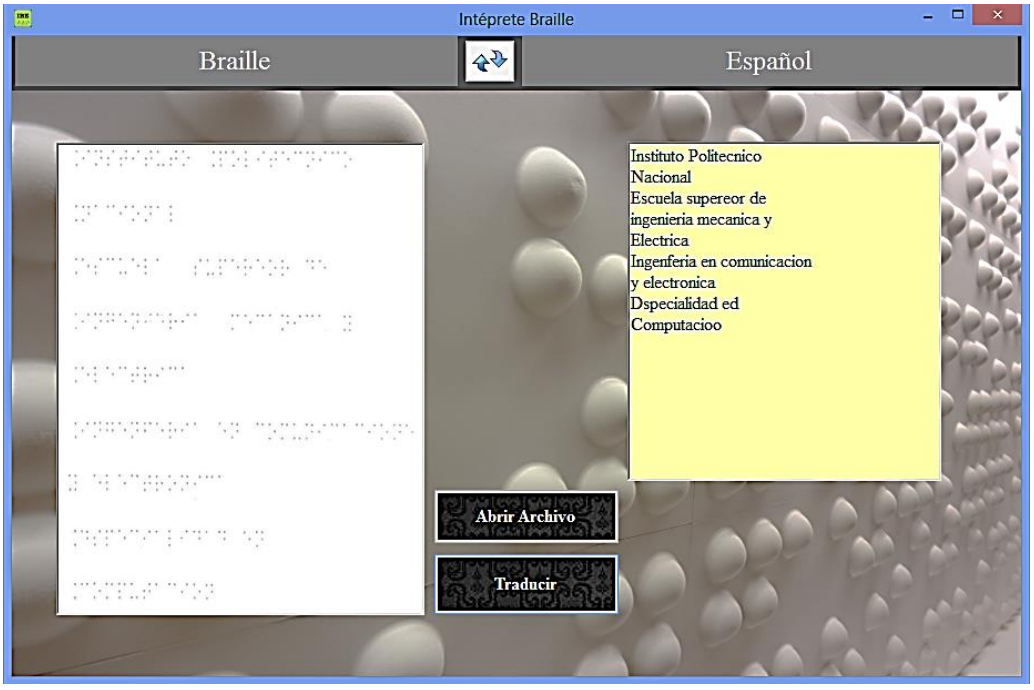


Figura 5.3.4 Traducción Braille- Español

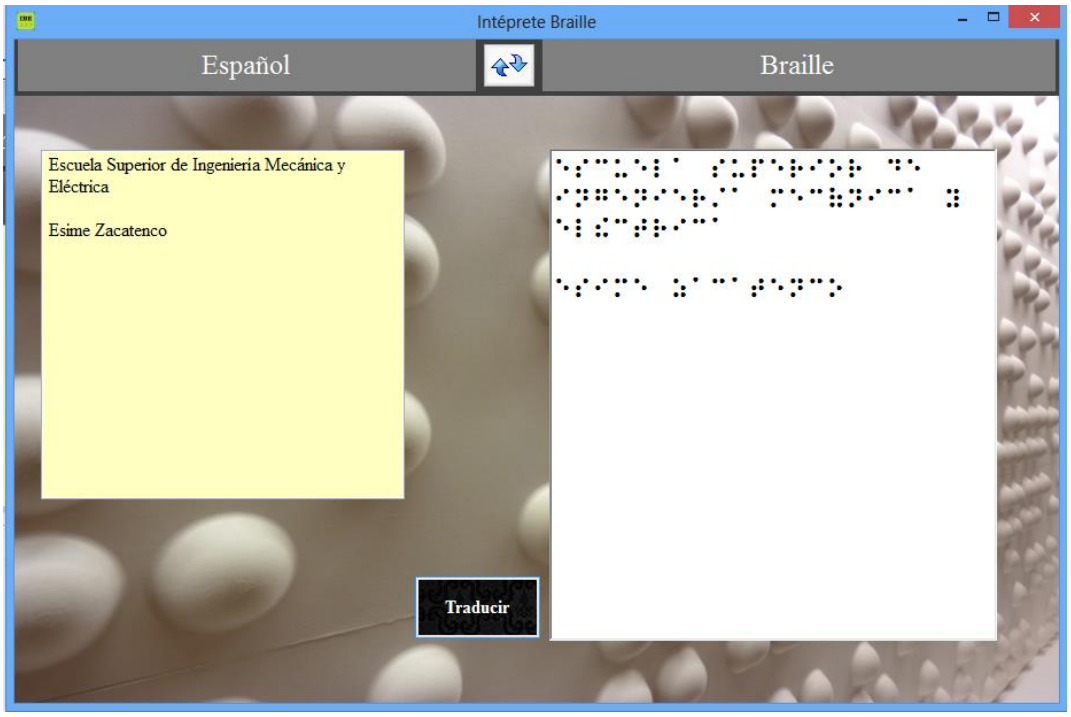


Figura 5.3.5 Traducción Español-Braille

5.3.1. Resultados Experimentales

Se realizaron diversas pruebas con diferentes documentos escritos en sistema Braille, elaborados a mano, mediante el uso de una regleta y punzón braille, se compararon los escritos originales con la traducción correspondiente obtenida a través del intérprete. Los resultados se muestran en la Tabla 5.3.1

Tabla 5.3.1. Porcentajes de recuperación del software para traducción Braille-Español.

Escrito Original	Escrito Obtenido del Traductor	Número de Carac.	Carac. erróneos	Porcentaje de efectividad
"Cada persona que pasa por Nuestra vida es unica Siempre deja un poco de si Y se lleva un poco de Nosotros Habra los que se llevaran mucho pero no habra de los Que no nos dejaran nada Esta es la prueba evidente De que dos almas no se Encuentran por casualidad"	Cada persona que pasa por Nuestra vida es unica Siempre deja un poco de si Y se lleva un feco de Nosotros Habra los que se llevaran xmucho pero no habra de los Que no nos dejaran nada Esta es la prueba evidente De que dos almas no e Encuentran por casualidad	249	4	98
"En ese instante oi que se	En ese instante oi ke s	217	7	96

Se quebraba algo en mi Interior por un instante Pense que era mi corazon pero no El corazon no se rompe Despues entendi que se me Habia roto la esperanza y Estaba saliendo por mis Ojos tibia y salada”	Se quebraba algo en mf Interior por un instante Pense que era mi corazon ypero no El corazon no se rompe Despues entendi que se me Habia roto la espehanoa n Estaba saliendo por mis Ojos tibia y salada			
“En un lugar de chapultpec A principios de siglo Pasado chapultepec era el Proyecto de parque privado De lujo que se abria a los Habitantes de la ciudad Fuentes diseadas por diego Rivera entre otros Entre gigantescos arboles Conocidos como ahuehuetes Que datan desde el periodo Azteca se encuentra la Fascinante fuente del quijote rodeada de asientos alejados en donde una Estatua del hidalgo ocupa el Lado opuesto a la del viejo Sancho panza en su una Cada cual descansando sobre Un pedestal con repisas interiores en la que se Pueden encontrar copia de Las historia de cervantes Entre otros”	an un gak e eamltpec A principo e sdbo Lasado chapkltepec era el Proyecto de parque privado De lujo que se abria a l Habitantes de la cuead Fuentes diseadas por dgo Rivera entre oto Entre gigantsc aebole Conocidos como ahuehuetes Que datan desde el lelodo Azteca se encuentra la Fascinante fuente del qjote rodeaea de aiento aoklejados n comde kma Estatua del hidalg ocupa el Lado opuesto a la del viejo Sancho panz en su ula Cada cual descaoando obre Un pedestal con repisas interikres en lai qke e Pueden encontrar cpia de Las historia de cervantes Entre otros	568	50	91
“No has vivido en vano Si le has dado alegria a Alguien No has vivido en vano Si has consolado al triste no has vivido en vano Si has defendido al debil Si has transmitido algun Conocimiento si has aliviado algun dolor Y si has amado a alguien tu vida tiene sentido has vivido en vano”	No has vivido en vano Si le has dado alegria a Alguien No has vivido en vano Si has consolado al triste xno has vivido en vano Si has defendido al debil Si has transmitido algun Conocimiento i has aliviado algun dolor Y si has amado a alguien u vida tiene sentido xhas vivido en vano	273	4	98
“La piedra el distraido tropezo Con ella el emprendedor construyo	La piedha mw distraido tropezo Con ella mel emprendedor construyo	178	7	96

Con ella para los campesinos sirvio De asiento En todos los casos la Difererencia no estuvo en la piedra si no en el Hombre”	Con ella ypara los campesinos sirvio De asiento mbtodos los casos la Difererencia no estuvo en ya piedra si no en el Hombre			
“Siento decirte que el libro que tienes en las manos es Extremadamente desagradable cuenta una triste historia Acerca de tres jovencitos Con muy mala suerte Aunque son encantadores y Muy listos Los hermanos baudelaire llevan una vida llena de dessgracias e infortunios”	Siento decirte que el libro gue tienes en las manos es Extremadamente desagradable mcuenta una triste historia Acerca de tres jovencitos Con muy mala suerte Aunque son encantadores y Muy listos Los hermanos baudelaire xlevan una vida llena de mfsgracias e infortunios	257	5	98
“Despues deun tiempo Aprenderas que el sol quema Si te expones demasiado Aceptaras incluso que las personas buenas podrian Herirte alguna vez y Necesitaras perdonarlas Descubriras que lleva tiempo construir confianza y apenas unos segundos Destruirla”	Despues deun tiempo Aprenderas que el sol quema Si te expones demasiado Aceptaras incluso que las ypersonas buenas podeian Herirte alguna vez y Necesitaras perdonarlas Descubriras que lleva tiempo construir confianza apenas unos segundos Destruirla	239	3	98

Con base en los resultados obtenidos mostrados en la Tabla 5.3.1, se obtuvo un porcentaje promedio de efectividad del método de traducción Braille-Español, el cual fue de 96.33%, este es aceptable para la comprensión del documento traducido.

Para el caso de traducción español-braille, se tiene una recuperación del 100% de todo el documento, esto debido a que en este modo no se hace un reconocimiento de imágenes para reconocer patrones que representen cada uno de los caracteres que componen dicho documento.
Cabe señalar que los resultados experimentales fueron realizados con condiciones óptimas de alineación de los documentos digitalizados. El margen de desviación en la alineación del documento es de ±2°. En caso de que la elaboración del documento tenga deficiencias que afecten esta alineación el software no realizará de forma correcta la traducción

.

Conclusiones

En este proyecto de tesis se desarrolló un software Interprete Braille- Español/ Español-Braille que utiliza del Modelo de las Memorias Asociativas Bidireccionales Alfa-Beta y el Procesamiento Digital de Imágenes para su funcionamiento.

De acuerdo a los objetivos planteados al inicio podemos concluir lo siguiente:

- Se elaboraron y digitalizaron diversos documentos escritos en sistema Braille con la utilización de una regleta y un punzón para la redacción de los documentos mientras que para la digitalización se utilizó un escáner.
- La imagen digital del documento Braille fue procesada utilizando un método propuesto en este trabajo que retoma algunas técnicas del procesamiento digital de imágenes permitiendo la extracción de los símbolos Braille contenidos en esta. Este método tiene la ventaja de no necesitar cálculos muy complejos por lo que su implementación es rápida y sencilla.
- Con la implementación del modelo matemático de las Memoria Asociativas Bidireccionales Alfa-Beta se logró que el software permita recuperar en un 96.33% cuando se realiza la traducción Braille-Español, mientras que cuando se efectúa la traducción Español-Braille se tiene una recuperación del 100% de acuerdo a los resultados experimentales.

Finalmente con base en el proyecto terminado se puede asegurar que el Intérprete Braille-Español/Español-Braille es una herramienta eficaz, ya que tiene un alto porcentaje de recuperación correcta, y su sencilla pero funcional interfaz gráfica lo posicionan como un software amigable para cualquier usuario.

ANEXOS

ANEXO 1.

Evaluación Económica del Proyecto

El fin de la creación de este proyecto es contribuir a una mejor integración de personas invidentes en el ámbito educativo, este proyecto tiene el fin de ser una herramienta de interpretación escrita del sistema braille que brinde una gran ayuda tanto a docentes como a personas invidentes en vías de aprendizaje de este sistema.
El trabajo realizado no busca una remuneración como tal sino tener un impacto social que mejore las vías de aprendizaje y la comprensión escrita para invidentes.

El producto va dirigido a instituciones académicas, que brinden enseñanza a personas invidentes, no necesariamente escuelas exclusivas para personas con ceguera, sino escuelas públicas o privadas que cuenten con alumnos o personas con esta condición y docentes que no conozcan el sistema braille. Así también, está dirigido a personas que convivan con personas con esta condición y /o que busquen aprender este sistema de escritura.

En México existen algunas organizaciones como "El Comité Internacional Pro-Ciegos I.A.P." y "El Centro de Estudios para Invidentes A.C. CEIAC" entre otras, que tratan de ofrecer las herramientas necesarias a personas con discapacidad visual a través de recursos económicos, materiales y humanos que les permitan su integración social.

Algunas de las instituciones más conocidas que forman parte de estas organizaciones y que se encuentran dentro de la República Mexicana. Ver Tabla 6.1.

Tabla 6.1. Centros de Estudios y Apoyo para Personas Invidentes en México, D.F.

Nombre de la Institución	Ubicación
Amigos del Estudiante Invidentes, IAP	Distrito Federal
Asociación Banco de Ojos Lions International I.A.P.	Tlalnepantla, México
Asociación Centro de Rehabilitación para Ciegos, I.A.P.	Cuernavaca, Morelos
Escuela Nacional de Ciegos IAP	México, D.F.
Escuela de Educación Especial No. 8 para Transtornos Visuales	Cuernavaca, Morelos
Asociación Centro de Rehabilitación para ciegos	Tepoztlán, Morelos
Centro de Rehabilitación para ciegos y débiles visuales	Cuauhtémoc, Distrito Federal
Asociación de Ciegos Cordobeses, A.C.	Córdoba, Veracruz
Asociación de Ciegos de Tuxtla Gutiérrez, A.C.	Tuxtla Gutiérrez, Chiapas

Asociación de Invidentes Santa Lucia, A.C.	San Miguel de Allende, Guanajuato
Asociación Ignacio Trigueros, I.A.P.	Miguel Hidalgo, México, D.F.
Asociación Mexicana de Retinitis Pigmentosa y Enfermedades de la Retina, A.B.P.	Monterrey, Nuevo León
Asociación de apoyo al invidentes y débil visual	Garza García, N.L.
Asociación Mexicana de Usuarios en pro del perro guía, A.C.	México, D.F
Escuela Asociación deportiva cultural de ciegos y débiles visuales D.F.	México, D.F
Asociación Nacional de Invidentes Comerciantes, A.C.	Cuauhtémoc, México, D.F
Asociación Oftalmológica de León, A.C.	León, Guanajuato
Asociación para evitar la ceguera en México, I.A.P.	Coyoacán, México, D.F
Asociación pro desarrollo e integración del niño ciego, A.C.	Coyoacán, México, D.F.
Asociación pro educación y rehabilitación de ciegos y débiles visuales, A.C.	Venustiano Carranza, México, D.F.
Asociación pro estudiantes ciegos y débiles visuales, A.C. ASPEC	Acapulco, Guerrero
Asociación pro invidentes de Monclova, A.C.	Monclova, Coahuila
Banco de ojos del Estado de Guanajuato, A.C.	León, Guanajuato
Biblioteca México, Sala de Invidentes	Cuauhtémoc, México, D.F.
Centro de Capacitación para Invidentes (Tijuana)	Tijuana, Baja California Norte
Centro de Capacitación para Invidentes, A.C. (Durango)	Durango, Durango

Fuente: Página Web de Libre Acceso, A.C.

Los beneficios directos de nuestro proyecto en el ámbito social estarían dados por la cantidad de individuos que resultarían beneficiados con el Software Interprete Braille-Español/Español-Braille para ello ponemos como ejemplo que si nuestro software pudiese ser implementado en la "Escuela Nacional de Ciegos IAP" la cual cuenta con una población estudiantil de aproximadamente 189 estudiantes, entonces podríamos darnos cuenta que tendríamos a 189 individuos beneficiados con la herramienta lo que a final de cuentas cumple con nuestro objetivo.

Cabe destacar que a pesar de que existen algunos proyectos que implican la conversión del sistema Braille al idioma Español son muy pocos los que realmente se utilizan comercialmente, y generalmente éstos son enfocados principalmente al área de traducción de libros en masa y no para uso individual.

En la actualidad hay disponibles en internet una serie de traductores online que permiten hacer únicamente la conversión de textos en **Español a Braille** (Ver Tabla 6.2). Pero ninguno de ellos te

brinda la posibilidad de reconocer los signos braille a través de una imagen digital como lo hace el Interprete Braille-Español que proponemos en este proyecto de tesis.

Tabla 6.2. Centros de Estudios y Apoyo para Personas Invidentes en México, D.F.

Nombre del Software	Descripción
Software Traductor a Braille Duxbury	Sistemas de Duxbury es líder mundial en software para Braille. El Duxbury Braille Translator (DBT) y MegaDots, son utilizados por prácticamente todos los principales editores del mundo en braille [11].
Alfabeto Braille en Línea	Es una aplicación desarrollada usando tecnología JAVA que permite a los usuarios escribir el texto deseado y convertirlo a braille, no usa un algoritmo muy complejo ya solamente se basa en hacer comparaciones[12].
Traducir Braille Simple	Es una aplicación que permite a los usuarios escribir el texto deseado y convertirlo a braille, posee una interfaz gráfica muy simple [14].
Dots	Un traductor de Braille para GNOME que permite "traducir" un documento (ODT, PDF o HTML) en una representación Braille en el ordenador de forma que podamos enviar ese código Braille a una impresora Braille[15].
Top Braille	El "Top Braille", un aparato que permite traducir instantáneamente en braille los textos impresos [16].

Costo del Proyecto

Tabla 6.3. Costos por Hora de Ingeniería.

Tareas Realizadas	No. De Horas	Precio por Hora	Costo Total
Análisis e Investigación	250	$ 60.00	$ 15,000.00
Redacción del Documento	200	$ 60.00	$ 12,000.00
Programación	300	$ 60.00	$ 18,000.00
Implementación y Depuración del Software	200	$ 60.00	$ 12,000.00
Pruebas Funcionales y Correcciones	100	$ 60.00	$ 6,000.00
		Total=	$ 63,000.00

Fuente: Página Web de Empleo Gobierno del D.F.

Tabla 6.4. Costos por Material y Equipo.

Tareas Realizadas	Descripción	Cant.	Precio Unitario	Precio Total (MXN)
MATLAB r2010a	Versión Estudiantil [26]	1	USD 89.00	$ 15,000.00
Visual Studio Professional 2012	Licencia Anual [27]	1	$ 6,611.00	$ 6,611.00
Computadora	Laptop [28]	1	$ 8000.00	$ 8000.00
Regleta y Punzón	Braille [25]	1	$ 100.00	$ 100.00
Hojas Especiales para Braille	Braille [25]	10	$ 20.00	$ 20.00
Impresiones	Avance de Tesis	100	$ 1.00	$ 100.00
Otros Gastos	Incluye: Internet, Luz Eléctrica, Escáner, etc.	-	$ 1,500.00	$ 1,500.00
			Total=	$ 17,517.00

Fuente: Página Web de Mathworks Company, Microsoft Company, Toshiba Company, Instituto para Ciegos.

Tabla 6.5. Costos Totales.

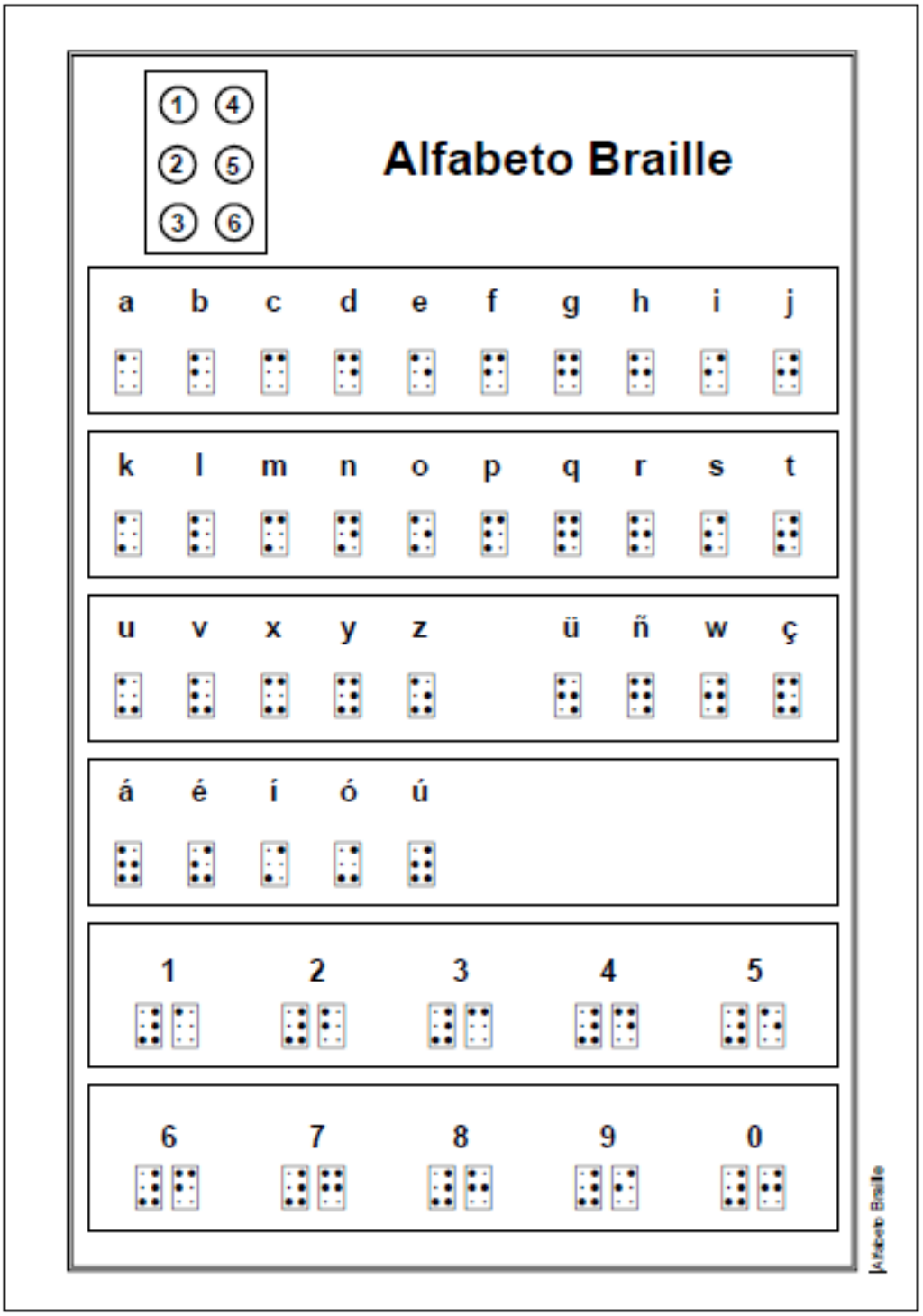
Concepto	Costo
Costos por Hora de Ingeniería.	$ 63,000.00
Costos por Material y Equipo.	$ 17,517.00
Total=	$ 80,517.00

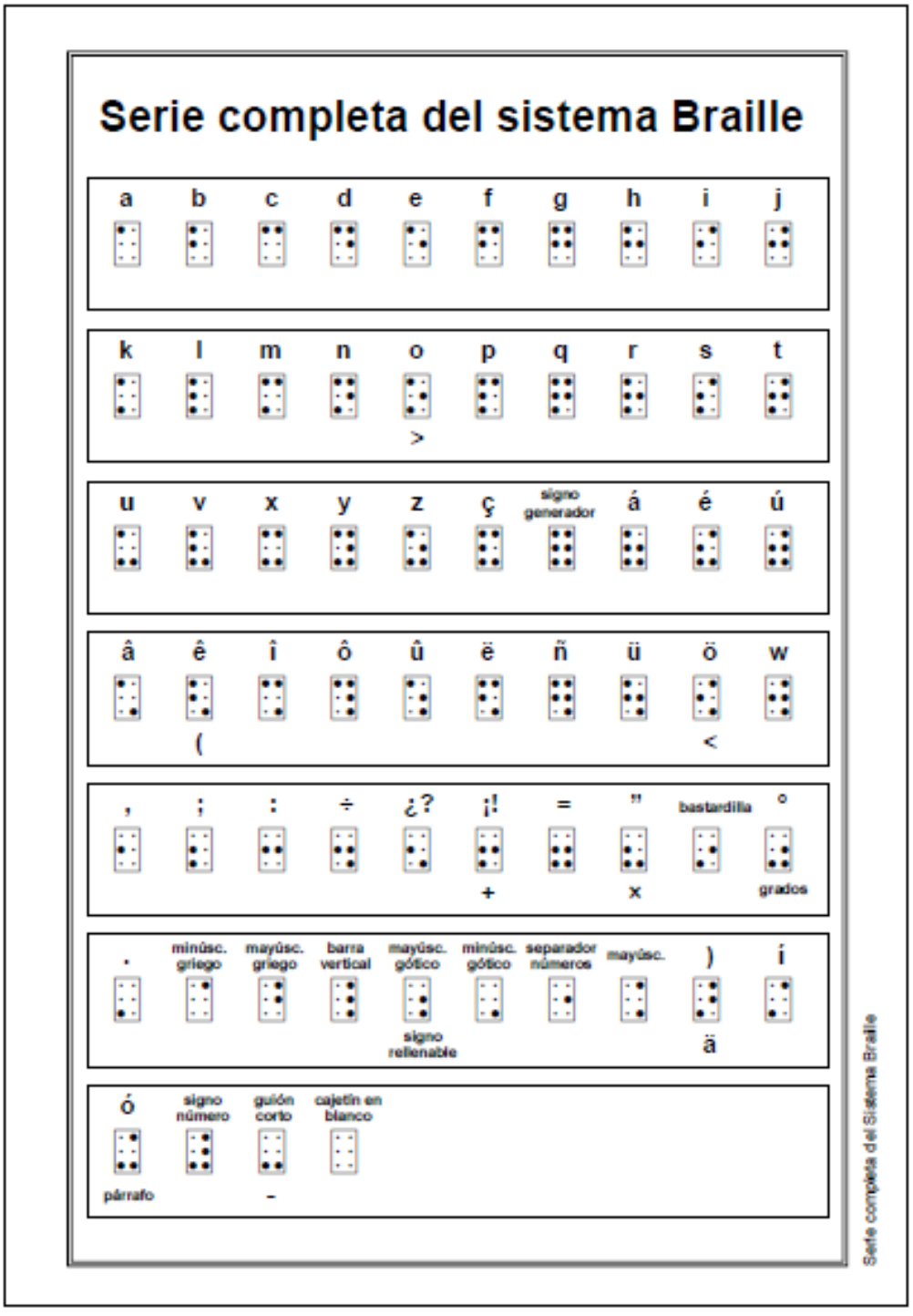
Ganancia del Proyecto

Haciendo énfasis en que el objetivo general de este proyecto de tesis es "Diseñar e implementar un Intérprete Braille-Español/Español-Braille utilizando el Modelo de las Memorias Asociativas Bidireccionales Alfa-Beta." Como primer punto de vista la ganancia que se obtiene al concluir este proyecto de tesis para obtener el título de Ingeniero en Comunicaciones y Electrónica recae meramente en el hecho de que este proyecto nos brinda la oportunidad de implementar los conocimientos adquiridos durante nuestra trayectoria a lo largo de la carrera y así poder obtener un título universitario. Sin embargo el otro punto de vista que tenemos para definir la ganancia de nuestro proyecto de tesis es indudablemente el beneficio social que representa. Al ser este un Proyecto Social implica que la ganancia está en función de las personas que puedan verse beneficiadas al utilizar nuestro Software.

ANEXO 2.

Alfabeto Braille-Español




Serie completa del sistema Braille
a b c d e f g h i j
k l m n o p q r s t
>
u v x y z ç signo generador á é ú
â ê î ô û ë ñ ü ö w
(
<
, ; : ÷ ¿? ¡! = " bastardilla °
+ x grados
. minúsc. griego mayúsc. griego barra vertical mayúsc. gótico minúsc. gótico separador números mayúsc.) í
signo relienable ä
ó signo número guión corto cajetín en blanco
párrafo -
Serie completa del Sistema Braille

ANEXO 3.

Código del Software en Visual Studio 2012

```
Globales.Inicializar();//INICIALIZANDO VARIABLES
Globales.Aprende();
```

```
//CARGAR LA IMAGEN A TRADUCIR
private void Abrir_Click(object sender, EventArgs e)
{
  OpenFileDialog ventanadialogo = new OpenFileDialog();
  ventanadialogo.Filter = "Archivos JPG(.jpg)|*.jpg" + "|Archivos BMP(.bmp)|*.bmp";
  if (ventanadialogo.ShowDialog() == DialogResult.OK)
  {
   pictureBoxoriginal.Image = Image.FromFile(ventanadialogo.FileName);
  }
}
```

```
//TRADUCIENDO LA IMAGEN
private void Traducir_Click(object sender, EventArgs e)
{
  imagen = new Bitmap(pictureBoxoriginal.Image);
  esc_grises(imagen);
  binario(imagen, .95);
  imagenoriginal = copiarimagen(imagen);
  detecta_renglones(imagen);
  ubica_renglones(imagen);
  for (int reng = 0; reng < renglones.Count-4; reng=reng+6)
  {
   selecciona_renglon(reng, reng+5);
   selecciona_renglon(reng,reng + 5);
   detecta_columnas(imagenunrenglon);
   ubica_columnas(imagenunrenglon);
   ubica_casillas();
   for (int casi = 0; casi < casillas.Count - 1; casi = casi + 2)
   {
    mayus = 0;
    selecciona_casilla(casi, casi+1);
    crea_vector(casi,reng);
    if (vector[0] == 0 && vector[1] == 0 && vector[2] == 0&& vector[3]
       == 1 && vector[4] == 0 && vector[5] == 1)
    {
     mayus =32;
     casi=casi+2;
     selecciona_casilla(casi, casi + 1);
     crea_vector(casi, reng);
    }
    textBoxtraducción.Text = textBoxtraducción.Text +
    obten_caracter(vector).ToString();
   }
   textBoxtraducción.Text = textBoxtraducción.Text +
                            System.Environment.NewLine;
  }
}
```

```
public static void Aprende()
```

```
{
 //LLENANDO MATRIZ DE ENTRADAS
 for (int i = 0; i < entradasx.Length; i++)
  {
   for (int j = 0; j < fil; j++)
   {
    entradasxt[j, i] = entradasx[i][j];
   }
  }
   //CREANDO EL LINEAR ASSOCIATOR
   for (int i = 0; i < fily; i++)
   {
    for (int j = 0; j < coly; j++)
    {
     salidasyt[i, j] = salidasy[j][i];
    }
   }
   //CREAR VECTORES ONE_HOT y ZERO_HOT
   for (int i = 0; i < col; i++)
   {
    for (int j = 0; j < col; j++)
    {
     if (j == i) { one_hot[j, i] = 1; zero_hot[j, i] = 0; }
     else { one_hot[j, i] = 0; zero_hot[j, i] = 1; }
    }
   }
   //CONCATENAR VECTORES CON ENTRADAS ONE HOT
   for (int i = 0; i < fil; i++)
   {
    for (int j = 0; j < entradasx.Length; j++)
    {
    EntradasOneHot[i, j] = entradasxt[i, j];
    EntradasZeroHot[i, j] = entradasxt[i, j];
    }
   }
   for (int i = 0; i < col; i++)
   {
    for (int j = 0; j < col; j++)
    {
    EntradasOneHot[j + fil, i] = one_hot[j, i];
    EntradasZeroHot[j + fil, i] = zero_hot[j, i];
    }
   }
   //OBTENER LA MATRIZ ALFA--->PARA MIN
   for (int k = 0; k < entradasx.Length; k++)
   {
    for (int i = 0; i < fil + entradasx.Length; i++)
    {
     for (int j = 0; j < fil + entradasx.Length; j++)
     {
      if (EntradasZeroHot[i, k] == 0 && EntradasZeroHot[j, k] == 0)
      { alfa[i, j, k] = 1; }
      if (EntradasZeroHot[i, k] == 0 && EntradasZeroHot[j, k] == 1)
      { alfa[i, j, k] = 0; }

      if (EntradasZeroHot[i, k] == 1 && EntradasZeroHot[j, k] == 0)
      { alfa[i, j, k] = 2; }
      if (EntradasZeroHot[i, k] == 1 && EntradasZeroHot[j, k] == 1)
      { alfa[i, j, k] = 1; }
```

```
  }
 }
}
//OBTENER MIN
int aux = 0;
for (int i = 0; i < fil + entradasx.Length; i++)
{
 for (int j = 0; j < fil + entradasx.Length; j++)
 {
  for (int k = 0; k < entradasx.Length; k++)
  {
   if (k == 0) { aux = alfa[i, j, k]; }
   if (k != 0)
   {
    if (alfa[i, j, k] < aux)
    {
     aux = alfa[i, j, k];
    }
   }
  }
  MIN[i, j] = aux;aux = 0;
 }
}
//OBTENER MATRIZ ALFA2--->PARA MAX
for (int k = 0; k < entradasx.Length; k++)
{
 for (int i = 0; i < fil + entradasx.Length; i++)
 {
  for (int j = 0; j < fil + entradasx.Length; j++)
   {
    if (EntradasOneHot[i, k] == 0 && EntradasOneHot[j, k] == 0)
    { alfa2[i, j, k] = 1; }
    if (EntradasOneHot[i, k] == 0 && EntradasOneHot[j, k] == 1)
    { alfa2[i, j, k] = 0; }
    if (EntradasOneHot[i, k] == 1 && EntradasOneHot[j, k] == 0)
    { alfa2[i, j, k] = 2; }
    if (EntradasOneHot[i, k] == 1 && EntradasOneHot[j, k] == 1)
    { alfa2[i, j, k] = 1; }
   }
  }
 }
 //OBTENER MAX
 int aux2 = 0;
 for (int i = 0; i < fil + entradasx.Length; i++)
 {
  for (int j = 0; j < fil + entradasx.Length; j++)
  {
   for (int k = 0; k < entradasx.Length; k++)
   {
    if (k == 0){ aux2 = alfa2[i, j, k]; }
    if (k != 0)
     {
      if (alfa2[i, j, k] > aux2)
      {
       aux2 = alfa2[i, j, k];
      }
     }
    }
    MAX[i, j] = aux2;aux2 = 0;
```

```
       }
      }
     }
public static int[] Recupera(int[] VecEntrada)
{
 int[] VecSalida = new int[Abin.Length];
 int[] ColaOnes = new int[entradasx.Length];
 int[] ColaZeros = new int[entradasx.Length];
 int[] F = new int[VecEntrada.Length + ColaOnes.Length];
 int[] G = new int[VecEntrada.Length + ColaZeros.Length];
 int[] R = new int[VecEntrada.Length + ColaOnes.Length];
 int[] S = new int[VecEntrada.Length + ColaZeros.Length];
 int[] r = new int[ColaOnes.Length];
 int[] s = new int[ColaZeros.Length];
 int[] t = new int[ColaZeros.Length];
 //LLENANDO EL VECTOR DE EXPANSION F y G
 Array.Copy(VecEntrada, F, VecEntrada.Length);
 Array.Copy(VecEntrada, G, VecEntrada.Length);
 for (int i = 0; i < ColaOnes.Length; i++)
 {
  ColaOnes[i] = 1; ColaZeros[i] = 0;
 }
 for (int i = VecEntrada.Length; i < F.Length; i++)
 {
  F[i] = ColaOnes[i - VecEntrada.Length]; G[i] = ColaZeros[i - VecEntrada.Length];
 }
 //RECUPERA CON MAX y MIN
 for (int i = 0; i < fil + entradasx.Length; i++)
 {
  for (int j = 0; j < fil + entradasx.Length; j++)
  {
   if (MAX[i, j] == 0 && F[j] == 0){ RecuMax[i, j] = 0; }
   if (MAX[i, j] == 0 && F[j] == 1) { RecuMax[i, j] = 0;}
   if (MAX[i, j] == 1 && F[j] == 0){ RecuMax[i, j] = 0; }
   if (MAX[i, j] == 1 && F[j] == 1){ RecuMax[i, j] = 1; }
   if (MAX[i, j] == 2 && F[j] == 0){ RecuMax[i, j] = 1; }
   if (MAX[i, j] == 2 && F[j] == 1){ RecuMax[i, j] = 1; }
   if (MIN[i, j] == 0 && G[j] == 0){ RecuMin[i, j] = 0; }
   if (MIN[i, j] == 0 && G[j] == 1){ RecuMin[i, j] = 0; }
   if (MIN[i, j] == 1 && G[j] == 0){ RecuMin[i, j] = 0; }
   if (MIN[i, j] == 1 && G[j] == 1){ RecuMin[i, j] = 1; }
   if (MIN[i, j] == 2 && G[j] == 0){ RecuMin[i, j] = 1; }
   if (MIN[i, j] == 2 && G[j] == 1){ RecuMin[i, j] = 1; }
   }
  }
  //LLENAR EL VECTOR R y S
  int mini = 0;
  int maxi = 0;
  for (int i = 0; i < fil + entradasx.Length; i++)
  {
   for (int j = 0; j < fil + entradasx.Length; j++)
   {
    if (j == 0) { mini = RecuMax[i, j]; maxi = RecuMin[i, j]; }
    if (j != 0)
    {
     if (RecuMax[i, j] < mini){mini = RecuMax[i, j];}
     if (RecuMin[i, j] > maxi){maxi = RecuMin[i, j];}
    }
```

```
   }
   R[i] = mini;mini = 0;S[i] = maxi;maxi = 0;
  }
  //Obtener vector r y s
  for (int j = 0; j < r.Length; j++)
  {
   r[j] = R[j + VecEntrada.Length];
   s[j] = S[j + VecEntrada.Length];
  }
  //APLICANDO AND
  for (int j = 0; j < ColaZeros.Length; j++)
  {
   if (r[j] == 0 && s[j] == 0) { t[j] = 0; }
   if (r[j] == 0 && s[j] == 1) { t[j] = 0; }
   if (r[j] == 1 && s[j] == 0) { t[j] = 1; }
   if (r[j] == 1 && s[j] == 1) { t[j] = 0; }
  }
  //MULTIPLICA LINER ASSOCIATOR Y VECTOR t
  int sum = 0;
  for (int i = 0; i < fily; i++)
  {
   for (int j = 0; j < coly; j++)
   {
    sum = sum + salidasyt[i, j] * t[j];
   }
   VecSalida[i] = sum;
   sum = 0;
   }
   return VecSalida;
  }
private Bitmap detecta_renglones(Bitmap imagen)
{
 int q;
 for (int j = 0; j < imagen.Height; j++)
 {
  q = 0;
  for (int i = 0; i < imagen.Width; i++)
  {
   q = q + imagen.GetPixel(i, j).B;
  }
  if (q <= 3 * 255)
  {
   for (int k = 0; k < imagen.Width; k++)
   {
    imagen.SetPixel(k, j, Color.White);
       }
      }
     }
     return (imagen);
    }

private List<int> ubica_renglones(Bitmap imagen)
{
 int inicio, final;
 inicio = 0;
 final = 0;
 for (int j = 0; j < imagen.Height; j++)
 {
```

```
  if (imagen.GetPixel(1, j).B != 0) {inicio = j+1;}
  if (imagen.GetPixel(1, j).B == 0)
  {
   while (imagen.GetPixel(1, j).B == 0)
   {
    j++;
   }
   final = j - 1;
  }
  if (inicio <= final)
  {
   renglones.Add(inicio);
   renglones.Add(final);
   inicio = 0;
   final = 0;
  }
 }
 return (renglones);
}

private Bitmap selecciona_renglon(int inir,int finr)
{
 iniciorenglon =renglones[inir];
 finrenglon = renglones[finr];
 anchorenglon = finrenglon - iniciorenglon;
 imagenunrenglon = new Bitmap(pictureBoxoriginal.Image, new Size(imagen.Width,
       anchorenglon + 1));
 for (int j = iniciorenglon; j <= finrenglon; j++)
 {
  for (int i = 0; i < imagen.Width; i++)
  {
   imagenunrenglon.SetPixel(i, j - iniciorenglon, imagenoriginal.GetPixel(i, j));
  }
 }
 return (imagenunrenglon);
}

private Bitmap detecta_columnas(Bitmap imagenunrenglon)
{
 int q;
 for (int i = 0; i < imagenunrenglon.Width; i++)
 {
  q = 0;
                for (int j = 0; j < imagenunrenglon.Height; j++)
                {
                    q = q + imagenunrenglon.GetPixel(i, j).B;
                }
                if (q >= 1 * 255)
                {
                    for (int k = 0; k < imagenunrenglon.Height; k++)
                    {
                        imagenunrenglon.SetPixel(i, k, Color.White);
                    }
                }
            }
            return (imagenunrenglon);
        }
private void ubica_columnas(Bitmap imagenunrenglon)
        {
```

```
            int inicio, final;
            columnas.Clear();
            inicio = 0;
            final = 0;
            for (int i = 0; i < imagenunrenglon.Width; i++)
            {
                if (imagenunrenglon.GetPixel(i, 1).B == 0)
                {
                    inicio = i + 1;
                }
                if (imagenunrenglon.GetPixel(i, 1).B != 0)
                {
                    while (imagenunrenglon.GetPixel(i, 1).B != 0)
                    {
                        i++;
                    }
                    final = i - 1;
                }
                if (inicio <= final)
                {
                    columnas.Add(inicio);
                    columnas.Add(final);
                    inicio = 0;
                    final = 0;
                }
            }
        }

private void ubica_casillas()
        {
            double rangoi, rangof;
            rangoi = 0;
            rangof = 0;
            ancho = 0;
            for (int i = 0; i < columnas.Count - 1; i = i + 2)
            {
                if (columnas[i + 1] - columnas[i] + 1 > ancho)
                    ancho = columnas[i + 1] - columnas[i] + 1;
            }
            casillas.Clear();
            for (int i = 0; i < columnas.Count - 3; i = i + 2)
            {
                rangoi = columnas[i + 1];
                rangof = rangoi + 2 * ancho;
                if (columnas[i + 2] < rangof)
                {
                    casillas.Add(columnas[i]);
                    casillas.Add(columnas[i + 3]);
                    i = i + 2;
                }
                else
                {
                    if (columnas[i + 2] < rangof + ancho)
                    {
                        casillas.Add(columnas[i] - 1 * (int)(ancho*1.5));
                        casillas.Add(columnas[i + 1]);
                    }
                    else
                    {
```

```
                        casillas.Add(columnas[i]);
                        casillas.Add(columnas[i + 1] + 1 * (int)(ancho*1.5));
                    }
                }
            }
            if (casillas[casillas.Count - 1] != columnas[columnas.Count - 1])
            {
                casillas.Add(columnas[columnas.Count-2]);
                casillas.Add(columnas[columnas.Count - 1] + 1 * (int)(ancho*1.5));
            }
            anterior =casillas[0];
        }

private Bitmap selecciona_casilla(int inic, int finc)
        {
            iniciocelda =casillas[inic];
            fincelda =casillas[finc];
            anchocelda = fincelda - iniciocelda;
            imagencasilla = new Bitmap(pictureBoxoriginal.Image, new
Size((int)anchocelda, anchorenglon+1));
            for (int j = iniciorenglon; j <= finrenglon; j++)
            {
                for (int i = (int)iniciocelda; i < fincelda; i++)
                {
                    imagencasilla.SetPixel(i -(int) iniciocelda, j - iniciorenglon,
imagenoriginal.GetPixel(i, j));
                }
            }
            return (imagencasilla);
        }

private int[] crea_vector(int casi,int reng)
        {
            int suma;
            vector = new int[6];
            vector[0] = 0;
            vector[1] = 0;
            vector[2] = 0;
            vector[3] = 0;
            vector[4] = 0;
            vector[5] = 0;
            if (casillas[casi] - anterior > 6 * ancho)
                textBoxtraducción.Text = textBoxtraducción.Text + " ";
            anterior = casillas[casi];
            for (int b = 0; b < 2; b++)
            {
                for (int a = 0; a < 3; a ++)
                {
                    suma = 0;
                    for (int i = b*imagencasilla.Width/2; i <(b+1)* imagencasilla.Width /
2; i++)
                    {
                        for (int j = renglones[reng + 2*a] - renglones[reng]; j <
renglones[reng + 2*a + 1] - renglones[reng]; j++)
                        {
                            suma = suma + imagencasilla.GetPixel(i, j).B;
                        }
                    }
                    if (suma >255*2)
```

```
                        vector[a + b * 3] = 1;
                }
            }
            return (vector);
        }

private char obten_caracter(int[] VecEntrada)
        {
            Globales.VecEntrada = VecEntrada;
            int[] VecSalida = new int[Globales.Abin.Length];
            string CadBin = "";
            char caracter;
            VecSalida=Globales.Recupera(Globales.VecEntrada);
            for (int i = 0; i <Globales.fily; i++)
                CadBin += VecSalida[i];
            byte NumBin = Convert.ToByte(CadBin, 2);
            caracter = Convert.ToChar(NumBin-mayus);
            return (caracter);
        }
```

Referencias

[1] Mackenzie, S.C. (1954). La escritura Braille en el mundo. París: Presidente del consejo de la Unesco.

[2] Navarro Saad, M. (1998). El Sistema Braille. Eureka. Diciembre, 1998. [Internet] Disponible en: <http://www.uaq.mx/ingenieria/publicaciones/eureka/n13/en1307.pdf> [Acceso el 17 de Marzo de 2013].

[3] Rodríguez Arredondo, D. (2001). Estudio exploratorio sobre la escritura de palabras homófonas en español, en Braille grado1 y Braille grado 2. UNAM: Posgrado de Lingüística. Gobierno Federal.

[4] Huertas, J.A., Rosa, A. (1988). Peculiaridades de la lectura táctil del Braille: un estudio empírico. Infancia y Aprendizaje: Journal for the Study of Education and Development. No.41. Pág.79-94. [Internet] Disponible en: < http://dialnet.unirioja.es/servlet/articulo?codigo=48294> [Acceso el 16 de Marzo de 2013]

[5] Image processing techniques to perform an autonomous system to translate relief Braille into black-ink, called: LectoBraille. [Internet]
Disponible en:
<http://bauhaus.ece.curtin.edu.au/~iain/PhD%20BU/A_Phd%20docs/To%20read/Accessibility%20info/Xplore%20with%20assistive%20technology/+LectoBraille%20.pdf>

[6] Optical recognition of Braille writing using standard equipment. [Internet] Disponible en: <http://bauhaus.ece.curtin.edu.au/~iain/PhD%20BU/A_Phd%20docs/To%20read/Accessibility%20info/Research/Braille_Articles/OCR%20of%20Braille.pdf>

[7] A portable device for optically recognizing Braille. [Internet] Disponible en: <http://cucat.org/publications/device_Braille_I.pdf>

[8] A neural network hybrid model for an optical Braille recognitor. [Internet] Disponible en: <http://www.wseas.us/e-library/conferences/skiathos2002/papers/447-320.pdf>

[9] Hassoun, M. H. 1993, Assosiative Neural Memories, Oxford University Press, New York.

[10] A Software Algorithm Prototype for Optical Recognition of Embossed Braille. [Internet] Disponible en: <http://homepages.engineering.auckland.ac.nz/~wabd002/Braille_camera_ready2.pdf>

[11] Software traductor a Braille Duxbury. [Internet] Disponible en:
<http://www.tecno-ayudas.com.ar/productos_software_traductor_braille_duxbury.html>

[12] Software Alfabeto Braille en Línea. [Internet] Disponible en:
<http://www.fbu.edu.uy/alfabeto/alfabeto-online.htm>

[13] Software Braille Translator. [Internet] Disponible en:
< http://www.brailletranslator.org/es.html>

[14] Software traductor Braille Simple. [Internet] Disponible en:
<http://libbraille.org/translator.php?src=hola&table=basic6>

[15] Top Braille. [Internet] Disponible en: <http://www.elmundo.es/elmundosalud/2007/07/09/medicina/1183965035.html>

[16] Dots: traductor Braille. [Internet] Disponible en: <http://www.muylinux.com/2010/10/11/dots-un-traductor-de-braille-para-gnome/>

[17] Acevedo Mosqueda María Elena, Yáñez Márquez Cornelio, López Yáñez Itzamá (2007). A New Model of BAM: Alpha-Beta Bidirectional Associative Memories [Internet] Disponible en: <http://www.academypublisher.com/jcp/vol02/no04/jcp02044956.html>

[18] Acevedo Mosqueda María Elena (2006). Memorias Asociativas Bidireccionales Alfa-Beta. Trabajo de Grado, Doctorado en Ciencias de la Computación, Instituto Politécnico Nacional, México, D.F.

[19] Olivares Montes Teresa, Cuenca Castillo Pedro Ángel (2004). La Morfología Matemática en el Tratamiento Digital de Imágenes. Escuela Universitaria Politécnica de Albacete. Departamento de Informática.

[20] Cárdenas Hidalgo Paul, Flores Vargas José Alfredo, López Zavaleta Jaime, Martínez Moreno Pablo (2009). Diseño de un Sistema de Reconocimiento de Placas utilizando MATLAB. Trabajo de Grado, Ingeniero en Comunicaciones y Electrónica, Instituto Politécnico Nacional, ESIME ZAC., México, D.F.

[21] Sucar L. Enrique, Gómez Giovani (2005). Visión Computacional. Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México.

[22] De la Rosa Flores R. (2007). Procesamiento de Imágenes Digitales. Instituto Tecnológico de Puebla. Facultad de ciencias de la computación, Puebla, México.

[23] Tinoco de la Luz Raúl (2009). Sistema de Reconocimiento Facial por medio de Eigenfaces y Redes Neuronales. Trabajo de Grado, Ingeniero en Comunicaciones y Electrónica, Instituto Politécnico Nacional, ESIME CULHUACAN., México, D.F.

[24] Lozada Mendez Cynthia Arlette, Luis Pineda Loyda Florencia (2007). Procesamiento Digital de Imágenes Biometricas Aplicado al Control de Sistemas de Seguridad. Trabajo de Grado, Ingeniero en Comunicaciones y Electrónica, Instituto Politécnico Nacional, México, D.F.

[25] Página Web del Instituto para Ciegos. [Internet] Disponible en:<http://www.institutoparaciegos.org/>

[26] Página Web de Mathworks Company. [Internet] Disponible en: <http://www.mathworks.com/academia/student_version/>

[27] Página Web de Microsoft Company. [Internet] Disponible en: <http://www.microsoft.com/visualstudio/esn/buy/>

[28] Página Web de Toshiba Company. [Internet] Disponible en:<http://www.toshiba.com/tai/>

[29] Regular Feature Extraction for Recognition of Braille. [Internet] Disponible en: <http://bauhaus.ece.curtin.edu.au/~iain/PhD%20BU/A_Phd%20docs/To%20read/Braille_Articles/feature%20extraction%20of%20braille.pdf>