Un algoritmo evolutivo aplicado a la optimización del reconocimiento de caracteres mediante procesamiento de imágenes

José Pablo González, Diego Segovia

Proyecto de Grado de la carrera de Ingeniería en Computación
Instituto de Computación – Facultad de Ingeniería – Universidad de la República
Montevideo, República Oriental del Uruguay
Diciembre de 2010

Resumen

El reconocimiento óptico de caracteres (OCR, por su sigla en inglés), tiene como objetivo extraer de una imagen los caracteres que componen un texto para almacenarlos en un formato con el cual puedan interactuar programas de procesamiento de texto.

Hoy en día la utilización de medios digitales para el reconocimiento óptico de caracteres ha logrado avances sustanciales y en materia de textos mecanografiados ha alcanzado un alto porcentaje de aciertos. Las cifras varían al tratarse de imágenes digitales a partir de textos escritos a mano y el porcentaje de aciertos disminuye considerablemente. Los factores que afectan para que los mecanismos de reconocimiento disminuyan, están relacionados a las deficiencias del trazo de las letras y a las suciedades que acumulan los textos, propios del uso y del paso del tiempo. Por lo tanto, el correcto reconocimiento de un texto por parte de una herramienta OCR se encuentra lejos del óptimo en este conjunto de casos. Los CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) utilizados frecuentemente en Internet para controlar la naturaleza del usuario que utiliza un servicio web, encapsulan dichas dificultades para el reconocimiento ya que especialmente son diseñados para dificultar las tareas de un reconocedor OCR, sumándole a éstas, que los textos escritos no pertenecen a un diccionario de palabras, lo que torna aun más complejo su reconocimiento.

Los algoritmos evolutivos (AEs) son una rama de la inteligencia artificial, utilizados principalmente en problemas con espacios de búsqueda extensos y no lineales, en donde otros métodos no son capaces de encontrar soluciones en un tiempo razonable. Son métodos de optimización y búsqueda de soluciones basados en los postulados de la evolución biológica. En ellos se mantiene un conjunto de entidades que representan posibles soluciones, las cuales se mezclan, y compiten entre sí, de tal manera que las más aptas son capaces de prevalecer a lo largo del tiempo, evolucionando hacia mejores soluciones cada vez.

Este trabajo presenta los avances en una propuesta innovadora de utilizar AEs en el proceso de mejora de las características de una imagen difusa o deteriorada, para elevar los niveles de reconocimiento de una herramienta OCR. El AE se utiliza para determinar los parámetros de un modelo de transformación que utiliza funciones de conversión de imágenes para cambiar el aspecto de la imagen a reconocer mejorando su legibilidad.

Se incluye un amplio relevamiento de herramientas de reconocimiento OCR, así también una amplia evaluación de herramientas de transformación de imágenes, de modo de utilizar aquellas que presenten las mejores características de desempeño para las condiciones implícitas en el trabajo.

Se reportan los resultados obtenidos que permiten verificar la eficacia del método propuesto sobre una serie de conjuntos de imágenes de CAPTCHAs de diversas características y con diferentes dificultades de reconocimiento utilizados en Internet, determinando así mismo el nivel de efectividad de los conjuntos de imágenes para cumplir la función de un CAPTCHA.

Palabras clave: algoritmos evolutivos, procesamiento de imágenes, reconocimiento de caracteres, OCR, CAPTCHA

Índice General

Capitulo 1 Introducción

Este capítulo presenta una descripción general del proyecto así como también sus objetivos y motivación.

1.1 Descripción General del Proyecto

Este proyecto presenta una propuesta innovadora de utilizar AEs en el proceso de mejora de las características de una imagen difusa, para elevar los niveles de reconocimiento de una herramienta OCR. El AE se utiliza para determinar los parámetros de un modelo de transformación que utiliza funciones de conversión de imágenes para cambiar el aspecto de la imagen a reconocer mejorando su legibilidad. El estudio posee una completa etapa de ensayos para establecer las principales virtudes y falencias de un conjunto amplio de herramientas de transformación de imágenes, así también un relevamiento de herramientas de tipo OCR buscando la más adecuada a las circunstancias planteadas. Se reportan los resultados obtenidos que permiten verificar la eficacia del método propuesto sobre una serie de conjuntos de imágenes de CAPTCHAs de diversas características y con diferentes dificultades de reconocimiento utilizados en Internet, determinando así mismo, el nivel de efectividad de los conjuntos de imágenes para cumplir la función de un CAPTCHA.

1.2 Objetivos y Motivación del Proyecto

El reconocimiento óptico de caracteres (OCR, por su sigla en inglés), tiene como objetivo extraer de una imagen los caracteres que componen un texto para almacenarlos en un formato con el cual puedan interactuar programas de procesamiento de texto. En la actualidad, el reconocimiento de caracteres en textos mecanografiados en alfabeto latino no se considera un desafío, y es una tarea desarrollada con éxito por varios productos de software. Sin embargo, cuando las imágenes son de baja calidad y/o contienen altos niveles de ruido, los niveles de reconocimiento exitoso suelen disminuir notablemente. Un punto clave para lograr altos niveles de reconocimiento consiste en preprocesar una imagen de características difusas para obtener una versión de mayor claridad a la que sea posible aplicar exitosamente una herramienta OCR. Existen variadas propuestas para definir modelos de trasformaciones para imágenes que se basan en funciones de transformación y en general involucran un número considerable de valores paramétricos que es necesario configurar para obtener resultados precisos.

Los algoritmos evolutivos (AEs) [2] son una familia de técnicas metaheurísticas que por su amplia aplicabilidad y robustez han ganado popularidad para la resolución de problemas de optimización en los últimos veinte años. Basados en un proceso iterativo que emula la evolución de los seres biológicos, los AEs exploran el espacio de búsqueda del problema aplicando operadores estocásticos sobre un conjunto de individuos que representan soluciones del mismo, con el propósito de mejorar su fitness, una medida de la calidad de los individuos para resolver el problema.

Este proyecto estudia la utilización de un AE en el proceso de mejora de las características de una imagen difusa, con el objetivo de elevar los niveles de reconocimiento de una herramienta OCR, mejorar su efectividad y su desempeño. El AE se utiliza para determinar los parámetros de un modelo de transformación que utiliza funciones de conversión de imágenes.

1.3 Organización del documento

El documento se organiza del modo que se describe a continuación. Una descripción de la tecnología de reconocimiento óptico de caracteres, el proceso de relevamiento de herramientas OCR y el de transformación de imágenes se describen en el capítulo 2. En el capítulo 3 se presenta una introducción a los algoritmos evolutivos que serán utilizados como componente fundamental de la solución propuesta. El planteo del problema y el proceso de resolución del mismo utilizando AEs se describe en el capítulo 4. El capítulo 5 presenta los resultados experimentales obtenidos en la validación de la propuesta formulada en este proyecto. Las conclusiones finales del trabajo, así como también un resumen de las dificultades encontradas y las líneas de trabajo futuras son mencionados en el capítulo 6. Por último, se presenta un apéndice con información relevante de las etapas de experimentación en el relevamiento de herramientas de conversión de imágenes.

Capitulo 2 Tecnología de reconocimiento óptico de caracteres

El reconocimiento óptico de caracteres (OCR, por su sigla en inglés) es el proceso mediante el cual una computadora analiza una imagen estática de un documento (como un archivo TIFF, JPEG o PDF de Adobe) y traduce las palabras dentro de la imagen en caracteres de texto. De esta forma, el texto resultante se puede modificar, copiar como en un documento de texto estándar o realizar búsquedas de palabras. Nada de esto es posible sobre la imagen original. La tecnología OCR actualmente se emplea en una amplia variedad de campos para digitalizar documentos que eran mantenidos en forma impresa [24].

El OCR permite reconocer los caracteres de un documento de origen utilizando las propiedades ópticas de los equipos y medios de comunicación, mejorar la exactitud de la recopilación de datos y reducir el tiempo requerido por las personas para operar con este tipo de documentos [7]. Sin esta tecnología, sería necesario que personas lean los documentos y al mismo tiempo ingresen en un computador los caracteres o palabras que van leyendo. Otro problema es la búsqueda de una palabra dentro de un documento extenso, sin el documento digitalizado, es una tarea que puede llevar mucho tiempo o ser impracticable en grandes cantidades de documentos; en cambio, con el documento digitalizado, se utilizan algoritmos triviales que permiten encontrar una palabra dentro de documentos extensos en tiempos despreciables.

El problema de reconocimiento de texto en imágenes con ruido involucra la utilización de herramientas de reconocimiento óptico de caracteres. En este capítulo se presenta una introducción a la tecnología de reconocimiento de caracteres y el relevamiento de las principales herramientas OCR de código abierto que incluye una breve descripción de sus características y la evaluación de su poder de reconocimiento. A partir de dicha evaluación, se presentan conclusiones sobre la inclusión o exclusión de las herramientas relevadas en el programa de reconocimiento desarrollado en este proyecto.

2.1 Evolución de las aplicaciones de OCR

Los inicios de OCR se remontan a 1809, cuando las primeras patentes para dispositivos de ayuda a personas no videntes fueron concedidas. En 1912, Emmanuel Goldberg patentó una máquina que lee caracteres y luego los convierte en un código telegráfico estándar, para transmitir mensajes telegráficos a través de cables sin intervención humana. En 1914, D'Albe Fournier inventa un artefacto llamado OCR optófono que es capaz de producir sonidos, donde cada sonido corresponde a una letra específica o carácter. Después de aprender el equivalente de caracteres para los diversos sonidos, personas con dificultades visuales fueron capaces de "leer" el material impreso. La evolución de OCR continuó durante la década de 1930, siendo cada vez más importante en los inicios de la industria de la computación en la década de 1940. El desarrollo en la década de 1950 estuvo centrado en abordar las necesidades del mundo empresarial [10].

Durante la evolución de las tecnologías de OCR han existido diversos tipos de aplicaciones y herramientas basadas en ellas. Por ejemplo, el reconocimiento óptico de marcas (OMR) utiliza un formulario que tiene una serie de formas rectangulares que se llenan mediante el uso de un lápiz. Dichos formularios son la entrada para un dispositivo de escaneado que los digitaliza. Luego, el software de exploración del OMR realiza un análisis elemental de los datos escritos logrando reconocer y diferenciar los rectángulos llenos de los vacíos. La tecnología OMR se utiliza

comúnmente para la puntuación de las pruebas estandarizadas con rapidez y precisión.
Existe otra aplicación en los códigos de barras, son marcas de tipo cebra (líneas verticales blancas y negras) de varios anchos que aparecen en productos manufacturados al por menor. El uso más común del código de barras es el Universal Product Code (UPC) que representa 10 dígitos. Otros tipos sistemas de código de barras se utilizan en variedad de lugares, desde paquetes de correo hasta etiquetas de equipaje de avión. El ancho y la combinación de las rayas en el código de barras representan datos. Un lector de código de barras consiste en un escáner y un decodificador. El escáner emite un haz de luz de manera de ir distinguiendo en lo reflejos de la luz las barras y espacios. Un fotodetector convierte los espacios en una señal eléctrica y las barras en la ausencia de una señal eléctrica. El decodificador de señal analiza los patrones para validar e interpretar los datos correspondientes [13].

Otro tipo de OCR, es el de reconocimiento de caracteres de cinta magnética (MICR), el cual es utilizado por varias industrias, principalmente los bancos. La enorme cantidad de papel en forma de cheques, préstamos y estados de cuenta bancarios, junto con la necesidad de un procesamiento preciso y rápido, le impuso a la banca buscar nuevas formas de gestionar el flujo de papel. En 1956, la American Bankers Association recomendó la adopción de tinta magnética para reconocimiento de caracteres de alta velocidad automática, resultando en el desarrollo de MICR. Con MICR, los datos son grabados con una cinta magnética que es legible por cualquier dispositivo de escaneo o persona. Los cheques bancarios, representan el uso más común del MICR, los caracteres en la cinta magnética detallan el número de identificación del banco, número de la persona, cuenta y el número de cheque. Los cheques pueden ser escaneados y los datos se leen con rapidez y precisión en un ordenador para su posterior procesamiento [13].

Algunos lectores OCR pueden convertir documentos manuscritos y mecanografiados en datos digitales. Estos lectores exploran la forma de una letra en un documento, comparan las características escaneadas con una forma predefinida, y convierten el carácter en su patrón de bits correspondiente para el almacenamiento en memoria de la computadora principal. Esta tecnología todavía está en desarrollo; los documentos escritos a mano no logran alcanzar una eficacia de reconocimiento del 100% [13]. Este tipo de aplicaciones, es una de las que ha sido más estudiada, tanto por las dificultades que presenta, como por los problemas que soluciona. Actualmente existe una gran cantidad de dispositivos que realizan reconocimiento de caracteres y que se utilizan ampliamente.

El reconocimiento preciso de caracteres de grafos de tipo latino, escritos a máquina, se considera un problema resuelto en gran medida, en aplicaciones donde las imágenes son claras. Las tasas de precisión típica de estos exceden el 99%, donde la precisión total sólo puede lograrse mediante la revisión humana. Otras áreas de aplicación, incluido el reconocimiento de la impresión de escritura cursiva, o impreso en otras secuencias de comandos (especialmente los caracteres de idiomas de Asia Oriental) siguen siendo objeto de investigación activa.

La precisión de estas aplicaciones puede medirse de diferentes maneras, debiendo tenerse siempre presente que los resultados alcanzados dependerán fuertemente del método utilizado. Por ejemplo, al reconocer los caracteres pertenecientes a la página de un libro, se puede medir la precisión dividiendo la cantidad de caracteres correctos entre los totales. Otra forma de evaluar la aplicación puede ser contar la cantidad de palabras reconocidas correctamente entre las totales. Éste último método puede dar resultados muy distintos al primero, ya que la tasa de caracteres correctos puede acercarse al 100% pero la de palabras correctas puede ser mucho menor. En los casos en que se verifica el contexto, o sea, que no se verifica carácter por carácter sino a nivel de palabras, puede haber muchas variables que también determinen la forma de medición de la tasa de exactitud. Puede

haber distintos niveles de exigencia y detalle aplicado a dichas mediciones, como sólo considerar una palabra correcta si pertenece a cierto diccionario y no contiene errores, ser menos exigente y aceptarla como correcta aunque contenga errores, o introducir algo más de detalle y hacer una medición ponderada teniendo en cuenta combinaciones de ambas [14]. Para problemas más complejos de reconocimiento, se utilizan generalmente sistemas inteligentes de reconocimiento de caracteres, que involucran la utilización de redes neuronales [15].

Los nuevos requerimientos en las aplicaciones web han incorporado nuevas herramientas visuales de seguridad como son los CAPTCHAs siendo para una herramienta OCR, un nuevo desafío en materia de reconocimiento.

2.2 CAPTCHAs

El concepto de CAPTCHA surgió de la necesidad de brindar seguridad a las aplicaciones web de ataques provenientes de herramientas automatizadas.

Un CAPTCHA es una herramienta para proteger sitios web contra ataques producidos por herramientas automatizadas realizando pruebas que los seres humanos pueden pasar pero que los programas informáticos actuales no pueden. Por ejemplo, los humanos pueden leer texto distorsionado como el que se muestra en la Figura 1, pero los programas informáticos actuales no lograrían pasar la prueba. Dicho ejemplo también posee una prueba de audio para que los usuarios no videntes puedan utilizar la herramienta reconociendo el texto a través del audio. El término CAPTCHA (por Completely Automated Public Turing Test To Tell Computers and Humans Apart) fue acuñado en el año 2000 por Luis von Ahn, Manuel Blum, Nicholas Hopper y John Langford, de la Universidad Carnegie Mellon [11]. En la actualidad, la mayoría de los sistemas de CAPTCHA utilizan imágenes con caracteres en ellas que no pueden ser reconocidos por aplicaciones de OCR.

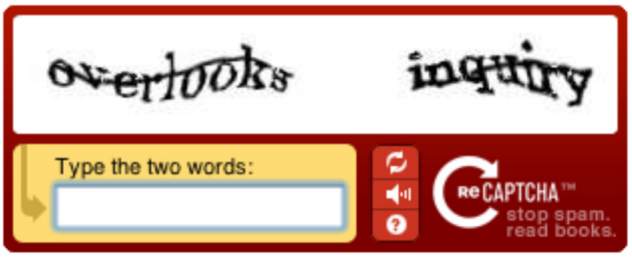


Figura 1: Ejemplo de un captcha

Los CAPTCHAs tienen varias aplicaciones prácticas para la seguridad de las aplicaciones. A continuación se describen algunas de las más importantes:

- **Prevenir los comentarios spam en blogs:** La mayoría de los bloggers están familiarizados con los programas que presentan comentarios falsos, por lo general con el fin de aumentar el posicionamiento en un motor de búsqueda de un sitio. Mediante el uso de un CAPTCHA, sólo los humanos pueden introducir comentarios en un blog ya que los programas que antes lo hacían automáticamente, ahora quedan inutilizados al no poder ingresar automáticamente los caracteres que se observan en la imagen del CAPTCHA.
- **La protección de inscripciónes Web:** Varias compañías ofrecen servicios gratuitos de correo electrónico. Hasta hace unos años, la mayoría de estos servicios han sufrido ataques provenientes de pequeñas aplicaciones llamadas "bots", capaces de crear miles de cuentas

de correo electrónico cada minuto. La solución a este problema fue utilizar CAPTCHAs para garantizar que sólo humanos puedan obtener cuentas gratuitas. En general, los servicios gratuitos deberían estar protegidos con un CAPTCHA para evitar abusos por parte de scripts automatizados.

- **Prevención de ataques de diccionario:** Un CAPTCHA también puede usarse para prevenir ataques de diccionario en los sistemas de contraseña. La idea es simple, impedir que un equipo sea capaz de recorrer todo el espacio de las contraseñas obligándolo a resolver un CAPTCHA después de un cierto número de intentos fallidos de acceso. Esto es mejor que el enfoque clásico de bloqueo de una cuenta después de una secuencia de intentos fallidos de acceso, ya que hacerlo permite a un atacante bloquear las cuentas a su antojo.

Dado las características de un CAPTCHA, el cual está diseñado para "engañar" a una herramienta automática, no es una tarea trivial el reconocimiento del texto contenido en una imagen utilizando únicamente una herramienta de tipo OCR.

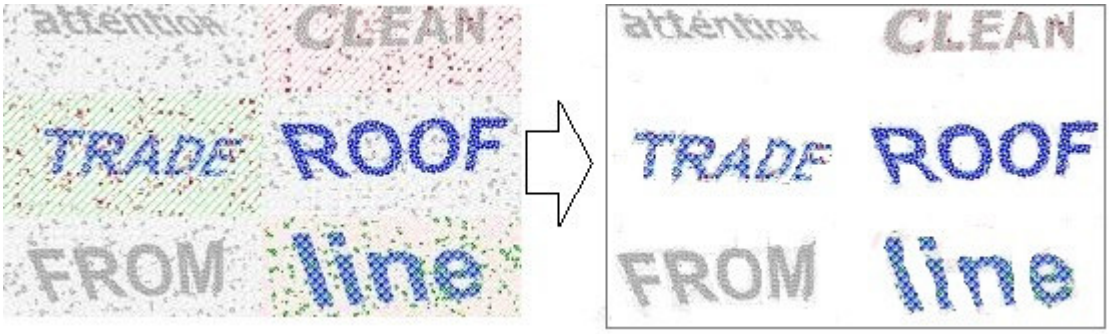


Figura 2: Trasformación previa al reconocimiento

El reconocimiento de un CAPTCHA puede separarse en dos etapas, una de transformación de la imagen para simplificarla y otra de reconocimiento propiamente dicho. Planteado este problema, y considerando a una herramienta OCR incapaz de reconocer un CAPTCHA, es de interés saber cuál será el desempeño del OCR sobre una imagen que fue previamente "limpiada" como se muestra en la Figura 2.

A continuación se realizará un relevamiento entre diferentes herramientas de reconocimiento OCR, evaluando su desempeño al intentar reconocer CAPTCHAS que fueron previamente simplificados y luego se relevan las herramientas de transformación de imágenes.

2.3 Herramientas de OCR

Se realizó un relevamiento de una serie de herramientas de código libre de diferentes características, evaluando su desempeño en el reconocimiento de CAPTCHAs de diversas dificultades y diferentes particularidades. El relevamiento incluye las herramientas HOCR, GOCR, OCRAD y Tesseract.

2.3.1 HOCR

LibHocr [20] es una biblioteca de GNU de reconocimiento óptico de caracteres principalmente diseñada para explorar textos bíblicos. La biblioteca permite analizar documentos de imágenes, en los diversos formatos de imagen (TIFF, JPEG, pnm, bmp, ...), utilizando el módulo de imagen GTK y brinda como salida resultados en un archivo de texto.

Posee la habilidad de convertir automáticamente las imágenes de color amarillo manchado en imágenes de color blanco y negro, facilitando el reconocimiento de caracteres. A su vez dicha herramienta automáticamente puede escalar y rotar las imágenes, según sea necesario dado el diseño de página y los módulos de reconocimiento que esta posee. A diferencia de otros reconocedores, puede reconocer automáticamente las columnas y los párrafos en la página, distinguiendo entre el texto y partes gráficas, y haciendo caso omiso a las zonas de ruido. Esta herramienta también tiene la posibilidad de mejorar la búsqueda apoyándose en un diccionario siempre y cuando los caracteres pertenezcan a un lenguaje ya determinado. A nivel académico, no aparecen registrados reportes de su desempeño que pueda servir como base a la hora de su evaluación.

2.3.2 GOCR

GOCR [18] es un programa de reconocimiento de caracteres ópticos para imágenes de extensión “pnm” con la posibilidad de desplegar la salida reconocida en un archivo de texto. GOCR tiene la posibilidad de mejorar la búsqueda apoyándose en una base de datos de imágenes ya reconocidas. El reconocedor brinda la posibilidad de ajustar los niveles de grises reconocidos de modo de calibrar el reconocimiento en textos con ruido agregado. Por último esta herramienta tiene la ventaja de potenciar su reconocimiento utilizando corrección basada en el contexto reconocido. A nivel académico, no aparecen registrados reportes de su desempeño que pueda servir como base a la hora de su evaluación.

2.3.3 OCRAD

OCRAD [19] es un programa de reconocimiento de caracteres ópticos basado en un método de extracción de características (feature extraction). OCRAD lee una imagen en formato pbm (mapa de bits), pgm (escala de grises) o ppm (color), y produce texto en formato byte (8-bit) o UTF-8. Incluye un analizador de composición (layout) capaz de separar las columnas o bloques de texto que forman normalmente las páginas impresas.

Dicho programa puede ser usado como aplicación autónoma en modo texto, o como complemento (backend) de otros programas. Esta herramienta también tiene una serie de funciones de transformación de imágenes para ampliar las posibilidades de reconocimiento. A nivel académico, no aparecen registrados reportes de su desempeño que pueda servir como base a la hora de su evaluación.

2.3.4 *Tesseract*

Tesseract [12] es un programa de reconocimiento de caracteres ópticos que lee un archivo binario, en escala de grises o en su defecto en escalas de más colores obteniendo como salida archivos de texto.

Este motor fue desarrollado por Hewlett-Packard entre 1985 y 1995. En 1995 fue reconocido como uno de los 3 mejores en el "UNLV Accuracy Test", teniendo en cuenta su precisión en pruebas de reconocimiento [25]. Luego de diez años sin desarrollo ni mejoras sobre el motor, en 2005 fue liberado como código abierto por HP y UNLV (University of Nevada, Las Vegas) bajo la licencia Apache en su versión 2.0. Desde ese año hasta la fecha actual, Tesseract es desarrollado y mantenido por Google.

Hoy en día es considerado como uno de los más precisos motores de OCR de código abierto disponible. Dicha biblioteca posee la posibilidad de potenciar su reconocimiento basándose en un diccionario de un lenguaje específico en el caso del proceso de textos. En este caso también posee la posibilidad de un aprendizaje basado en el contexto obtenido de reconocimiento de archivos anteriores. Tesseract posee la ventaja de haber sido testeada exitosamente en diferentes plataformas incluyendo diferentes versiones de Ubuntu, Windows (x86/32), Mac OS X (x86, PPC).

2.3.5 *Evaluación de las herramientas OCR*

Debido a la falta de información acerca del desempeño comparativo de las herramientas descriptas previamente en la subsección anterior, se realizó una evaluación de las herramientas. A continuación se detallan las primeras conclusiones acerca de su poder y efectividad para la aplicación en este proyecto.

El proceso de evaluación de las herramientas consistió en medir el porcentaje de acierto de palabras reconocidas que tuvo cada una de las bibliotecas. Los experimentos fueron realizados sobre los distintos grupos de imágenes definidos en el proyecto. Los conjuntos de imágenes de prueba se presentan en detalle en la sección 7.1. Las imágenes fueron "limpiadas a mano" como se muestra a modo de ejemplo en la Tabla 1 previo al reconocimiento para disminuir considerablemente su ruido debido a que de otra forma el reconocimiento del texto en las imágenes sería casi imposible para sus características. No se utilizaron características específicas de apoyo como lo pudo ser la ayuda de un diccionario o calibración especial para el reconocimiento de ninguna de las herramientas, de modo de no sobrestimar herramientas utilizando mecanismos que no aplican directamente al problema. Los resultados obtenidos de esta experiencia plantearan un piso de posibilidades de aciertos para la aplicación y se detallan en la Tabla 2.

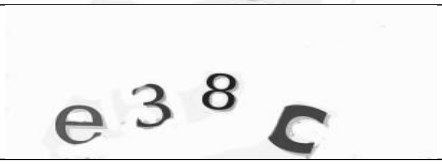
Imagen original	Imagen limpia
	
	
	
	

Tabla 1: Ejemplos de imágenes procesadas manualmente.

Herramienta	Aciertos conjunto 1	Aciertos conjunto 2	Aciertos conjunto 3	Aciertos conjunto 4	Aciertos conjunto 5	Aciertos conjunto 6
OCRAD	0%	0%	37%	0%	0%	0%
HOCR	0%	0%	0%	0%	0%	0%
GOCR	21%	0%	37%	0%	0%	0%
TESSERACT	85%	0%	50%	0%	0%	0%

Tabla 2: Resultados de reconocimiento obtenidos para cada herramienta

2.3.6 Conclusiones del relevamiento de herramientas OCR

A continuación se detallan las conclusiones obtenidas a través del relevamiento de información de las características de las herramientas OCR, así como también aquellas obtenidas a partir del proceso de experimentación.

El relevamiento de información de las herramientas de reconocimiento OCR aporta escasos datos referentes al desempeño de las herramientas o evaluaciones comparativas debidamente documentadas que puedan indicar alguna preferencia por alguna de ellas. Por otra parte existen ciertos complementos de alguna de las herramientas que en otro contexto hubieran sido útiles como el uso de diccionarios de palabras en el caso de reconocimiento de palabras de un determinado lenguaje. Esto último, no será considerado en este proyecto, pero podría resultar ventajoso incorporarlo en un futuro. Finalmente, la herramienta HOCR, en su documentación es sugerida para

caligrafías antiguas, aunque no se especifica si su desempeño es bueno en caligrafías modernas.

Por otro lado, el proceso de experimentación con los diferentes conjuntos de imágenes y las diferentes herramientas de experimentación, permitió obtener más elementos para evaluar el poder de los reconocedores. A pesar de las diferentes características de las imágenes pertenecientes a cada uno de los conjuntos, los resultados experimentales obtenidos por Tesseract, la posicionan como la herramienta de reconocimiento más eficaz. Las restantes siempre estuvieron por debajo de esta en los porcentajes de reconocimiento. Este comportamiento también pudo comprobarse si se considera el reconocimiento parcial (incompleto) de palabras. Los tiempos de ejecución que tuvieron las diferentes bibliotecas OCR, para realizar su tarea, fueron de orden muy similar, en el entorno de las décimas de segundo, sin diferencias significativas.

Las herramientas se comportaron de manera diferente dependiendo de las características de los diferentes grupos de imágenes. Existieron cuatro grupos de imágenes para los cuales los resultados obtenidos para todas las herramientas fueron 0%. En el caso del grupo de "LANAP Software BotDetect" (ver sección 7.1), y dadas las características de las imágenes de este grupo, podría deducirse, que las herramientas OCR utilizadas encontraron suma dificultad en el reconocimiento de caracteres los cuales no están determinados por líneas continuas. En todas las instancias de este grupo, los textos reconocidos mostraban más de un carácter por renglón, como si los reconocedores distinguieran las discontinuidades de las líneas de un carácter como caracteres distintos. Por otro lado, en el caso de los grupos "Framework Aardvark Topsites PHP", "Loquo Forum" y "Otras imágenes" en el que los caracteres estaban perfectamente conformados por líneas continuas, la baja eficacia de las herramientas, puede deberse a las considerables rotaciones que poseen como característica cada uno de los caracteres de este conjunto. Parecería imprescindible para mejorar la tasa de aciertos cambiar el ángulo de las imágenes de modo de facilitar la tarea de los reconocedores.

Tomando en cuenta lo antes expuesto, en este trabajo se adopta para el reconocimiento de caracteres, la herramienta Tesseract.

2.4 Relevamiento de herramientas de transformación de imágenes

A continuación se presenta el relevamiento realizado de las distintas herramientas de conversión de imágenes.

Para favorecer la tarea de reconocimiento de imágenes, a partir de diversas experiencias de reconocimiento OCR, se intentó determinar qué tipos de cambios en las características visuales de las imágenes, facilitaban las tareas realizadas por las herramientas OCR.

En primer lugar, las experiencias mostraron que aquellas imágenes, cuyos caracteres no estaban trazados con líneas continuas, eran difícilmente reconocidas. Otra clara dificultad, radicaba en aquellas imágenes cuyos caracteres aparecían mezclados con diversas distorsiones como lo pueden ser manchas de colores, trazos de líneas u formas geométricas, y los diversos tipos de ruidos que son comúnmente vistos en los CAPTCHAs. Por último, otra de las principales dificultades que tuvieron los reconocedores, tuvo que ver con la variación del ángulo de inclinación que poseen los caracteres en virtud del eje horizontal, así también como con ciertas desviaciones en el trazo corriente de los caracteres.

Dadas las experiencias antes mencionadas y en procura de simplificar la tarea de las herramienta OCR, se deben identificar aquellas funciones de transformación de imágenes que permitan eliminar las dificultades planteadas. En resumen, una transformación aplicada sobre una imagen será favorable, si contribuye a definir el trazo de los caracteres de la imagen, a eliminar los ruidos que perturban la visibilidad de los caracteres, a disminuir el ángulo de rotación sobre el eje horizontal o actúa reduciendo desviaciones en el trazo de cada carácter.

A continuación se presenta el relevamiento de distintas funciones de transformación de imágenes de la biblioteca ImageMagick [21], de forma de obtener una valoración de las posibilidades de que éstas resulten útiles para facilitar el desempeño de los motores OCR.

ImageMagick es una biblioteca de software que permite la edición y manipulación de imágenes en más de 100 formatos (JPEG, GIF, TIFF, DPX, PDF, PNG, SVG, etc.), incluyendo los más utilizados actualmente en diversas aplicaciones e Internet. Además de ser software libre (distribuido bajo una licencia muy similar a la Apache 2.0) y tener más de siete años de desarrollo, posee varias interfaces para acceder a sus funcionalidades, desde directamente la línea de comandos, hasta librerías para los lenguajes de programación más utilizados en la actualidad (Java, C++, C, PHP, Python, Ruby, Perl, etc.). Estas fueron las razones, además de la cantidad de funciones y la potencia de las mismas, por las que se eligió esta biblioteca para la transformación de las imágenes en este proyecto.

2.4.1 Características del proceso de experimentación

El proceso de experimentación y relevamiento fue efectuado cuidadosamente de manera de intentar cubrir la mayor parte de las características de los diferentes tipos de imágenes abordados en este trabajo. De esta forma, para cada una de las funciones evaluadas, fue necesario repetir la evaluación en imágenes pertenecientes a todos los grupos de imágenes involucrados. Dicha evaluación debió considerar las variaciones en el efecto de transformación de las imágenes al variar los valores de los parámetros de las mismas. Dependiendo de las características de cada función y a través de la observación de las transformaciones reales, se debió decidir de qué forma variar dichos parámetros considerando distintos intervalos de valores. En resumen, el proceso consistió en realizar múltiples ejecuciones para cada una de las funciones tomando imágenes de los diferentes grupos, y para cada una de ellas realizar las evaluaciones variando el valor de los parámetros de las mismas. Luego del proceso de evaluación de cada función, se consideró para la evaluación aquella que logró mejores resultados teniendo en cuenta los aspectos favorables para el reconocimiento establecidos en este capítulo. Dichos aspectos se corroboraron visualmente en cada una de las imágenes, observando por ejemplo cuales funciones servían para aumentar el contraste entre los caracteres y el fondo o cuales contribuían en el suavizado de imperfecciones de los caracteres.

Las funciones fueron clasificadas en las categorías: claramente favorables, probablemente favorables, no favorables y aquellas que no aplican. El objetivo de esta evaluación es poder descartar aquellas funciones que no resultan favorables para el reconocimiento o que no se aplican y ponderar las que sí lo son.

2.4.2 Evaluación de las funciones de transformación de imágenes de la biblioteca ImageMagick

En esta subsección se describe el resultado obtenido experimentalmente de la evaluación de las funciones pertenecientes a la biblioteca ImageMagick más relevantes. La herramienta utilizada para acceder a dichas funciones, es el ejecutable "convert" provisto por la biblioteca. Dicha herramienta provee una interfaz para acceder a todas las funciones de la biblioteca desde la línea de comandos.

En el Apéndice I se presenta un resumen de los resultados completos obtenidos para conjunto completo de funciones relevadas. Se evaluaron más de 130 funciones, variando los distintos parámetros que posee cada una y comprobando los resultados con un conjunto de imágenes representativas. En los casos en que se identificaba una favorable, se profundizaba la investigación y la experimentación, leyendo en detalle la documentación para entender sus efectos sobre las imágenes y realizando más experimentos, combinados con otras funciones por ejemplo, para evaluar su utilidad en el contexto de este proyecto.

2.4.3 Conclusiones del relevamiento de las funciones de transformación de imágenes

Se obtuvo un conjunto de 12 funciones para procesar las imágenes, conformado por las que mostraron los mejores resultados para el objetivo de este proyecto. Dichas funciones no fueron las únicas con resultados favorables, pero debido a las limitaciones de tiempo de ejecución y poder de procesamiento, debieron dejarse de lado varias que también mostraban buenos resultados. Todas ellas pueden ser consultadas en el Apéndice I.

Se lograron establecer los rangos útiles de valores de parámetros para cada función. Estos valores se obtuvieron luego de observar y analizar los parámetros con los que cada función generaba buenos resultados en distintos tipos de imágenes. Para esto no sólo se tuvieron en cuenta las observaciones de los resultados sobre las imágenes sino también las limitaciones de procesamiento, ya que algunas de las funciones son costosas en términos de utilización de CPU.

En la Tabla 3, se presenta la descripción de las funciones de transformación y el rango de parámetros que se considera más favorables para mejorar el reconocimiento.

Función	Descripción	Parámetros	Rango
Grayscale	Convierte el espacio de colores de la imagen a escala de grises.	No tiene.	0..1
Threshold	Aplica un umbral sobre blanco y negro de forma tal que maximiza los valores de los pixeles que superan dicho umbral y minimiza al resto.	Umbral para la aplicación del filtro.	5..99 %
Paint	Extiende los trazos de color de la imagen.	Factor que aumenta o disminuye la dilución de los colores.	1..4
Negate	Modifica cada pixel por su valor opuesto en la imagen.	No tiene.	0..1

Noise	Adhiere o remueve ruido a la imagen.	Factor que modifica el efecto del ruido sobre la imagen.	0..3
Equalize	Ecualiza los colores de la imagen según su histograma.	No tiene.	0..1
Blur	Reduce los niveles de ruido y detalle en la imagen.	Factor que define la potencia de la función aplicada.	1..4
Charcoal	Realiza un efecto de filtro de tipo "carbón" sobre la imagen.	Radio utilizado como factor de ajuste para el efecto.	1..5
Colorize	Colorea la imagen con el valor especificado.	Valor que define el color a utilizar.	20..70 %
Posterize	Limita el número de niveles de colores de la imagen.	Número de niveles de colores de la imagen.	2..7 %
Level	Ajusta los niveles de contraste de la imagen.	Factor de ajuste para los niveles de contraste de la imagen.	20..100 %
Fuzz	Considera iguales a los colores que se encuentran a una determinada distancia en la imagen.	Valor de la distancia mínima entre los colores de la imagen.	20..100 %

Tabla 3: Descripción de las funciones de transformación favorables

Es importante observar que algunos rangos se definen mediante porcentajes. En principio se utilizaron valores absolutos, esto requería la utilización de rangos mucho mayores, ya que los valores útiles para algunas imágenes estaban muy alejados de los de otros. Luego se optó por utilizar porcentajes, ya que así los parámetros son relativos a cada imagen y permite reducir considerablemente el espacio de los valores posibles de los parámetros. Se puede clarificar el concepto de parámetros relativos a imágenes con un ejemplo. La biblioteca posee una función que convierte el color de todos los pixeles de la imagen con un brillo menor al parámetro dado, en negro. Utilizando el parámetro absoluto, el rango de valores que pude tomar, pertenece a los números naturales entre 0 y 255, pero utilizando relativos, el rango también pertenece a los naturales pero entre 0 y 100. Los valores entre 0 y 100 representan porcentajes y son relativos a cada imagen; en este caso, el 0 representa al valor con menor brillo de la imagen y el 100 al de mayor, los cuales no necesariamente deben ser iguales a negro y blanco respectivamente.

Capitulo 3 Algoritmos evolutivos

Charles Darwin introdujo su teoría de la evolución natural de las especies en el libro "El origen de las especies" [3]. En dicha teoría se establece que los organismos biológicos evolucionan siguiendo el principio de selecciona natural que está basado en la "supervivencia del más apto". En la naturaleza, los individuos de la población de una misma especie compiten entre sí para obtener pareja para la reproducción y recursos vitales como alimentos, vivienda, etc. Los individuos con baja aptitud tienen menos posibilidades de sobrevivir y de reproducirse. Durante la reproducción se produce una recombinación de las características de cada ancestro que puede producir descendientes mejor adaptados. Con el paso de las generaciones, la especie evoluciona espontáneamente a estar mejor adaptada a su medio ambiente.

A partir de dichas ideas y como éstas permiten en la naturaleza la mejora de las especies, varios autores han planteado métodos para la resolución de los problemas de optimización, inspirados en la evolución natural [11]. En 1975, John Holland desarrolló esta idea en su publicación "Adoptation in natural and artificial systems" [3] en la que describe como aplicar los principios de la evolución natural a la resolución de problemas de optimización. Actualmente, la idea de Holland ha sido desarrollada ampliamente y los Algoritmos Evolutivos (AEs) son una poderosa herramienta para resolver problemas de búsqueda y optimización.

En este capítulo se presenta una breve introducción a los algoritmos evolutivos, sus principales características, su evolución histórica y el contexto biológico en el que están inspirados.

3.1 Contexto biológico

La palabra "genética" se deriva de la palabra griega "génesis", que significa "crecer" o "convertirse". La ciencia de la genética nos ayuda a diferenciar entre la herencia y las variaciones, dando cuenta de las semejanzas y diferencias con los conceptos derivados directamente de la herencia natural. Los conceptos biológicos en los que se basan los algoritmos evolutivos están directamente derivados de la evolución natural. Los principales términos involucrados en el contexto biológico de las especies son las que se presentan a continuación [11]:

- Célula: las células de animales forman un complejo sistema. El funcionamiento de dicho sistema está centrado en el núcleo de las células, donde está contenida la información genética.

- Cromosomas: toda la información genética se almacena en los cromosomas. Cada cromosoma es parte de la construcción del ADN. Los cromosomas se dividen en varias partes llamados genes. Estos codifican las propiedades de las especies, es decir, las características individuales. Las posibles formas o valores de los genes de una propiedad se llaman alelo y un gen puede tener diferentes alelos. Por ejemplo, hay un gen para el color de los ojos, y todos los diferentes alelos posibles son la posibilidad de ser de color negro, marrón, azul y verde (ya que nadie posee ojos de color rojo u ojos de color violeta). El conjunto de todos los alelos presentes pueden determinar todas las variantes posibles para las generaciones futuras. El tamaño de la reserva genética ayuda a determinar la diversidad de los individuos en la población. El conjunto de todos los genes de una especie en particular se llama genoma. Cada gen tiene una posición única en el genoma llamada locus. De hecho, la mayoría de los organismos vivos almacenan su genoma en varios cromosomas.

- Genética: para un individuo en particular, toda la combinación de genes se llama genotipo. El fenotipo describe el aspecto físico de descifrar un genotipo. Un punto interesante de la evolución es que la selección se hace siempre en el fenotipo, mientras que en la reproducción se recombina el genotipo. En las formas de vida más altas, los cromosomas contienen dos conjuntos de genes, conociéndose como diploides. En caso de conflictos entre dos valores del mismo par de genes, la parte dominante será capaz de determinar el fenotipo, mientras que el otro, llamado recesivo, todavía estará presente permitiendo una mayor diversidad de alelos. En representación de haploides, sólo uno de cada gen se almacena, por lo tanto el proceso de determinar el alelo dominante es evitado.

- Reproducción: la reproducción de las especies a través de la información genética se lleva a cabo de dos formas distintas, la mitosis o la meiosis. En la mitosis la misma información genética es copiada a nuevas crías, no existiendo intercambio de información. Esta es una forma normal del crecimiento de las estructuras celulares de varios organismos, por ejemplo los órganos. La meiosis se basa en la reproducción sexual. La reproducción consiste en la creación de un nuevo individuo a partir de 2 progenitores (reproducción sexual) o de un único progenitor (reproducción asexual). Durante la reproducción sexual ocurre la recombinación o cruzamiento. En el caso de individuos haploides, se intercambian los genes entre los cromosomas de los dos padres. En el caso de individuos diploides, para cada padre se intercambian los genes entre cada par de cromosomas, formando un gameto. Luego, los gametos de los dos padres se aparean para formar un único conjunto de cromosomas diploides.

- Selección natural: la selección natural estimula la preservación de las variaciones favorables y el rechazo de las variaciones desfavorables. La variación consiste en las diferencias mostradas por un individuo de la especie y también por los descendientes de los mismos padres. En la mayoría de las especies, nacen más individuos de los que pueden sobrevivir. Los individuos más aptos tienen una mayor posibilidad de sobrevivir en la continua lucha por la vida.

3.2 Algoritmos genéticos

Los Algoritmos Genéticos (AG) fueron desarrollados a partir de la idea de John Holland que consistía en un método heurístico para la resolución de problemas basado en "la supervivencia del más apto". El trabajo de Holland tenía un doble objetivo, por un lado mejorar la comprensión del proceso de adaptación natural y por otro lado, el diseño de sistemas con propiedades similares a los sistemas naturales.

Los AGs son métodos estocásticos que trabajan sobre una población de soluciones. El manejo de más de una solución permite recombinar características de soluciones diferentes, obteniendo soluciones mejor adaptadas. Los AGs son robustos ya que tienen un buen rendimiento para una amplia gama de problemas. No existen requisitos especiales sobre el tipo de problemas que se pueden resolver. El funcionamiento de los AGs se basa en la aplicación de los operadores de cruzamiento y mutación.

La recombinación o la reproducción sexual es clave para la evolución natural. Técnicamente se necesitan dos genotipos mediante los cuales se produce un nuevo genotipo, a través de la mezcla de los genes. En biología, la forma más común de recombinación es el cruzamiento, lo que implica que dos cromosomas se cortan en un punto y las mitades son unidas para crear nuevos cromosomas. El

efecto de la recombinación es muy importante porque permite que características de ambos padres se transmitan a los hijos. Los padres poseen diferentes cualidades siendo deseable que las buenas cualidades pasen a sus hijos. El éxito de los AGs para obtener buenas soluciones se basa en gran medida en la mezcla de material genético que se logra mediante el cruzamiento.

La mutación es otra manera de obtener nuevos genomas. La mutación suele consistir en cambiar aleatoriamente el valor de los genes. En la evolución natural, la mutación no es un operador muy frecuente. Sin embargo, en optimización, un cambio al azar puede ser una buena manera de generar un salto de una solución a otra región del espacio de búsqueda.

Los AGs se han mostrado como una herramienta útil para la resolución de problemas de búsqueda y optimización. Debido al éxito de los AGs han surgido otros tipos de algoritmos basados en el mismo principio de la evolución natural. La clasificación no siempre es clara entre las diferentes técnicas. En lo que resta del documento se hará referencia a los Algoritmos Evolutivos (AEs) que engloban a las distintas técnicas.

3.3 Definición de algoritmos evolutivos

Los AE son técnicas estocásticas de optimización y búsqueda que han sido ampliamente aplicadas para resolver problemas complejos en múltiples áreas de aplicación [4]. Un AE es una técnica iterativa basada en emular el proceso de evolución natural de los seres vivos que aplica operadores estocásticos sobre un conjunto de individuos (la población), con el propósito de mejorar su fitness, una medida relacionada con la función objetivo del problema de optimización. Cada individuo codifica una solución tentativa al problema estudiado. La población inicial se genera mediante un procedimiento específico, habitualmente aleatorio. Dicha población debe ofrecer una amplia diversidad de materiales genéticos. La reserva genética debería ser tan grande como sea posible por lo que cualquier solución del espacio de búsqueda puede ser engendrada. La aplicación iterativa de operadores como la recombinación de partes de dos individuos y la mutación aleatoria de su codificación, aplicados a individuos seleccionados según su fitness, guían al AE a soluciones tentativas de mayor calidad. La formulación clásica de un AE sigue el esquema presentado en la Figura 3.

```
Inicializar(P(0))
gener = 0
Evaluar(P(0))
Mientras (no CriterioParada) hacer
       Padres = Selección(P(gener))
       Hijos = Operadores Evolutivos(Padres)
       NuevaP = Reemplazar(Hijos, P(gener))
       gener ++
       P(gener) = NuevaPop
Fin mientras
Retornar Mejor Solución Hallada
```

Figura 3: Esquema de un algoritmo evolutivo.

Cada iteración se compone de los siguientes pasos [11]:

• Selección: El primer paso consiste en la selección de individuos para la reproducción, la cual se realiza aleatoriamente con una probabilidad en función de la aptitud de los individuos.

• Reproducción: En el segundo paso, los individuos seleccionados se reproducen generando hijos. Para generar nuevos cromosomas, el algoritmo puede utilizar tanto la recombinación como la mutación.

• Evaluación: A continuación, se evalúa la aptitud de los nuevos cromosomas.

• Reemplazo: Durante el último paso, algunos o todos los individuos de la población son dados de baja y son sustituidos por los nuevos.

La paralelización de los AE se ha popularizado como un mecanismo para mejorar su desempeño. Dividiendo la población o el cálculo de fitness entre varios elementos de procesamiento, los AE paralelos permiten abordar problemas complejos de grandes dimensiones, hallando resultados de buena calidad en tiempos razonables [1].

3.4 Representación de algoritmos evolutivos

Al utilizar un algoritmo evolutivo para resolver un problema de optimización, lo primero que debe establecerse es como se mapea el problema original a la representación con la que trabajan los AEs. Para esto las posibles soluciones del problema son codificadas en individuos [5].

El AE maneja una población de posibles soluciones donde cada solución se representa a través de un cromosoma, siendo esta sólo una representación abstracta. La representación tradicionalmente utilizada para un algoritmo genético simple es una cadena binaria como se muestra en la Figura 4. La longitud de la representación depende de las características del problema (número de variables, número de funciones objetivo, dimensión del dominio), y de características de la solución buscada (precisión deseada, por ejemplo).

Figura 4: Representación binaria de un cromosoma

Existe otro tipo de representación que suele utilizarse para los problemas de ordenamiento. Estos problemas se expresan como permutaciones de *n* variables entonces la representación consiste en una lista de *n* enteros, cada uno ocurriendo una única vez. En dicha representación se debe tener sumo cuidado en el diseño de los operadores evolutivos ya que los operadores podrían producir permutaciones inválidas, es decir, permutaciones que tengan etiquetas repetidas y que omitan alguna etiqueta.

3.5 Operadores evolutivos: selección

Para que la población evolucione, es necesario crear nuevos individuos con el fin de que algunos de estos resulten mejor adaptados que los actuales. Los individuos son creados a partir de uno o más individuos ya existentes, a los cuales se les llama padres. El mecanismo de selección de padres escoge qué individuos serán utilizados para generar nuevos individuos, llamados hijos. Escoger a los mejores individuos como padres, aumenta la posibilidad de generar buenos individuos [5]. El

propósito de la selección es hacer hincapié en los individuos mejor adaptados de la población con la esperanza de que sus descendientes sean aún mejores. El problema es cómo seleccionar estos individuos. La selección es un método que elige aleatoriamente a los cromosomas de la población de acuerdo con la función de aptitud. Cuanto mayor sea el valor de la función de fitness, mayor es la probabilidad de que el individuo sea seleccionado. La tasa de convergencia de un AE es determinada en gran medida por la magnitud de la presión de la selección, resultando en tasas de convergencia mayor [11].

Los AEs deben ser capaces de alcanzar soluciones óptimas o casi óptimas, sin embargo, si la presión de selección es demasiado baja, la velocidad de convergencia será lenta, y se tomará mucho tiempo para encontrar una solución óptima. Si la presión de selección es demasiado alta, puede darse una convergencia prematura a soluciones subóptimas. Por lo general se pueden distinguir dos tipos de sistema de selección, la selección proporcional y la selección ordinal. La selección proporcional escoge los individuos en base a su aptitud relativa a los otros individuos de la población y la de tipo ordinal a su valor de fitness absoluto. Los métodos de selección más comúnmente usados son los siguientes [11]:

- Selección de rueda de ruleta: la selección de la ruleta es una de las técnicas de selección tradicionales de un AG. La frecuencia con que se utiliza el operador para la reproducción es proporcional a la aptitud. Se crea un pool genético formado por cromosomas de la generación actual, en una cantidad proporcional a su fitness. Si la proporción hace que un individuo domine la población, se le aplica alguna operación de escalado. Dentro de este pool, se seleccionan parejas aleatorias de cromosomas y se emparejan, sin importar incluso que sean el mismo individuo. Esta es una técnica de selección moderadamente fuerte, ya que no garantiza que los individuos más aptos sean seleccionados para la nueva generación, sólo son los que tienen mayores posibilidades. Hay otras variantes, en las que en la nueva generación se puede incluir el mejor representante de la generación actual. En este caso, se denomina método elitista.

- Selección aleatoria: esta técnica selecciona al azar uno de los padres de la población. Los resultados de este método son bastante más impredecibles que los obtenidos en la selección por rueda de ruleta.

- Selección por rango: esta técnica consiste en elegir los mejores elementos previamente ordenados por su aptitud.

- Selección por torneo: se eligen un subconjunto de elementos y a partir de los mismos se seleccionan un número fijo de elementos a partir de su aptitud.

3.6 Operadores evolutivos: cruzamiento

El cruzamiento es el proceso por el que se producen dos hijos a partir de dos soluciones padres. El operador de cruce se aplica con la esperanza de crear un descendiente mejor.

Las diferentes técnicas de cruzamiento se describen a continuación:

- Cruzamiento de un punto: el operador de cruzamiento más sencillo consiste en elegir un punto de cruce al azar, copiar la información del primer padre antes de este punto y copiar la información del otro padre después del punto de cruce. Este procedimiento se ilustra en la Figura 5.

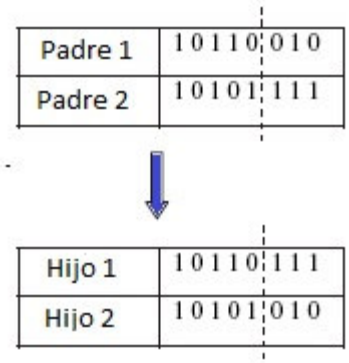


Figura 5: Cruzamiento de un punto

- Cruzamiento de dos puntos: el problema con la adición de otros puntos de cruce es que los bloques genéticos contiguos son más propensos a ser interrumpidos. Sin embargo, una ventaja de tener más puntos de cruce es que la cantidad de soluciones distintas que se pueden construir es mayor. En el cruce de dos puntos, se eligen dos puntos de cruce y el contenido entre estos puntos es intercambiado como se muestra en la Figura 6.

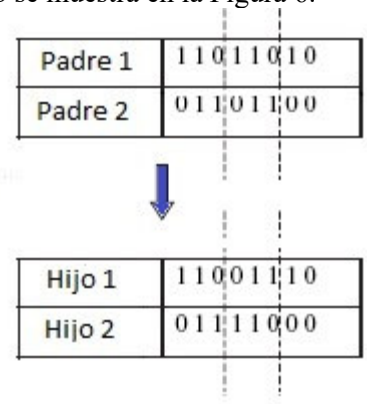


Figura 6: Cruzamiento de dos puntos

- Cruzamiento multi-punto: en este tipo de cruce se seleccionan los *n* puntos en donde se realizará el corte. De esta forma, y de manera análoga al cruzamiento de dos puntos, se irán intercambiando los fragmentos de las cadenas para formar los dos nuevos hijos.

- Cruzamiento uniforme: el cruzamiento uniforme es muy diferente al cruzamiento de N-Puntos. En el descendiente cada gen se crea copiando el gen correspondiente de uno u otro padre elegido de acuerdo a una máscara binaria aleatoria generada de la misma longitud que los cromosomas. Donde hay un 1 en la máscara de cruce, el gen se copia del primer padre, y donde hay un 0 en la máscara del gen se copia del segundo padre. Una máscara para un nuevo cruzamiento es generado aleatoriamente para cada par de padres. Los descendientes, por lo tanto, contienen una mezcla de genes de cada padre. En la Figura 7, los hijos se producen utilizando el enfoque de cruce uniforme.

Padre 1	1 0 1 1 0 0 1 1
Padre 2	0 0 0 1 1 0 1 0
Mascara	1 1 0 1 0 1 1 0
Hijo 1	1 0 0 1 1 0 1 0
Hijo 2	0 0 1 1 0 0 1 1

Figura 7: Cruzamiento uniforme

- Cruzamiento PMX (Patially Mapped Crossover): Consiste en elegir aleatoriamente dos puntos de cruce para luego intercambiar estos 2 segmentos en los hijos que se generan (como el cruce de 2 puntos convencional). El resto de las cadenas que conforman los hijos se obtienen haciendo mapeos entre los 2 padres teniendo en cuenta que si un valor no está contenido en el segmento intercambiado, permanece igual, mientras que si está contenido en el segmento intercambiado, entonces se sustituye por el valor que tenga dicho segmento en el otro padre. Esto se muestra en la Figura 8. En primer lugar se determinan los elementos de los hijos sin tener en cuenta los elementos conflictivos. Como segundo paso para completar los hijos, se copian primero los valores que no están en el segmento intercambiado. Finalmente se mapean los valores restantes.

Padre 1	9 8 4 | 5 6 7 | 1 3 2 10
Padre 2	8 7 1 | 2 3 10 | 9 5 4 6
Hijo 1	x x x | 2 3 10 | x x x x
Hijo 2	x x x | 5 6 7 | x x x x
Hijo 1	9 8 4 | 2 3 10 | 1 x x x
Hijo 2	8 x 1 | 5 6 7 | 9 x 4 x
Hijo 1	9 8 4 | 2 3 10 | 1 6 5 7
Hijo 2	8 10 1 | 5 6 7 | 9 2 4 3

Figura 8: Cruzamiento PMX

3.7 Operadores evolutivos: mutación

La mutación es la operación que a partir de un individuo genera otro mediante el cambio de una parte del genotipo del individuo [5]. La mutación impide que el algoritmo quede atrapado en mínimos locales. La mutación desempeña el papel de la recuperación ante la pérdida de material genético [11]. La mutación se ha considerado tradicionalmente como un operador de búsqueda simple. El cruzamiento fomenta la explotación de la solución actual para encontrar otras mejores, mientras que la mutación contribuye a la exploración del espacio de búsqueda. La mutación es vista como un operador que contribuye a mantener la diversidad genética en la población. De esta forma es que se introducen nuevas estructuras genéticas en la población de forma aleatoria. Hay muchas formas diferentes de mutación que dependen de los diferentes tipos de representación.

Para la representación binaria, una mutación simple puede consistir en invertir el valor de cada gen con una probabilidad pequeña. La probabilidad se toma generalmente cerca de 1 / L, donde L es la longitud del cromosoma. También es posible implementar operadores de mutación que se realicen sólo si se mejora la calidad de la solución. Este tipo de operadores puede acelerar la

búsqueda, pero debe tenerse cautela ya que también podría reducir la diversidad de la población y esto hace que el algoritmo converja hacia óptimos locales. Algunos ejemplos de técnicas de mutación son:

- Mutación de cambio de bit: dicho método de mutación implica el cambio de 0 por 1 y 1 por 0. La Figura 9 muestra un ejemplo de este operador. Para un cromosoma 1 en la mutación, el bit correspondiente en el cromosoma padre se cambia (0-1 y 1-0).

Padre	10110101
Cromosoma de mut:	10001001
Hijo	00111100

Figura 9: Mutación de cambio de bit

- Intercambio: dos posiciones aleatorias de la cadena son elegidos y los bits correspondientes a dichas posiciones se intercambian. Esto se muestra en la Figura 10.

Padre	10110101
Hijo	11110001

Figura 10: Mutación de intercambio

3.8 Función de fitness

La primera etapa de un algoritmo evolutivo consiste en la selección. Esta se realiza a partir de los valores de la función de fitness o de aptitud de cada individuo. Cada cromosoma tiene asociado un valor correspondiente a la idoneidad de la solución que representa. La aptitud o fitness corresponde a la evaluación de que tan buena es una solución. La solución óptima es la que maximiza la función de aptitud. En el caso de que el problema consista en la reducción de una función de costos, la adaptación es muy fácil, la función de costo se puede transformar en una función de aptitud, por ejemplo invirtiéndola.

Capitulo 4 Planteo y resolución del problema utilizando AEs

Esta sección presenta la formulación genérica del modelo de conversión utilizado para mejorar la calidad de las imágenes, ofrece una breve reseña de trabajos relacionados con la aplicación de AE al problema de reconocimiento de caracteres y presenta el planteo del problema y su resolución mediante la utilización de AEs.

4.1 Formulación del problema

El problema de reconocimiento de texto en imágenes con ruido consiste en analizar una imagen distorsionada y extraer los caracteres de texto que están presentes. La distorsión en la imagen puede ser provocada por el deterioro o suciedad del papel si se trata de imágenes escaneadas o provocada artificialmente para dificultar el reconocimiento automático de caracteres, si se trata de CAPTCHAS.

En este trabajo se consideran solamente CAPTCHAS. Sin embargo, el enfoque planteado es general y podría ser aplicado sin cambios sobre imágenes escaneadas con ruido.

4.2 Antecedentes: AE aplicados al reconocimiento de imágenes

Los algoritmos evolutivos no han sido muy usados para el reconocimiento de imágenes o tareas similares. Refiriéndonos a la utilización de técnicas similares para el reconocimiento de imágenes la primera cita corresponde al trabajo acerca de "Restoration of old documents with genetic algorithms" [8]. Fue realizado en la Universidad de Coruña y plantea la restauración de documentos antiguos mediante el uso de algoritmos genéticos. El objetivo del trabajo fue utilizar un algoritmo genético con el fin de poder discriminar dentro de la imagen, a manera de filtro, lo que corresponde a letras y al fondo. Se obtuvieron buenos resultados, mejorando la legibilidad del texto original.

El segundo trabajo relevado es "Seeing the character images that an OCR system sees - analysis by genetic algorithm" [9]. Se enfoca en utilizar algoritmos genéticos para analizar el espacio de soluciones del sistema, mediante la visualización de las formas de las imágenes de caracteres que corresponden a los vectores con que el OCR los reconoce, de una manera que los humanos puedan comprender. Los experimentos realizados mostraron que 100 generaciones fueron necesarias para que las soluciones convergieran logrando de forma exitosa la reconstrucción de imágenes utilizando algoritmos genéticos y aumentando la tasa de reconocimiento.

4.3 Modelo de conversión para OCR

La resolución del problema de reconocimiento de texto en imágenes con ruido puede ser separado en dos etapas. En la primera etapa se realiza una transformación de la imagen con perturbaciones de modo de mejorar su legibilidad, mientras que en la segunda etapa se realiza el reconocimiento del texto de la imagen transformada.

En la Figura 11 se presenta un ejemplo de la metodología planteada. Inicialmente la imagen que contiene el texto tiene ruido que dificulta su reconocimiento. A continuación, se le aplica un

proceso de transformación obteniendo una imagen más "limpia". Finalmente, se utiliza una herramienta de reconocimiento OCR para obtener el texto de la imagen transformada.

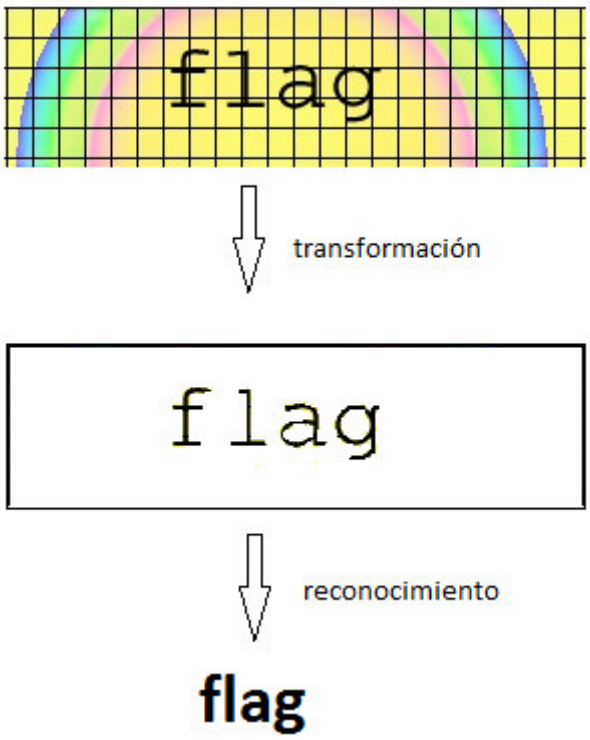


Figura 11: Modelo de conversión.

Para realizar el proceso de transformación y reconocimiento de imágenes se utilizan dos herramientas de software libre sobre el sistema operativo GNU/Linux. La herramienta *convert*, que forma parte de la librería ImageMagick [21], es utilizada para mejorar la imagen, mientras que para el reconocimiento de la imagen transformada se utiliza OCR Tesseract [12]. Ambas herramientas fueron cuidadosamente seleccionadas a partir de las diversas experiencias previamente detalladas en el Capítulo 2.

El modelo de conversión trabaja con *n* funciones $f_i \subset S$ y considera la función de transformación dada por la composición:

$$\prod_{i=1}^{n} f_i(p_1, \dots, p_{K_i})$$

donde p_i son los parámetros asociados a cada función de transformación f_i. Los valores de p_i son números naturales definidos en un rango de valores mínimo y máximo para cada p_i. El problema de conversión consiste en determinar un subconjunto de funciones de conversión, su orden de aplicación y los valores paramétricos que permitan mejorar la calidad de una imagen, para ser utilizada eficazmente por una herramienta de reconocimiento de caracteres. Para determinar el subconjunto de funciones de conversión, el orden de aplicación y los valores paramétricos se utiliza un AE que se presenta en la próxima sección.

El conjunto *S* está compuesto por 12 de las funciones de transformación que posee la herramienta

convert, seleccionadas de acuerdo a criterios de experiencia y experimentación antes mencionados en el Capítulo 2, como aquellas que proporcionan resultados visibles útiles para el reconocimiento de caracteres. A modo de ejemplo, las funciones que segmentan la imagen o cambian totalmente su color se descartan en el modelo de conversión, ya que su aplicación no sería útil para el reconocimiento. El conjunto *S* incluye a las funciones que pertenecen a la Tabla 4.

Función	Descripción	Parámetros	Rango
grayscale	Convierte el espacio de colores de la imagen a escala de grises.	No tiene.	0..1
threshold	Aplica un umbral sobre blanco y negro de forma tal que maximiza los valores de los pixeles que superan dicho umbral y minimiza al resto.	Umbral para la aplicación del filtro.	5..99 %
paint	Extiende los trazos de color de la imagen.	Factor que aumenta o disminuye la dilución de los colores.	1..4
negate	Modifica cada pixel por su valor opuesto en la imagen.	No tiene.	0..1
noise	Adhiere o remueve ruido a la imagen.	Factor que modifica el efecto del ruido sobre la imagen.	0..3
equalize	Ecualiza los colores de la imagen según su histograma.	No tiene.	0..1
blur	Reduce los niveles de ruido y detalle en la imagen.	Factor que define la potencia de la función aplicada.	1..4
charcoal	Realiza un efecto de filtro de tipo “carbón” sobre la imagen.	Radio utilizado como factor de ajuste para el efecto.	1..5
colorize	Colorea la imagen con el valor especificado.	Valor que define el color a utilizar.	20..70 %
posterize	Limita el número de niveles de colores de la imagen.	Número de niveles de colores de la imagen.	2..7 %
level	Ajusta los niveles de contraste de la imagen.	Factor de ajuste para los niveles de contraste de la imagen.	20..100 %
fuzz	Considera iguales a los colores que se encuentran a una determinada distancia en la imagen.	Valor de la distancia mínima entre los colores de la imagen.	20..100 %

Tabla 4: Funciones seleccionadas

4.4 Resolución del problema con Algoritmos Evolutivos

El AE utiliza una codificación que representa la secuencia de funciones a aplicar en el proceso de conversión de las imágenes. Para cada función del subconjunto S' se codifican tres componentes: los valores de los parámetros de la función, un valor booleano que indica si el comando se aplica o no en la conversión, y un valor entero que indica el orden de aplicación del comando en el proceso.

Un ejemplo de esta codificación se presenta en la Figura 12.

valores paramétricos				aplicación (si/no)				orden de aplicación			
0	0	12500	0	1	1	1	0	3	1	2	0

Figura 12: Ejemplo de codificación del modelo de conversión.

Como ejemplo, para un caso que utilice $n = 4$ y el subconjunto S' = {type grayscale, paint, threshold, noise}, la codificación del individuo corresponde al vector presentado en la Figura 13.

I =	paint: 0	threshold: 12500	type grayscale

Figura 13: Vector de parámetros que representa a un individuo.

Cuando se aborda el reconocimiento de una clase determinada de imágenes, el usuario puede especificar en un archivo de configuración un subconjunto $S' \subseteq S$ que sea útil para el tipo de imágenes en cuestión, y rangos alternativos para los valores de los parámetros. De este modo se puede reducir el espacio de búsqueda del problema utilizando información adicional, que puede provenir de la experiencia y/o de análisis empíricos previos. Si no se especifican estas limitaciones, el modelo de conversión considerará las 12 funciones incluidas en *S* y los rangos predefinidos para los valores de los parámetros.

A continuación se presenta la función de fitness y los operadores evolutivos utilizados y el proceso que se llevó a cabo para determinar una buena función de fitness para este problema.

4.4.1 Función de Fitness

La función de fitness pretende medir la exactitud con que fueron reconocidos los caracteres de la imagen y se tendrá por objetivo maximizar esa medida. Este punto no es trivial ya que no está claro y puede depender del problema en particular, que soluciones son más o menos similares que otras al texto original. Por ejemplo, si el texto original es “creativity” ¿se debe considerar que “crea.tivity” es más similar que “CreaTivity” o “craativity”?.

Un enfoque simple consiste en que luego de haber reconocido una serie de caracteres en la imagen, se comparan con el texto que realmente contiene la imagen, sumando un cierto valor por cada carácter en que haya coincidencia. Este enfoque presenta grandes deficiencias ya que compara posición por posición. Es decir que si el carácter reconocido en una cierta posición no coincide con el real no se suma puntos. Esto generaría que, por ejemplo, una palabra reconocida perfectamente pero con un carácter extraño al principio tenga fitness nulo.

Una mejora a dicho enfoque es asignar un valor si el carácter reconocido pertenece al conjunto de caracteres reales (sin tener en cuenta posición). Esta mejora soluciona algunos problemas pero incorpora otros. Si la palabra tiene las mismas letras en distintos lugares puede llegar a obtener el fitness máximo para esa palabra sin que esta palabra coincida con la real. Este hecho interfiere fuertemente en que el algoritmo evolutivo pueda encontrar mejores soluciones. Por ejemplo, en el caso de que la palabra real fuera “pearl” y el algoritmo reconoce “pelar” le asignaría el fintess

máximo.

Una tercera aproximación a la definición de la función de fitness es asignar un valor a cada letra que coincida con la letra real en la posición exacta o en una posición muy próxima. De esta forma, es posible encontrar la letra en la posición correcta o con un corrimiento pequeño (entre 1 y 5 posiciones) y de todas formas incrementar el valor de la función de fitness. Varios experimentos se debieron realizar para estimar el mejor valor para el máximo de posiciones a tener en cuenta. Determinar dicho valor, no es una tarea trivial, ya que depende fuertemente del motor OCR que se utilice y de observar cómo se comporta cuando no se reconocen los caracteres o cuando reconoce caracteres distintos a los correctos. En la Figura 14 se ilustran estos aspectos. La imagen pauta el comportamiento del motor OCR en el reconocimiento de la misma, en este caso se puede apreciar caracteres extraños reconocidos, como también se aprecian letras reconocidas correctamente en posiciones incorrectas, quedando claramente expuesto la no trivialidad de la definición de la función de fitness.

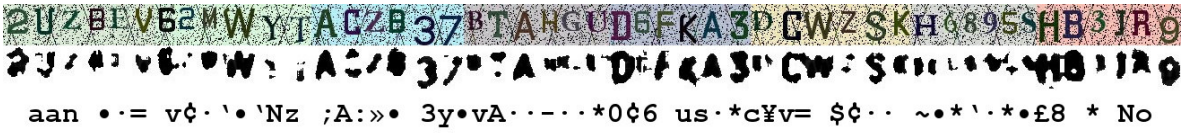


Figura 14: Contiene la imagen original, luego la transformada como solución de una ejecución del algoritmo y por último los caracteres reconocidos por el OCR en ese caso.

Otra mejora en la definición de la función consiste en asignar un peso al valor que se suma al fitness dependiendo de que el carácter se encuentre exactamente en la posición correcta o desplazado cierta distancia. Es decir que lo que más suma es que el carácter esté exactamente en la posición esperada. Si la coincidencia es a una posición de distancia aporta un valor menor. El valor es menor aún si está a dos posiciones y así sucesivamente. La determinación de los mejores pesos involucró una gran experimentación de modo de cubrir una gran variedad de casos.

Por último, se incorpora la suma de un valor extra cuando la palabra se reconoce perfectamente. Este agregado permite evitar que las soluciones incorrectas tengan un fitness mayor que la solución correcta. Por ejemplo, si la palabra real fuera "breath", y se reconociera la palabra "breathh" tendría un fitnes mayor que "breath", ya que la segunda 'h' aumenta el fitness por ser un carácter correcto que esta desplazado un lugar. Al sumar un valor extra cuando se reconoce la palabra exacta se evita ese tipo de comportamientos de la función de fitness.

La idea de sumar un valor extra al reconocer la palabra perfectamente se implementó de una forma particular, con una resta en vez de una suma. Al finalizar el cálculo del fitness, al mismo se le resta la diferencia entre la cantidad de caracteres de la palabra reconocida y la original. De esta forma se logra que su resultado dependa de la cantidad de caracteres de la imagen tratada. Si se utilizara directamente una constante, no tendría distinto peso al evaluar imágenes con cantidades de caracteres diferentes.

Para la evaluación de la función de fitness es necesario que el nombre del archivo que contiene la imagen sea el texto que está presente en la imagen. Por ejemplo, una imagen que contiene los caracteres “quantum” debe ser almacenada en un archivo con el nombre “quantum.jpg”. El motor OCR genera un archivo de texto con los caracteres reconocidos, por lo que la función de fitness debe leer este archivo y comparar la información que contiene con el nombre del archivo de partida. El pseudo-código que implementa la función de fitness se muestra en la Figura 15.

```
FUNTION FITNESS (FileName: String , OCRText: String)

INT const1 >  const2  >  customValue1 > customaValue2

 FOREACH char1 in FileName
   FOREACH char2 in OCRText
     IF EQUAL (char1, char2)
         retValue += const1
   ELSE
      IF EQUAL (TO_LOWER (char1), TO_LOWER ( char2))
        retValue += const2
   ELSE
       IF EQUAL_IN_OTHER_POSITION (char1, char2)
           retValue +=  customValue1
   ELSE
       IF EQUAL_IN_OTHER_POSITION (TO_LOWER (char1), TO_LOWER ( char2))
           retValue +=  customValue2

   END
 END
END
retValue -=  DIF_LENGTH(FileName, OCRText)
```

Figura 15: Pseudo-código función de fitness.

4.4.2 Operador de cruzamiento

Para el caso de la codificación presentada, se utiliza un operador de cruzamiento que trabaja en forma distinta en cada uno de los sectores de la solución ya mencionados anteriormente.

Para el primer sector, el operador de cruzamiento consiste en cruzar los valores de los parámetros de dos soluciones, de manera que la solución descendiente tendrá como parámetros un nuevo valor dado por una distribución normal con media el valor promedio entre los valores de los parámetros de las soluciones generadoras, ponderando de forma positiva el parámetro perteneciente a la solución con mayor fitness y cuyo desplazamiento, es igual al doble del intervalo considerado para dicho parámetro. La ponderación se realiza duplicando el valor del parámetro perteneciente a la solución con mayor fitness de modo de aumentar las posibilidades de influir al descendiente con los valores del padre de mejor aptitud.

Para el segundo sector, el cual únicamente posee valores binarios, el operador de cruzamiento consiste en un cruzamiento simple de un punto entre los valores del sector en cuestión.

Para el tercer sector y dado que los valores son un conjunto al cual se le aplican únicamente permutaciones, el cruzamiento debe tener en cuenta no generar valores repetidos para las soluciones

descendientes, por tanto no se pudo aplicar un cruzamiento simple y fue necesario implementar un cruzamiento de tipo PMX (partial mapped Crossover). PMX en primer lugar aplica un cruzamiento de dos puntos y al conjunto solución obtenido, le realiza una corrección sustituyendo valores repetidos de forma de que el conjunto solución resultante sea una permutación del primero.

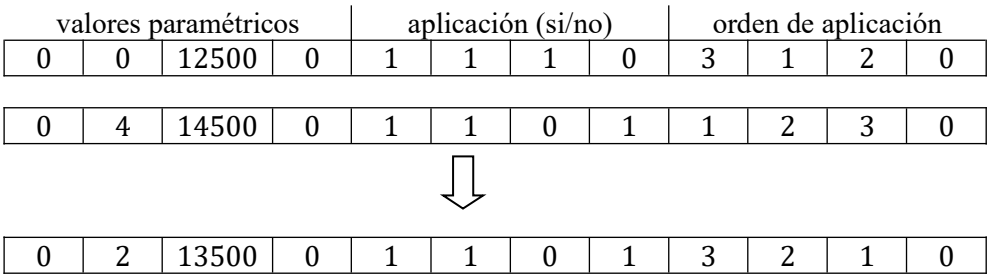


Figura 16: Ejemplo de operador de cruzamiento

4.4.3 Operador de mutación

De manera similar también se utiliza un operador de mutación que trabaja en forma distinta para cada uno de los sectores de la solución.

Para el primer sector (valores numéricos de los parámetros) el operador de mutación cambia uno de los valores de los parámetros por un nuevo valor seleccionado aleatoriamente de acuerdo a una distribución normal con media el valor previo a la mutación y con valor de desplazamiento igual al doble del intervalo considerado para dicho parámetro. Ejemplificando, para una variable que mantiene sus valores en el intervalo [0,3,6,9…..] el valor del desplazamiento será 6. De esta forma el valor de la solución mutada no presenta cambios radicales con respecto a sus valores previos.

Para el segundo sector (valores indicadores de uso o no uso de parámetros) el operador de mutación consiste simplemente en cambiar uno de los valores seleccionado aleatoriamente por su valor de verdad opuesto.

Para el tercer sector (valores de posición en que es aplicado cada parámetro) el operador de mutación consiste en realizar un intercambio entre dos valores del sector, de modo de mantener el mismo conjunto de valores inicial pero modificando su posición en el conjunto, siendo utilizados a modo de permutación.

En la Figura 17 queda detallado el proceso de mutación de un individuo. En el caso planteado para el primer sector el elemento sorteado es el que ocupa la primera posición. Dicho elemento que tenía valor 0 cambio su valor por 2, este último número elegido aleatoriamente a partir de una distribución normal como es explicado previamente en este capítulo. Para el segundo sector la mutación correspondió simplemente a modificar el elemento en la posición 3 que en principio era 1 por su valor opuesto 0. Por último, en el sector 3, se realizó un intercambio de 2 valores, los mismos ubicados en las 2 primeras posiciones del arreglo, se permutaron de modo de pasar de los valores 3 1 a los valores 1 3, respectivamente para cada uno de los dos individuos.

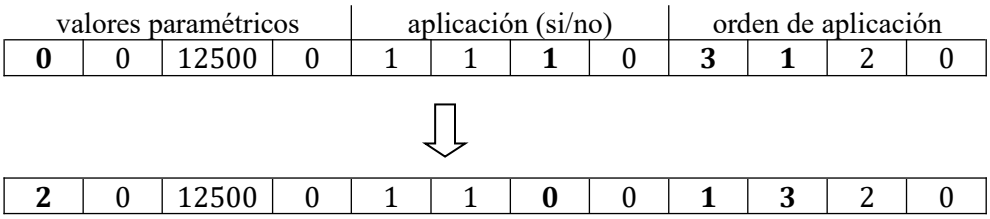


Figura 17: Ejemplo de operador de mutación

4.4.4 Operador de selección

Para el caso de la codificación presentada, el operador de selección usado no posee características especiales. Para este caso se utilizó uno de los operadores provistos por MALLBA que consiste en la estrategia de selección de la ruleta.

4.5 La biblioteca MALLBA

MALLBA es una biblioteca de algoritmos para optimización capaz de tratar eficientemente con el paralelismo, manteniendo una interfaz amigable para el usuario [1]. El AE presentado en este trabajo está implementado en MALLBA como un esqueleto de software, que debe ser instanciado con las características del problema por parte del usuario. En MALLBA, los métodos de resolución, la interacción con el problema y el paralelismo se implementan mediante clases C++ requeridas y provistas que abstraen a las entidades involucradas en cada método de resolución:

• Clases provistas: implementan aspectos internos del algoritmo, de un modo independiente al problema. Las principales clases provistas son Solver (el algoritmo) y SetUpParams (para fijar parámetros).

• Clases requeridas: especifican la información del problema a resolver. Cada esqueleto incluye las clases requeridas Problem y Solution que encapsulan entidades dependientes del problema, necesarias para los métodos de resolución.

La infraestructura del proyecto MALLBA se compone de clusters de computadores localizados en Málaga, La Laguna y Barcelona en España, conectados por redes de comunicaciones de alta velocidad. La biblioteca MALLBA está disponible en http://neo.lcc.uma.es/mallba/easy-mallba.

Capitulo 5 Resultados experimentales

Este capítulo describe los experimentos realizados para evaluar formalmente el algoritmo bajo distintas condiciones y niveles de dificultad. En primer lugar se presentan los conjuntos de imágenes elegidos. Luego se describe la plataforma utilizada y una primer calibración de los parámetros del algoritmo. Con los resultados de dicha calibración se describe el proceso que fue realizado en pos de disminuir las funciones de transformación utilizadas por el AE y así reducir el espacio de soluciones. Una vez obtenido un conjunto definitivo de funciones de transformación se calcula una aproximación al tamaño del espacio de soluciones y se procede con calibraciones formales para la selección de los parámetros del AE. El primer experimento de calibración formal busca encontrar los parámetros óptimos para los valores de probabilidad de cruzamiento y mutación y el segundo el tamaño de población y cantidad de iteraciones. Una vez calibrados los parámetros del AE, se procede con los experimentos de validación del AE. Se comienza comparando los resultados del AE con una búsqueda aleatoria, luego se compara con otro proyecto de investigación y por último se analiza que tan bien generaliza una solución obtenida con una imagen aplicada a otras del mismo tipo. Para finalizar, se efectúa un análisis de eficiencia computacional y se presentan observaciones interesantes encontradas durante los diferentes experimentos.

5.1 Conjunto de imágenes

En esta sección se presentan imágenes representativas de cada conjunto y las características por las cuales fueron elegidas.

Es importante destacar que la selección de captchas como fuente para las pruebas se debe principalmente a que presentan muy variados tipos de dificultades y a la facilidad que presenta obtenerlas ilimitadamente desde internet. Si bien el algoritmo de este proyecto en muchos casos es útil para romper captchas o evaluar la efectividad de los mismos, ese no es su objetivo. Las dificultades que se observan en dichas imágenes, son en la mayoría de los casos, ampliamente superiores a las existentes en documentos que se pretenden digitalizar. Esto es debido a que el propósito de los captchas es que sólo puedan ser reconocidos por humanos y no por motores OCR u otras técnicas.

Los conjuntos de imágenes utilizados en la evaluación son Aardvark Topsites PHP, BotDetect CAPTCHA Online Demo, PHP Link Directory, Breaking a Visual CAPTCHA y Sitio web LoQuo. A continuación se presentan los distintos conjuntos considerados.

Aardvark Topsites PHP

Aardvark Topsites PHP es un framework ampliamente utilizado en la actualidad para generar sitios web que proveen funcionalidades de ranking y revisión de otros sitios web [17]. Las imágenes contenidas en este conjunto son las que este framework utiliza como captchas. Dichas imágenes presentan dificultades interesantes ya que contienen una cantidad variable de números y letras de distintos tipos, colores e inclinaciones. Además contiene fondos de diferentes colores con más caracteres del mismo tipo que los de la imagen pero con menor contraste.

Figura 18: Imágenes del CAPTCHA utilizado por Aardvark Topsites PHP

BotDetect CAPTCHA Online Demo, LANAP Software

El software analizado en este caso provee distintos tipos de captchas para ser utilizados en aplicaciones web [22]. Se seleccionó un tipo de estas imágenes que contiene dificultades debido al poco contraste entre los caracteres y el fondo e interferencia sobre toda la imagen. Vale la pena destacar que estas dificultades son similares a las que se podrían encontrar en un documento viejo digitalizado o digitalizaciones de mala calidad.

Figura 19: Imágenes del CAPTCHA presente en la demo de BotDetect CAPTCHA

PHP Link Directory

PHP Link Directory es uno de los frameworks más utilizados para implementar foros online [23]. Las imágenes que utilizan como captchas poseen dificultades interesantes para este proyecto. Además de los diferentes colores de los caracteres y los fondos, se observan dos tipos de interferencias sobre toda la imagen, líneas curvas que pasan a través de las letras y muchos puntos o ruido de fondo.

Figura 20: Imágenes del CAPTCHA utilizado en PHP Link Directory

Proyecto Breaking a Visual CAPTCHA

Este proyecto de la Universidad de Berkeley, California, desarrolló y experimentó con técnicas de reconocimientos de objetos para lograr reconocer diferentes captchas con distintos grados de dificultad generadas con el motor Gimpy utilizado por Yahoo [6]. Las imágenes de este conjunto poseen una gran variedad de interferencias y dificultades. Entre ellas se pueden destacar las líneas rectas sobre toda la imagen del mismo color que los caracteres, poco contraste entre caracteres y fondo, caracteres borrosos, fondos con gran cantidad de interferencias de alto contraste y letras levemente curvadas e inclinadas.

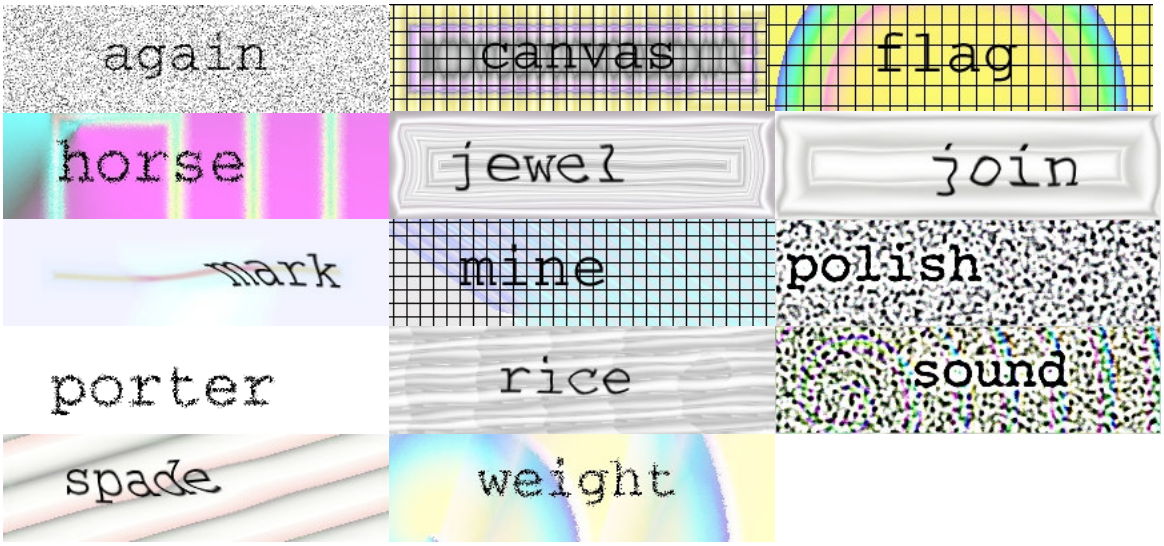


Figura 21: Imágenes del proyecto Breaking a visual captcha

Sitio web LoQuo

Loquo es un sitio web de compra y venta de artículos nuevos y usados con millones de usuarios, utiliza captchas para prevenir el abuso por parte de robots [16]. En las imágenes de sus captchas se observan distintos colores de caracteres, algunos con muy poco contraste con respecto al fondo, inclinaciones y desalineadas horizontalmente. Además contienen líneas de fondo, en algunos casos muy similares a los colores de alguna letra.

Figura 22: CAPTCHAs utilizados en el sitio web LoQuo

5.2 Plataforma de ejecución

Los experimentos fueron realizados en las PCs de las salas UNIX de la Facultad de Ingeniería. Dichos equipos poseen un procesador Intel Pentium IV a 2.80GHz, con 1 GB RAM y sistema operativo Linux Fedora 10. El software utilizado incluyó a MPICH 2 versión 1.0.7, ImageMagick versión 6.4.0 y Tesseract versión 2.03.

Todas las ejecuciones fueron hechas utilizando el comando *nice*, de esta forma quedaban en background con la mínima prioridad para no interferir con el correcto funcionamiento de dichas PCs ya que son utilizadas al mismo tiempo por otros estudiantes.

Para minimizar los tiempos de ejecución, el algoritmo se implementó de forma tal de poder ejecutarse al mismo tiempo en distintos directorios utilizando diferentes archivos de configuración.

Así se utilizaron decenas de PCs al mismo tiempo para los experimentos que lo requerían. Se desarrollaron scripts que automáticamente recorren todas las *pcunix* y ejecutan el algoritmo en las que no tengan usuarios conectados ni procesos en background consumiendo CPU.

5.3 *Experimentos de calibración y refinamiento de la elección de las funciones de conversión*

Los parámetros de los AEs son muy importantes, ya que influyen directamente en la calidad de las soluciones obtenidas por el algoritmo. Estos parámetros pueden ser estudiados independientemente, pero el desempeño global del algoritmo no depende exclusivamente de un único parámetro sino de la combinación de todos los parámetros. La calibración de parámetros es una técnica para encontrar los mejores valores para los parámetros estudiados del algoritmo genético antes de correr el algoritmo y luego configurar el algoritmo con ellos. Los valores de los parámetros para un problema pueden no son los mejores para otro problema, es por eso que dicha calibración debe hacerse para cada problema en particular que se resuelve [5].

En esta etapa del proceso y dado el costo computacional que conlleva para el algoritmo la utilización de cada una de las herramientas de transformación, la etapa de calibración también será de utilidad para descartar aquellas funciones de transformación que en primer término podían ser consideradas como útiles, pero que desde el punto de vista práctico no lo resultan.

5.3.1 *Calibración para una imagen en particular*

Para los experimentos de calibración usando una imagen en particular, se utilizó la imagen "canvas" que se presenta en la Figura 23. Se optó por dicha imagen porque posee un nivel de dificultad adecuado para realizar estos experimentos, esto es, su reconocimiento no es trivial pero tampoco requiere una cantidad excesiva de iteraciones o un tamaño grande de población para que el algoritmo encuentre una solución óptima. Esto se sabe porque previo a los experimentos formales, se han realizado gran cantidad de experimentos informales con esta y otras imágenes.

Figura 23: Imagen con dificultad adecuada para este experimento

Hubo que buscar un equilibrio entre la elección de la imagen, la cantidad de iteraciones y el tamaño de población del algoritmo. O sea, una imagen muy compleja requeriría cantidad de iteraciones e individuos mayores, lo que generaría tiempos de ejecución demasiado altos, considerando las limitaciones de procesamiento y tiempo con que se cuenta, para la cantidad de ejecuciones independientes que se deben hacer.

Los parámetros estudiados en este caso son el tamaño de población (#pop) y las probabilidades de cruzamiento (p_C) y mutación (p_M). La cantidad de iteraciones fue igual a 80 para todas las ejecuciones. Los valores candidatos para cada uno de los parámetros correspondieron a:

- #pop: 10, 50 y 100 individuos.
- p_C: 0.75, 0.85, 0.95.
- p_M: 0.01, 0.15, 0.3.

El análisis de los resultados consiste, en primer lugar, en el desarrollo de scripts que permitan obtener los valores de interés como parámetros del algoritmo y fitness alcanzados por las soluciones en un formato fácilmente importable en una planilla de cálculo. Con esos valores se procede al armado de tablas que permiten analizar los resultados de las diferentes combinaciones. Son consideradas mejores combinaciones de parámetros, las que generan, en promedio, soluciones con mayores fitness. En los casos en que las diferencias entre los promedios son pequeñas, se procede a analizar con mayor detalle las propiedades de cada ejecución, como por ejemplo, cantidad de casos en los que se encuentra la solución óptima, tiempos de convergencia a la solución encontrada y en que iteración, promedio del fitness de la población total, etc. En los casos en que los fitness promedio de una combinación de parámetros, superan mínimamente a otra en la que se encuentra una cantidad de veces mayor la solución óptima, se elige como mejor esta última.

Analizar los resultados de las 27 combinaciones posibles de los valores paramétricos en 10 ejecuciones independientes del AE para cada una, mostró que los mejores resultados de reconocimiento se obtuvieron para la configuración paramétrica determinada por #pop = 100 individuos, p_C: 0.85 y p_M: 0.15. Una observación importante es que, en los experimentos con poblaciones menores (10 y 50 individuos) los mejores resultados se obtuvieron con una probabilidad de mutación igual a 0.3. Esto permite formular la hipótesis de que con tamaños de población bajos, que abarcan una porción muy pequeña del espacio de soluciones, una probabilidad de mutación alta ayuda a recorrer puntos más lejanos del espacio de soluciones y así se encuentran mejores soluciones. En cambio, utilizando una población mayor que cubre más al espacio de soluciones de las posibles transformaciones de las imágenes, una mutación tan alta no es necesaria y se obtienen mejores resultados con una menor. Poder concluir esto inequívocamente requeriría un experimento particular para verificar o refutar esa hipótesis.

En la Tabla 5 se muestran algunas de las soluciones con fitness óptimo de estos experimentos.

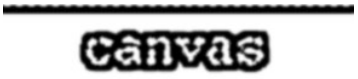
Imagen transformada	**Solución hallada**
	charcoal 4 | paint 3 | blur 4 | noise 0 | posterize 4 | threshold 85% | type grayscale | negate | level 78% | fuzz 57%
	threshold 46% | paint 2 | fuzz 21% | level 99% | posterize 3 | blur 1 | equalize
	threshold 47% | paint 2 | blur 1 | posterize 4 | charcoal 5

Tabla 5: Algunas soluciones óptimas de este experimento

Los tiempos de ejecución fueron muy variables ya que se utilizaron PCs compartidas y se ejecutó el algoritmo con una prioridad muy baja para no interferir con el correcto funcionamiento de las PCs de parte del resto de los usuarios. En promedio, cada ejecución en los casos de poblaciones iguales a 10 individuos demoró 1 hora y 20 minutos, en los casos de 50 individuos 8 horas y en los casos de 100 individuos 13 horas.

5.3.2 Calibración considerando varias imágenes

Esta etapa tiene como objetivo evaluar si una calibración es generalizable para los distintos grupos de imágenes, o por el contrario se está ante una situación en la que será necesario realizar diferentes calibraciones para cada uno de los tipos de imágenes.

La evaluación se realiza ejecutando varias veces el algoritmo, una por cada grupo de imágenes, utilizando para cada ejecución una imagen representativa del grupo al cual pertenece que se presentan en la Tabla 6. La selección de dicha imagen, se realiza en función de la experiencia de las diversas pruebas realizadas, teniendo como criterio de selección, elegir a aquellas imágenes que recojan las diversas dificultades de reconocimiento que posee el grupo al cual pertenecen.

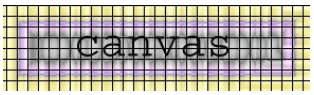
Conjunto	Imagen utilizada
Aardvark Topsites PHP	
BotDetect CAPTCHA Online Demo	
PHP Link Directory	
Breaking a Visual CAPTCHA	
Sitio web LoQuo	

Tabla 6: Imágenes representativas de cada conjunto

En este caso, los parámetros del algoritmo evolutivo son distintos a los resultantes del experimento anterior por dos razones. En primer lugar, en este experimento no se eligen en pos de encontrar soluciones óptimas para cada conjunto de imágenes, sino que permitan obtener buenos resultados para todos los conjuntos. Además se tuvieron que considerar los tiempos de ejecución requeridos para este experimento y así adecuar el tamaño de población y cantidad de iteraciones en pos de finalizar en plazos razonables. Se realizó un estudio exhaustivo para determinar, en base a la dificultad de las imágenes elegidas, población, iteraciones, probabilidades de cruzamiento y mutación, de forma tal de no llegar a resultados óptimos fácilmente pero tampoco finalizar con resultados notablemente deficientes. Esto es porque dichos casos no permitirían probar ni refutar la hipótesis de este experimento.

Los parámetros utilizados fueron:

- Población: 50
- Iteraciones: 30

- Probabilidad de
    - Cruzamiento: [0.7, 0.8, 0.9]
    - Mutación: [0.01, 0.05, 0.1]

Para cada una de las 5 imágenes representativas de su conjunto, se tienen 9 combinaciones diferentes de parámetros, por cada una se realizan 10 ejecuciones independientes del algoritmo.

Los resultados obtenidos son ampliamente satisfactorios. En primer lugar, demuestran que los valores de los parámetros utilizados dieron resultados positivos para los diferentes grupos de imágenes. Aunque no es completamente asegurable, los resultados indican que el comportamiento del sistema a partir de los parámetros establecidos de calibración, puede ser generalizable a nuevos grupos de imágenes. Esto último no significa que una combinación de parámetros sea la óptima para distintos tipos de imágenes, sino que será relativamente buena en casos diferentes; pero en base a la experiencia obtenida se puede suponer que distintos tipos de imágenes tendrán diferentes combinaciones de parámetros óptimas para el algoritmo evolutivo. Expresado de otra forma, conjuntos de imágenes muy diferentes tienen configuraciones de parámetros óptimas diferentes por lo que hay que realizar estudios particulares en cada caso; sin embargo, se pueden encontrar configuraciones que sin llegar a ser óptimas, son buenas en conjuntos de imágenes muy variados. Otra conclusión no menos importante, tuvo que ver con la regularidad de las soluciones encontradas. Tomando en cuenta cada una de las mejores soluciones encontradas para cada uno de los grupos de imágenes, los resultados obtenidos fueron regulares en correctitud, sin existir soluciones muy alejadas de las de mejor fitness. Por último, se pudo reconocer que las mejores soluciones, en los diferentes grupos de imágenes utilizaron de forma regular todo el conjunto de funciones disponibles para las transformaciones.

La Tabla 7 muestra soluciones óptimas encontradas por el algoritmo para cada imagen. Como se ha visto anteriormente, no existe una única solución óptima sino múltiples.

Imagen transformada	Solución óptima
	level 55% | paint 2 | equalize | charcoal 5 | blur 1 | threshold 64% | posterize 2 | fuzz 29%
	type grayscale | level 57% | posterize 2 | negate | blur 3 | threshold 45% | charcoal 5 | paint 1
	fuzz 36% | blur 2 | negate | threshold 26% | charcoal 3
	negate | fuzz 50% | blur 3 | noise 0 | type grayscale
	paint 1 | threshold 59% | posterize 4 | blur 1 | colorize 20% | type grayscale | level 64% | fuzz 70%

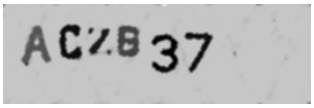
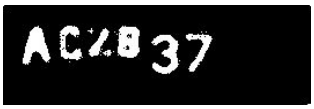
	negate | threshold 36% | fuzz 27% | colorize 70% | charcoal 5 | level 35% | equalize
	noise 0 | level 80% | colorize 20% | equalize | paint 1 | blur 1 | fuzz 24% | negate | type grayscale
	type grayscale | threshold 75% | paint 2 | blur 1 | noise 3 | level 20% | negate | fuzz 53% | posterize 2
	level 94% | posterize 2 | colorize 70% | type grayscale | blur 1
	threshold 42% | level 100% | negate | blur 3 | colorize 20% | posterize 7 | type grayscale

Tabla 7: Algunas soluciones óptimas para las distintas imágenes

5.3.3 Refinamiento de herramientas de transformación de imágenes

En esta etapa del proceso y dado el costo computacional que conlleva para el algoritmo la utilización de cada una de las herramientas de transformación, la etapa de calibración también será utilizada para descartar aquellas funciones de transformación que en primer término podían ser consideradas como útiles, pero que desde el punto de vista práctico no lo resultan así. Para esto se considerará el conjunto más amplio de funciones de transformación identificadas como útiles que consta de 40 funciones.

5.3.3.1 Proceso de refinamiento de las funciones de transformación

Dada la cantidad de funciones de transformación que maneja el algoritmo, las cuales se muestran en la Tabla 8, es de esperar que no todas las funciones disponibles participen de las mejores soluciones. El siguiente experimento permitirá refinar el conjunto de funciones de transformación utilizadas mediante una nueva evaluación de las características de cada función y su efectividad para los objetivos del algoritmo. La evaluación tendrá en cuenta los resultados previos obtenidos, pudiendo ahora establecer cuales funciones participaron frecuentemente en las mejores soluciones, cuáles en algunas y cuáles no participaron en ninguna de las mismas.

negate	unsharp
dither	colorize
threshold	blur
posterize	rotate
normalize	equalize

level	type grayscale
fuzz	noise
contrast	linear-stretch
gamma	median
adaptive-blur	charcoal
modulate	paint
segment	emboss
gaussian-blur	despeckle
monochrome	quantize
contrast-stretch	raise
background	transparent
adaptive-resize	adaptive-sharpen
black-threshold	separate
white-threshold	edge
random-threshold	cycle

Tabla 8: Conjunto inicial de 40 funciones utilizadas por el algoritmo

Para este experimento, se observaron las soluciones de más de 400 ejecuciones independientes para distintos tipos de imágenes y se eliminaron aquellas funciones que no aparecían en ninguna de las soluciones. Se utilizaron herramientas distribuidas libremente, como por ejemplo, *grep*, *sort*, *uniq*, *awk*, y *cat* para facilitar el procesamiento de los resultados.

5.3.3.2 Conclusiones del proceso inicial de refinamiento

En esta etapa de evaluación, dada la gran cantidad de funciones utilizadas como conjunto base de funciones con las que trabaja el algoritmo, la combinación aplicada de dicho conjunto de funciones, no logró los resultados esperados para esta experiencia. En las diversas ejecuciones realizadas durante este proceso de refinamiento sobre los distintos grupos de imágenes, no se logró reconocer ninguno de los textos, peor aún, no se lograron reconocer ninguno de los caracteres. Dada esta situación, fue necesario modificar la estrategia planeada para conseguir el objetivo de incrementar el número de funciones utilizadas potenciando el alcance de imágenes reconocidas. Como punto de partida, analizando el contexto de las pruebas, la primera hipótesis apunta a que los malos resultados pueden deberse a que el algoritmo, al considerar un mayor conjunto de funciones utilizadas para la evaluación, no tuvo la suficiente cantidad de iteraciones para obtener buenas soluciones. El siguiente paso, será el de repetir las pruebas anteriores incrementando el número de iteraciones del algoritmo a 500.

5.3.3.3 Aumento de iteraciones del algoritmo

Se reiteraron las ejecuciones para los diversos grupos de imágenes incrementando el número de iteraciones y los resultados volvieron a ser negativos, sin encontrar variaciones significativas con respecto a las pruebas anteriores. Analizando esta nueva situación se formuló la hipótesis de que el espacio de soluciones es demasiado grande y se decidió repetir los experimentos pero en este caso, utilizar un conjunto menor de funciones. El criterio utilizado, para reducir el conjunto de funciones aplicadas, fue intentar perder el menor poder de alcance de transformación del algoritmo posible. Con este fin, se buscó eliminar aquellas funciones cuya transformación ya estaba expresada por otra función. Queda claro, que no existe un método formal para determinar dichas similitudes en la acción de las funciones sobre la imagen y será necesario recurrir a los conocimientos obtenidos en el proceso de experimentación y a la propia experiencia en el uso de las funciones. Luego de un breve análisis se redujo el conjunto de funciones de 40 a 30, las cuales se muestran en la Tabla 9. A continuación se repetirá nuevamente el proceso de refinamiento.

negate	unsharp
dither	colorize
threshold	blur
posterize	rotate
normalize	equalize
level	type grayscale
fuzz	noise
contrast	linear-stretch
gamma	median
adaptive-blur	charcoal
modulate	paint
segment	emboss
gaussian-blur	despeckle
monochrome	quantize
contrast-stretch	raise

Tabla 9: Conjunto de 30 funciones resultantes luego del primer refinamiento

5.3.3.4 Reducción del conjunto de funciones de transformación

Se reiteraron las ejecuciones para los diversos grupos de imágenes decrementando la cantidad de elementos del conjunto de funciones sin lograr mejores resultados. Las diversas combinaciones de funciones que son aplicadas en la transformación se tornan excesivamente numerosas para que el algoritmo pueda encontrar buenas soluciones y evolucionar favorablemente, por el contrario la búsqueda de soluciones termina pareciéndose a una búsqueda aleatoria. Por esta razón, la búsqueda del conjunto óptimo de funciones de transformación no es tan sencilla como en un principio fue planeado. El conjunto de funciones de transformación deberá irse reduciendo mediante sucesivos experimentos de calibración sobre los diversos grupos de imágenes hasta que el algoritmo tenga un nivel de reconocimiento aceptable.

5.3.3.5 Aumento de la cantidad inicial de individuos

Nuevamente se ejecutaron los experimentos para los diversos grupos de imágenes incrementando notoriamente la cantidad de individuos de la población inicial de manera de generar diversidad. A partir de este nuevo grupo de experimentos, utilizando poblaciones de hasta 500 individuos, se notó una clara diferencia con respecto a los anteriores, los resultados finales obtuvieron un alto reconocimiento, teniendo como factor negativo, que el tiempo en finalizar las ejecuciones se incrementó considerablemente.

Considerar un número mayor de funciones en el AE requiere más individuos para alcanzar soluciones relativamente buenas, implicando así tiempos de ejecución inviables en este contexto. Algunos de los experimentos realizados en este caso demoraron más de 20 horas en finalizar. Los tiempos de ejecución son muy variables entre distintas ejecuciones del algoritmo con diferentes parámetros, funciones e imágenes. Una característica que influye en las variaciones de tiempos es que no todas las funciones aplicadas a las imágenes requieren el mismo poder de procesamiento. En la infraestructura sobre la que se trabajó, aplicar una función de las más simples a una imagen pequeña como las utilizadas, requiere aproximadamente 1 segundo, algunas más complejas como *charcoal* por ejemplo, puede llegar a requerir más de 10 segundos dependiendo del valor con que se la utilice. Hay que tener en cuenta que en la ejecución del algoritmo dichas funciones se ejecutan cientos de miles de veces.

A continuación, se buscará un balance entre el tiempo de ejecución y la cantidad de funciones utilizadas por lo que se reducirá el conjunto de funciones utilizadas. Se descartaron aquellas que no tuvieron participación en las diferentes soluciones cercanas a las óptimas, de cada uno de los grupos de imágenes, teniendo como objetivo disminuir los tiempos de ejecución del programa reconocedor sin perder su poder de reconocimiento.

5.3.3.6 Supresión de funciones de uso poco frecuente

Para este experimento se desarrollaron scripts que, con los resultados de las ejecuciones como entrada, obtuvieron las soluciones de las distintas imágenes con fitness superiores para luego contar la cantidad de veces que aparecía cada función. A continuación se muestra la Tabla 10, que contiene los resultados de dichos experimentos, mostrando la cantidad de veces que apareció cada función en soluciones cercanas a las óptimas.

Función	**Cantidad de apariciones**
negate	96
unsharp	88
dither	80
colorize	73
threshold	70
blur	68

posterize	66
rotate	66
normalize	61
equalize	61
level	60
type grayscale	48
fuzz	47
noise	42
contrast	42
linear-stretch	42
gamma	37
median	37
adaptive-blur	37
charcoal	32
modulate	32
paint	28
segment	24
emboss	6
gaussian-blur	4
despeckle	4
monochrome	0
quantize	0
contrast-stretch	0
raise	0

Tabla 10: Resultados experimentales de funciones en soluciones

A partir del conjunto de soluciones obtenidas y dado el alto costo de agregar funciones irrelevantes para el desempeño del reconocimiento se descartaron aquellas funciones que tuvieron poca participación en el conjunto de soluciones, en este caso las últimas 7 de la Tabla 9, quedando el siguiente conjunto de funciones mostrados en la Tabla 11.

type grayscale	colorize
threshold	gamma
paint	linear-stretch
negate	median
normalize	posterize
noise	modulate
equalize	unsharp

contrast	segment
dither	level
blur	fuzz
charcoal	rotate
adaptive-blur	

Tabla 11: Conjunto parcial de 23 funciones de transformación

5.3.3.7 Supresión de funciones que no afectan la solución final

El proceso de refinamiento de funciones obtuvo un conjunto de 23 funciones, las cuales fueron seleccionadas por participar reiteradas veces en las soluciones finales de los diversos experimentos que se realizaron en la antes mencionada etapa de refinamiento y calibración. Esta etapa de refinamiento consistió en detectar precisamente cuales de estas funciones fueron imprescindibles para efectivamente obtener cada una de las soluciones. Dicho de otra forma, se modificó cada una de las soluciones encontradas, relevando funciones y verificando que pese a eliminar dicha función, el resultado final seguía siendo óptimo y no se veía afectado por la exclusión de dicho elemento. También se profundizó en la documentación de cada función para detectar algunas que sus resultados estaban contenidos o eran muy similares a los de otras. Luego de esta etapa el conjunto de funciones final quedo determinado por los elementos que muestra la Tabla 12.

type grayscale
threshold
paint
negate
noise
equalize
blur
charcoal
colorize
posterize
level
fuzz

Tabla 12: Conjunto definitivo de 12 funciones

Vale la pena mencionar que no se pudo hacer un proceso completamente automático para obtener las transformaciones a partir del conjunto más amplio de funciones de transformación en tiempos razonables. Esto se debe al tamaño del espacio de búsqueda de dicho problema. En la próxima subsección se estima el tamaño del espacio de búsqueda al considerar el conjunto de 12 funciones de transformación para dar una idea de lo grande que puede resultar al considerar las 40 funciones originales.

5.3.3.8 Estimación del tamaño del espacio de soluciones

Para estimar el tamaño del espacio de soluciones se considera el conjunto de funciones candidatas y los posibles valores de los parámetros de entrada. En la Tabla 13 se presentan las funciones, sus parámetros y la cantidad de valores posibles que los mismos implican para cada una. La primer y segunda columna de los parámetros expresan los valores mínimo y máximo que se utilizaran en cada función. La tercer columna de parámetros es el intervalo que se utiliza para generar valores para la función, o sea, si los parámetros de una función son [40, 70, 10], los valores posibles para esa función son 40, 50, 60 y 70. La cantidad de valores es en realidad una aproximación, porque si bien lo explicado anteriormente da un piso para la cantidad de valores que puede tomar una función, en algunos casos los operadores de mutación y cruzamiento de AE generan nuevos valores.

Función	Parámetros			Cantidad de valores
type grayscale	0	1	1	2
threshold	5	99	1	95
paint	1	4	1	4
negate	0	1	1	2
noise	0	3	1	4
equalize	0	1	1	2
blur	1	4	1	4
charcoal	1	5	1	5
colorize	20	70	10	6
posterize	2	7	1	6
level	20	100	10	9
fuzz	20	70	10	6

Tabla 13: Análisis del espacio de soluciones

En este caso, tomando en cuenta el conjunto de 12 funciones candidatas, y sin considerar las permutaciones que provoca el orden de aplicación de las funciones y el efecto de aplicar o no las funciones lo que tornaría muy dificultoso el análisis, el conjunto de soluciones contiene 472.780.800 elementos. Teniendo en cuenta los tiempos de ejecución que se consideran aceptables para el algoritmo planteado, y tomando en cuenta el tamaño del espacio de soluciones para 12 funciones candidatas, parece razonable trabajar con ese conjunto limitado de funciones.

5.3.4 Calibración formal

El proceso formal de calibración tiene un valor preponderante en toda experimentación que involucre un algoritmo evolutivo y el éxito del algoritmo evolutivo depende considerablemente de que la calibración formal de parámetros se haya realizado de una forma correcta. Este proceso tiene como objetivo encontrar la mejor combinación de valores de parámetros de manera de lograr el mejor desempeño del algoritmo.

El proceso se realizará en dos etapas para disminuir la cantidad de experimentos. En la primera etapa se harán variar los parámetros de mutación y cruzamiento, para luego, establecidos estos, se variaran los parámetros de población inicial e iteraciones quedando de esta forma, los cuatro parámetros determinados.

5.3.4.1 Probabilidad de mutación y cruzamiento

El proceso formal de calibración de los parámetros de mutación y cruzamiento se realizó tomando en cuenta los cinco grupos de imágenes identificados en los cuales se ha desarrollado el proyecto. Para cada uno de dichos grupos se realizaron 10 experimentos por cada combinación posible de parámetros. Los valores del tamaño de la población y cantidad de iteraciones se mantuvieron fijos en 50 y 30 respectivamente. A continuación se muestra un esquema con los promedios de los valores de fitness obtenidos en los distintos experimentos.

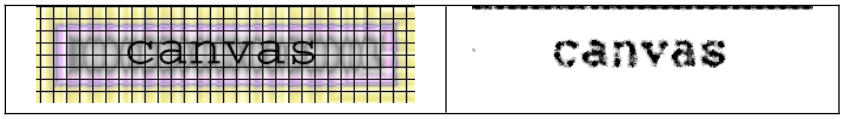


Figura 24: Imagen "canvas" representativa del conjunto Breaking a Visual CAPTCHA y una solución óptima

		Cruzamiento		
		0.7	**0.8**	**0.9**
Mutación	**0.1**	27	33,3	40,5
	0.05	30,4	40,4	40,9
	0.01	38,9	53,55	34,9

Tabla 14: Resultados según valores de probabilidad de mutación y cruzamiento para imagen "canvas"

La Figura 24 muestra la imagen seleccionada para realizar la calibración del primer conjunto antes de ejecutar el algoritmo junto con la imagen obtenida luego de ejecutar el algoritmo en una de las soluciones de fitness óptimo.

Las siguientes imágenes presentadas ilustran de manera similar tanto los resultados obtenidos como las transformaciones efectuadas en el resto de las imágenes pertenecientes a los cuatro conjuntos de imágenes restantes.

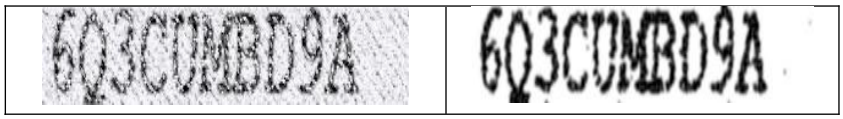


Figura 25: Imagen "6Q3CUMBD9A" representativa del conjunto BotDetect CAPTCHA Online Demo y una solución óptima

		Cruzamiento		
		0.7	**0.8**	**0.9**
Mutación	**0.1**	90	90	90
	0.05	89,5	90	90
	0.01	90	89,6	90

Tabla 15: Resultados según valores de probabilidad de mutación y cruzamiento para imagen "6Q3CUMBD9A"

Figura 26: Imagen "81f5c1" representativa del conjunto Aardvark Topsites PHP y una solución óptima

		Cruzamiento		
		0.7	**0.8**	**0.9**
Mutación	**0.1**	38,3	39,9	43,2
	0.05	44,1	37,7	42,5
	0.01	37,7	36,3	39,4

Tabla 16: Resultados según valores de probabilidad de mutación y cruzamiento para imagen "81f5c1"

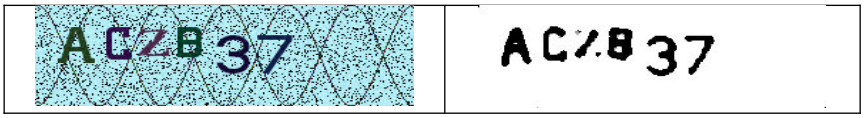


Figura 27: Imagen "ACZB37" representativa del conjunto PHP Link Directory y una solución óptima

		Cruzamiento		
		0.7	**0.8**	**0.9**
Mutación	**0.1**	52,7	54	53,7
	0.05	53	53,3	51,9
	0.01	52,6	53	53,5

Tabla 17: Resultados según valores de probabilidad de mutación y cruzamiento para imagen "ACZB37"

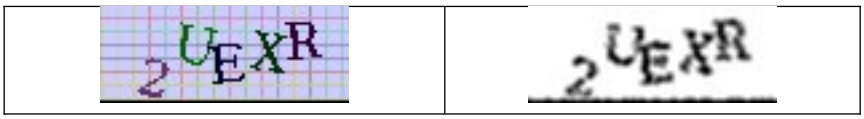


Figura 28: Imagen "2UEXR" representativa del conjunto Sitio web LoQuo y una solución óptima

		Cruzamiento		
		0.7	**0.8**	**0.9**
Mutación	**0.1**	20,6	21,9	19,5
	0.05	17,7	21,2	22,2
	0.01	23,5	22,9	21,3

Tabla 18: Resultados según valores de probabilidad de mutación y cruzamiento para imagen "2UEXR"

Observando los resultados obtenidos en las cinco diferentes calibraciones para cada uno de los cinco grupos de imágenes, como era de suponer, no existe una combinación de parámetros que se destaque de manera relevante sobre el resto de las combinaciones en todos los conjuntos. Los mejores valores de los parámetros variarán dependiendo de las características del conjunto al que pertenece la imagen. Para determinar una combinación de valores que obtiene buenos resultados en todos los conjuntos, se analiza la sumatoria de los promedios del fitness ponderando por el valor máximo de fitnesss alcanzado para cada uno de los cinco conjuntos de imágenes. A partir de dicho análisis, es posible concluir que la configuración que mejor se adapta a los distintos escenarios tiene

como valor de probabilidad de mutación 0.01 y de cruzamiento 0.8. Dichos valores serán tomados fijos en la próxima etapa de calibración que tiene como objetivo determinar los valores de los parámetros de cantidad de iteraciones y tamaño de población.

Los experimentos realizados en los distintos conjuntos de imágenes implicaron diferentes tiempos de ejecución. En la Tabla 19 se muestran los promedios de tiempos incurridos por cada ejecución simple del algoritmo, para las distintas imágenes, junto con información adicional de las imágenes.

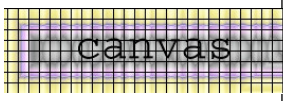
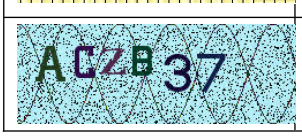
Imagen	Tamaño (KB)	Tipo	Tiempo promedio (horas)
	3.3	JPEG image data, JFIF standard 1.01	0.2
	9	JPEG image data, JFIF standard 1.01	1.5
	7	JPEG image data, JFIF standard 1.01	1.8
	34.9	JPEG image data, JFIF standard 1.01	4
	42.3	PNG image, 266 x 87, 8-bit/color RGB, non-interlaced	1.5

Tabla 19: Imágenes y tiempos de ejecución de este experimento

No es posible sacar conclusiones formales con respecto a los tiempos porque no se trabajó sobre equipos dedicados, por lo tanto, existieron elementos externos que influyeron sobre los tiempos de ejecución. Además, los algoritmos se ejecutaron con baja prioridad, por lo tanto los tiempos de procesamiento son aún más susceptibles a otros procesos que estén utilizando la CPU.

5.3.4.2 Cantidad de iteraciones y tamaño de población

El proceso formal de calibración de los parámetros de cantidad de iteraciones y tamaño de población se realizó tomando en cuenta los cinco grupos de imágenes. Para cada uno de dichos grupos se realizaron 10 experimentos por cada combinación posible de parámetros, tomando fijos los parámetros de mutación y cruzamiento calculados anteriormente. A continuación se muestra un esquema de los promedios de los valores obtenidos en los distintos experimentos.

		Población		
		10	**30**	**80**
Iteraciones	**20**	28,7	17,1	17,4
	50	27.3	37,7	44,7

	100	45,9	47,6	45

Tabla 20: Resultados con la imagen correspondiente al conjunto Breaking a Visual CAPTCHA

La Tabla 20 muestra los promedios de fitness obtenidos para cada uno de los cruces de población e iteraciones evaluados para la primera imagen. Las siguientes tablas exponen de manera similar dichos valores para el resto de las imágenes representantes de los cuatro conjuntos restantes.

		Población		
		10	**30**	**80**
Iteraciones	**20**	81,8	89,1	86,1
	50	90	90	90
	100	90	90	90

Tabla 21: Resultados con la imagen correspondiente al conjunto BotDetect CAPTCHA Online Demo

		Población		
		10	**30**	**80**
Iteraciones	**20**	33	37,7	36,6
	50	39,3	37,8	38,6
	100	41,5	43,6	45,6

Tabla 22: Resultados con la imagen correspondiente al conjunto Aardvark Topsites PHP

		Población		
		10	**30**	**80**
Iteraciones	**20**	46,5	49,4	48,1
	50	44,8	53,6	53
	100	53,5	53,8	53,7

Tabla 23: Resultados con la imagen correspondiente al conjunto PHP Link Directory

		Población		
		10	**30**	**80**
Iteraciones	**20**	16,9	15,8	17,7
	50	17,2	19,4	19,7
	100	24	31,8	28,1

Tabla 24: Resultados con la imagen correspondiente al conjunto Sitio web LoQuo

Luego de observar los resultados obtenidos, a diferencia de los resultados de mutación y cruzamiento, los valores de los cinco conjuntos parecen estar suficientemente acordes como para a simple vista poder distinguir cual fue la mejor combinación de parámetros de la experiencia. En este caso la combinación tiene un valor de población de 30 individuos y una cantidad de 100 iteraciones. Con una población de 80 individuos en vez de 30, se observan pequeñas mejores en algunos de los fitness promedio, en muchos casos no se justifica casi triplicar el tamaño de población (con el consiguiente incremento en el tiempo de ejecución) para obtener sólo pequeñas mejoras en las soluciones.

5.4 *Experimentos de validación*

Luego de establecidos los valores de los parámetros del algoritmo evolutivo fueron realizados diversos experimentos de validación de manera de medir las virtudes y falencias del algoritmo propuesto.

5.4.1 *Comparación del algoritmo evolutivo con una búsqueda aleatoria*

Una primera etapa de los experimentos de validación consistió en comparar el AE con un método de búsqueda aleatoria para una instancia representativa del problema. Se trabajó sobre la imagen "81f5c1" que se presenta en la Figura 29 perteneciente a uno de los conjuntos de prueba y posee notorias dificultades para el reconocimiento.

Figura 29: Imagen utilizada en este experimento

Para el algoritmo evolutivo se utilizaron valores de probabilidad de cruzamiento y mutación iguales a 0.9 y 0.1 respectivamente. La población fue de 100 individuos y la cantidad de iteraciones igual a 80. Éstos parámetros fueron obtenidos luego de analizar los mejores resultados en los experimentos de calibración teniendo en cuenta solo los resultados con la imagen del conjunto Aardvark Topsites PHP (Tablas 16 y 23). Puede observarse que los valores de probabilidad y mutación no son exactamente los mayores de la Tabla 16. En este caso, como la combinación de parámetros que tenía el mayor promedio era atípica (0.7 y 0.05) con respecto a las obtenidas con esa misma imagen y otras en diferentes experimentos, se realizaron pruebas especificas sobre esta imagen con dicha combinación y la que tenía el siguiente mayor promedio (0.9 y 0.1). Los resultados de dichas pruebas derivaron en los parámetros que se utilizaron finalmente en este experimento.

La búsqueda aleatoria se implementó utilizando el mismo esqueleto que para el algoritmo evolutivo, sólo que en este caso se realizó una sola iteración con la misma cantidad de evaluaciones que en el caso evolutivo. Es decir, que como el algoritmo evolutivo ejecuta 8080 evaluaciones (100x80+80), se generaron aleatoriamente esa cantidad de individuos.

El análisis experimental mostró que el AE fue capaz de encontrar la solución que posibilitó el reconocimiento óptimo (de todos los caracteres de la imagen) en 30 ejecuciones independientes realizadas. El método aleatorio fue incapaz de alcanzar la solución óptima pese a haberse ejecutado en 60 oportunidades. Además, el fitness promedio de las soluciones obtenidas por el algoritmo evolutivo fue de 46, siendo 54 el óptimo para esta imagen, mientras que el de la búsqueda aleatoria fue de 39. Estos resultados muestran que el AE tiene una mayor capacidad de resolución del problema de reconocimiento que una estrategia aleatoria simple.

El tiempo de ejecución del algoritmo evolutivo fue en promedio igual a 45 minutos. Existieron variaciones en los tiempos de ejecución de la búsqueda aleatoria, pero fueron insignificantes para poder compararlas teniendo en cuenta que no se está trabajando sobre servidores dedicados.

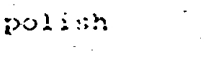
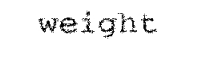
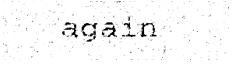
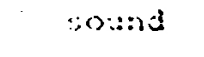
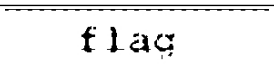
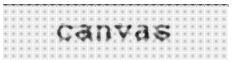
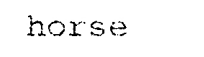
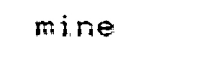
5.4.2 *Comparación del método evolutivo con otro proyecto de investigación*

El método propuesto, a manera de comparación, fue ejecutado para el reconocimiento de las imágenes de uno de los conjuntos descrito en la sección 5.1 de este capítulo perteneciente al "Gimpy: Breaking a visual captcha" (Berkeley Computer Vision Group, Simon Fraser University) [6]. Aunque el proyecto Gimpy se basó en un enfoque totalmente diferente respecto a la técnica de reconocimiento de caracteres, definió un conjunto de imágenes de prueba de dificultad variada, y publicó resultados que permiten verificar la validez del enfoque utilizando un AE.

Para este experimento se utilizaron los parámetros obtenidos en las calibraciones formales, probabilidad de cruzamiento y mutación de 0.8 y 0.01 respectivamente, tamaño de población igual a 30 y 100 iteraciones. Se realizaron 10 ejecuciones independientes del algoritmo para cada imagen.

Los tiempos de ejecución del algoritmo fueron de 34 minutos en promedio. En todos los casos se encontraron soluciones óptimas, en algunos, en los que se puede considerar que tienen dificultades menores, las 10 ejecuciones independientes evolucionaron a una solución óptima. En otros, los más complejos, solo algunas de las ejecuciones llegaron a una solución óptima.

La Tabla 26 muestra los resultados experimentales del proyecto "Breaking a visual captcha" en comparación con los obtenidos con el enfoque que utiliza el AE.

Imagen	Resultado Breaking a visual captcha	Imagen transformada	Resultado AE
	POLISH		POLISH
	WEIGHT		WEIGHT
	AGAIN		AGAIN
	SOUND		SOUND
	FLAG		FLAG
	CANVAS		CANVAS
	HORSE		HORSE
	SOCK		MARK
	JEWEL		JEWEL
	MINE		MINE

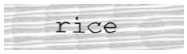
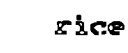
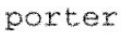
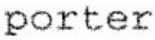
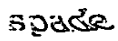
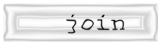
	RICE		RICE
	PORTER		PORTER
	SPACE		SPADE
	JOIN		JOIN

Tabla 26: Comparación de resultados AE con otro proyecto

El análisis de la Tabla 26 permite observar que para los casos de prueba considerados, el AE obtiene resultados comparables con los alcanzados en el proyecto Breaking a visual captcha. El AE es capaz de encontrar soluciones que reconocen correctamente todos los caracteres reconocidos en el otro proyecto, y a modo comparativo, el algoritmo de dicho proyecto falla el reconocimiento completo en dos ocasiones mientras que el algoritmo evolutivo logra un 100% de aciertos en el reconocimiento de todas las imágenes del conjunto.

5.4.3 Aprendizaje y reconocimiento

El análisis de aprendizaje y reconocimiento consiste en reconocer imágenes que compartan características de distorsión similares, entrenando el AE con otra imagen que comparta los rasgos de distorsión. El entrenamiento se realizó con la imagen "E8WY34" perteneciente al conjunto BotDetect CAPTCHA Online Demo, presentada en la Figura 30.

Figura 30: Imagen utilizada para el aprendizaje de este experimento

Los valores de probabilidad de cruzamiento y mutación utilizados fueron de 0.8 y 0.01 respectivamente. Se comenzaron utilizando los valores para cantidad de individuos e iteraciones obtenidos en la calibración formal (30 y 100 respectivamente) pero de esa forma los resultados obtenidos no eran buenos. Luego de realizar pruebas específicas con la imagen de la Figura 30 validando las soluciones encontradas con otras imágenes del mismo tipo, se observó que la combinación de 80 individuos y 150 iteraciones presentaba mejores resultados y los tiempos de ejecución se mantenían razonables. El tiempo de ejecución del AE con dichos parámetros fue de 30 minutos.

normalize | equalize | noise: 4 | dither | blur: 0 | type grayscale

Figura 18: Solución para el problema de aprendizaje y reconocimiento.

La solución encontrada presentada en la Figura 18, se utilizó para reconocer 20 imágenes con patrones de interferencia y distorsión similares a la imagen "E8WY34", para las cuales el motor OCR es incapaz de reconocer ningún carácter cuando trabaja con la imagen sin convertir. Los resultados obtenidos luego de aplicar la conversión determinada por la solución hallada por el AE

se presentan en la Tabla 27.

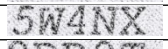
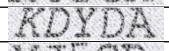
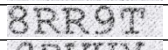
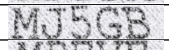
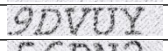
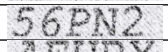
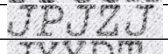
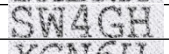
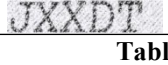
Imagen a reconocer	Texto reconocido	Imagen a reconocer	Texto reconocido
	2C9WN		K4YGA
	5W4NX		KDYDA
	8RR9T		MJ 5GB
	9DVUY		MPEVE
	56PN2		NZJXH
	AEUBX		PTHZE
	DBSNB		RTF62
	FFCSA		SMBNA
	JPJZJ		SW4GH
	JXXDT		YGN6U

Tabla 27: Resultados del reconocimiento aplicando el AE.

El análisis de la Tabla 27 muestra que de un total de 100 letras, 99 fueron reconocidas exitosamente luego de aplicada la transformación hallada por el AE. El único error consistió en reconocer un '5' como una 'S' y otro detalle fue que en la imagen con los caracteres "JXXDT" se reconoció un espacio en blanco al final. El error de reconocimiento puede estar inducido porque la imagen de entrenamiento no posee ningún '5'.

Puede observarse que las distorsiones e interferencias en las imágenes del experimento anterior son homogéneas y muy similares entre distintas imágenes. A continuación se realiza el mismo experimento pero con imágenes del conjunto PHP Link Directory, las cuales poseen diversas distorsiones e interferencias con dificultades mayores que las del conjunto BotDetect CAPTCHA Online Demo.

Debido a que las imágenes del conjunto PHP Link Directory poseen un nivel de dificultad muy alto, para la búsqueda de la solución con el AE se creó una imagen que incluye varias de las pertenecientes al conjunto en cuestión y se presenta en la Figura 31. De esta forma se busca encontrar una solución lo mejor posible para imágenes con distintas características.

Figura 31: Imagen creada uniendo varias del conjunto PHP Link Directory

Se utilizaron los mismos valores para las probabilidades de cruzamiento y mutación del AE que en el experimento anterior pero en este caso, debido a la mayor dificultad de la imagen, se utilizó una población de 300 individuos y 200 iteraciones. La solución encontrada se muestra a continuación en la Figura 19. Como era predecible, los tiempos de ejecución fueron mucho mayores, en este caso requirió 18 horas en promedio.

type grayscale | colorize 49% | charcoal 3 | level 82% | blur 1 | threshold 31% | equalize

Figura 18: Solución para el problema de aprendizaje y reconocimiento.

En este experimento, el nivel de reconocimiento fue mucho menor que en el anterior. Las notorias diferencias entra las imágenes de este conjunto no permitieron llegar a los niveles de exactitud en el

reconocimiento obtenidos con el conjunto BotDetect CAPTCHA Online Demo. De todos modos, teniendo en cuenta que el AE se está ejecutando con parámetros y funciones de transformación calibradas de forma genérica, los resultados obtenidos no se consideran totalmente deficientes. Es razonable suponer que al calibrar el AE y principalmente las funciones de transformación utilizadas para este conjunto de imágenes en particular, se pueden lograr mejores resultados en tiempos mucho menores. A continuación se presenta la Tabla 28 con las imágenes utilizadas para la validación de la solución encontrada y los caracteres reconocidos en cada una de ellas.

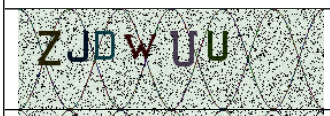
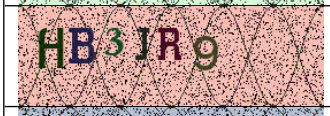
Imagen	Caracteres reconocidos
	ZJD wUU .
	TCBSWA `
	Ur6U\s}RC
	1A.cvAu ,
	
	HB3I}é{gr ,7 4
	uGFKAz

Tabla 28: Caracteres reconocidos en imágenes del conjunto PHP Link Directory

Al observar los resultados presentados en la Tabla 28 pueden identificarse más errores que en el experimento anterior, incluso en una de las imágenes no se reconoció ningún carácter. En un total de 42 caracteres, 26 fueron reconocidos correctamente.

Una observación muy importante que puede ser fundamental para comprender los resultados de este experimento, es que el tipo de imagen, en este caso no referido a características visuales sino a su formato, niveles de compresión, parámetros de los algoritmos utilizados para la compresión, etc., es determinante a la hora de utilizar soluciones del AE obtenidos con una imagen para reconocer caracteres en otras. Esto puede clarificarse con un ejemplo, si para determinada imagen se obtiene una solución óptima con el AE y luego esa misma imagen se procesa para almacenarla en otro formato o el mismo formato que la original pero utilizando una compresión mayor para obtener un tamaño menor, con seguridad la solución encontrada va a ser muy deficiente en la nueva imagen. Esto se debe a que las funciones de transformación sobre las imágenes son muy susceptibles a cambios en la imagen aunque éstos no sean visibles a simple vista. A partir de los resultados obtenidos en este experimento se infiere que el procesamiento efectuado para crear una imagen

mayor conformada por varios captchas, modificó las imágenes originales y esas pequeñas modificaciones imperceptibles a simple vista, tuvieron repercusiones mayores a la hora de los resultados obtenidos de aplicar las funciones de transformación.

Con el fin de profundizar mínimamente en esta última hipótesis, se experimentó con varias de las soluciones obtenidas por el AE para la imagen del conjunto PHP Link Directory en la etapa de calibración. Se observó un reconocimiento mayor en la mayoría de los casos. Esto confirma la hipótesis enunciada anteriormente sobre los errores generados por el procesamiento de la imagen con varios captchas. En la Tabla 29 se presentan los caracteres reconocidos con las mismas imágenes con una solución obtenida en la etapa de calibración, ésta se muestra en la Figura 32.

negate | charcoal 3 | blur 4 | threshold 31% | posterize 5 | type grayscale | equalize

Figura 32: Solución obtenida en la etapa de calibración para una imagen del conjunto PHP Link Directory

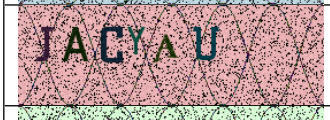
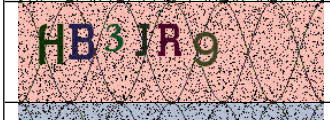
Imagen	Caracteres reconocidos
	ZJDwUU
	TCBSWA 1
	U 6DwR<·
	JACYA U
	MKARQG
	HBHR9
	DEFKAE

Tabla 29: Caracteres reconocidos en imágenes del conjunto PHP Link Directory al utilizar otra transformación

Se observan resultados notablemente mejores, en el mismo total de 42 caracteres, 36 caracteres fueron reconocidos correctamente. Además, 3 de las 7 imágenes fueron reconocidas perfectamente mientras que en el experimento anterior en ningún caso se logró esto. Otra mejora notable es que se reconocieron mucho menos caracteres extraños y menos caracteres extra erróneos.

Es importante es que notar que la solución obtenida en el último caso, requirió solamente 16 minutos de ejecución y posibilitó un reconocimiento notablemente mejor.

5.5 *Análisis de eficiencia computacional*

Tomando en cuenta el considerable tiempo necesario para evaluar la función de fitness (del orden de varios segundos), que impacta negativamente en el tiempo de ejecución del AE, se llevaron a cabo experimentos preliminares para intentar mejorar los tiempos de reconocimiento utilizando una versión paralela del AE.

Se utilizó un modelo de AE paralelo de subpoblaciones distribuidas [1] que divide la población en 10 subpoblaciones (islas), y utiliza un operador adicional de migración que intercambia tres individuos seleccionados por torneo según una topología de anillo unidireccional. Cuando una isla recibe migrantes, éstos reemplazan a los peores individuos de la población.

Los experimentos se orientaron a comparar los resultados y el desempeño computacional de las versiones secuencial y paralela del AE. El análisis permitió comprobar que el AE paralelo fue capaz de alcanzar resultados comparables a los de la versión secuencial, reduciendo en 1/3 el tiempo de ejecución total cuando se utilizan 10 procesadores. El AE paralelo se mostró como una herramienta eficaz para reducir los tiempos de cómputo sin impactar en la calidad de soluciones, aunque el bajo valor de eficiencia computacional (0,3) sugiere que aún resta trabajo por hacer para diseñar una versión eficiente, capaz de aprovechar de mejor manera el poder computacional de largos clusters de computadores.

5.6 *Observaciones del proceso experimental*

En esta sección se detallaran una serie de interesantes observaciones que surgieron a partir de los diversos experimentos realizados. Estas observaciones fueron realizadas tanto en las etapas de calibración como en las etapas de evaluación de las herramientas de reconocimiento.

5.6.1 *Imágenes con diferentes soluciones de transformación*

En los diversos experimentos realizados en los casos de estudio se pudo observar que para una misma imagen, a partir de diferentes soluciones de transformación, la herramienta OCR logró igualmente el reconocimiento de todos los caracteres para los diferentes casos. Puntualmente, existieron diferentes conjuntos de transformaciones que aplicadas a una misma imagen facilitaron el completo reconocimiento del texto de la misma. En las tablas 30 a 34 se muestran 5 diferentes transformaciones para imágenes de los diferentes conjuntos donde se puede observar que a partir de diferentes funciones de transformación el texto reconocido fue el mismo en todos los casos.

Imagen transformada	Solución hallada	Texto reconocido

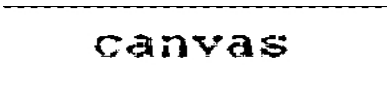
	Threshold 48 | level 48 | blur 1 | posterize 2 | fuzz 66 | negate | colorize 21 | paint 2	canvas
	Level 43 | type_grayscale | charcoal 3 | negate | noise 0 | equalize | threshold 8 | posterize 6 | colorize 70 | blur 1	canvas
	type_grayscale | threshold 50 | paint 2 | posterize 3 | fuzz 60	canvas
	Negate | colorize 20 | noise 2 | threshold 42 | type_grayscale | posterize 2 | blur 3	canvas
	Threshold 56 | paint 2 | colorize 34 | negateblur 4	canvas

Tabla 30: Soluciones óptimas para la imagen del conjunto Breaking a Visual CAPTCHA

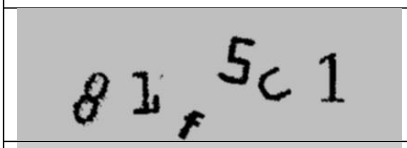
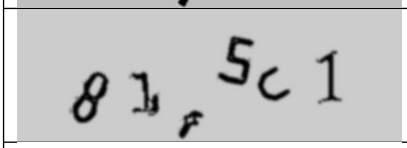
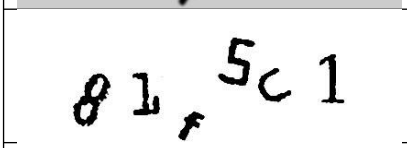
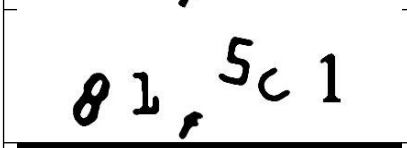
Imagen transformada	Solución hallada	Texto reconocido
	paint: 1 | threshold: 60% | level: 38% | equalize | posterize: 7 | colorize: 25% | type grayscale | fuzz: 23% | blur: 1	81f5c1
	level: 20% | normalize | negate | noise: 2 | threshold: 45% | charcoal: 3 | blur: 2 | fuzz: 43% | colorize: 20%	81f5c1
	type grayscale | paint 1 | negate | threshold 37% | blur 2 | level 59% | fuzz 28%	81f5c1
	negate | normalize | threshold: 35% | blur: 3 | noise: 3 | level: 64%	81f5c1
	level: 45% | colorize: 24% | equalize | negate | noise: 1 | type grayscale | blur: 1 | threshold: 45% | posterize: 3 | fuzz: 70%	81f5c1

Tabla 31: Soluciones óptimas para la imagen del conjunto Aardvark Topsites PHP

Imagen transformada	Solución hallada	Texto reconocido

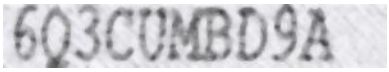
	Blur 1 | noise 0 |fuzz 48	6Q3CUMBD9A
	Blur 2 | equalize | level 100 | colorize 38 | threshold 41	6Q3CUMBD9A
	Fuzz 20 | blur 4 | negate | noise 1 | equalize | colorize 20 | level 46 | threshold 58 | posterize 3	6Q3CUMBD9A
	Negate | noise 1 | threshold 27 | equalize | charcoal 5 | colorize 30 | blur 1 | posterize 6	6Q3CUMBD9A
	Noise 0 | blur 4 | level 80 | equalize | colorize 30 | type_grayscale | fuzz 40 | negate	6Q3CUMBD9A

Tabla 32: Soluciones óptimas para la imagen del conjunto BotDetect CAPTCHA Online Demo

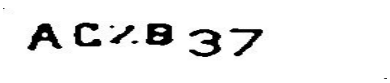
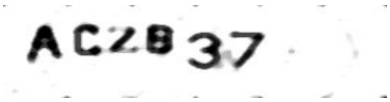
Imagen transformada	Solución hallada	Texto reconocido
	Colorize 20 | noise 0 | blur 1 | type_grayscale | threshold 33 | paint 1	ACZB37
	Colorize 48 | charcoal 5 | noise 0 | threshold 66	ACZB37
	Posterize 4 | negate | threshold 28 | colorize 40 | blur 2 | fuzz 20 | level 50	ACZB37
	Charcoal 3 | negate | noise 3 | level 47 | colorize 66 | type_grayscale | paint 1	ACZB37
	Negate | threshold 36 | paint 1 | blur 4 | noise 3 | fuzz 20 | level 92 | type_grayscale	ACZB37

Tabla 33: Soluciones óptimas para la imagen del conjunto PHP Link Directory

Imagen transformada	Solución hallada	Texto reconocido

	Fuzz 70 | blur 1 | threshold 58 | equalize | posterize 2 | type_grayscale | negate	2UEXR
	Fuzz 50 | negate | colorize 52 | level 35 | type_grayscale | equalize | charcoal 1	2UEXR
	Threshold 31 | equalize | negate | type_grayscale | posterize 6 | fuzz 40 | blur 2	2UEXR
	Threshold 46 | fuzz 29 |blur 4 | colorize 26 | negate	2UEXR
	Threshold 42 | level 100 | charcoal 3 | blur 4 | colorize 66 | type_grayscale | fuzz 23	2UEXR

Tabla 34: Soluciones óptimas para la imagen del conjunto Sitio web LoQuo

5.6.2 Impacto del orden de las funciones de transformación

A medida que se avanzó en la selección de funciones de transformación, se fueron develando interesantes características del proceso de transformación de imágenes. Una de ellas tuvo que ver con el orden de aplicación de las funciones de transformación sobre las imágenes. El éxito o el fracaso del reconocimiento de una imagen pueden estar dados simplemente por el orden en que fueron aplicadas las funciones de transformación. Más en concreto, una imagen que fue sujeta a una serie de funciones de transformación puede ser transformada en una imagen totalmente legible, mientras que la misma imagen sujeta a las mismas funciones de transformación pero en orden diferente puede quedar completamente ilegible. La Tabla 35 ejemplifica el caso en que una imagen a partir de aplicarles las mismas funciones de transformación pero en diferente orden, el texto reconocido varía notoriamente.

Imagen original	Función de transformación	Imagen transformada	Texto reconocido
	Type grayscale | blur 2 |negate | posterize 2 | charcoal l5 | level 20 | noise 2		6Q3CUMBD9A
	charcoal l5 | blur 2 | noise 2 | posterize 2 | level 20 |Type grayscale | negate		

Tabla 35: Ejemplo de cambio de orden en parámetros de solución

5.6.3 Características intuitivas de las funciones de transformación

Estudiando las soluciones obtenidas, se encontraron resultados que fueron en contra de la intuición previa que se podía tener sobre el problema. Un claro ejemplo es el de la función *negate*, en principio, la herramienta Tesseract reconoce de igual manera palabras de color negro sobre fondo blanco que palabras de color blanco sobre fondo negro. Tomando como base esa hipótesis, y dado las características ya descriptas de la función *negate*, dicha función resultaría innecesaria para favorecer el reconocimiento. En la práctica, y yendo contra esta hipótesis, existieron una serie de casos en que sin la utilización de dicha función, las imágenes no hubieran sido reconocidas. Esto último puede ser explicado, dado que cada función influye sobre las demás transformaciones, logrando resultados no tan predecibles de manera intuitiva. En las Tablas 36, 37 y 38 se muestran distintos ejemplos de los resultados obtenidos al eliminar la transformación *negate* de soluciones óptimas para la imagen "81f5c1".

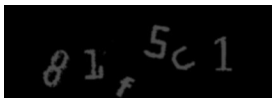
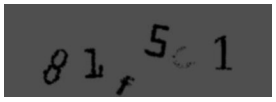
Imagen original	Función de transformación	Imagen transformada	Texto reconocido
	type grayscale | paint: 1 | negate | threshold: 38% | equalize | fuzz: 43% | colorize: 69% | blur: 4		81f5c1
	type grayscale | paint: 1| threshold: 38% | equalize | fuzz: 43% | colorize: 69% | blur: 4		8LfS ·_·__1

Tabla 36: Ejemplo 1 de solución con y sin función negate

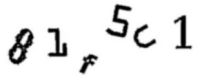
Imagen original	Función de transformación	Imagen transformada	Texto reconocido
	threshold: 65% | equalize | type grayscale | fuzz: 45% | level: 28% | paint: 1 | blur: 1 | posterize: 4 | negate | charcoal: 3		81f5c1
	threshold: 65% | equalize | type grayscale | fuzz: 45% | level: 28% | paint: 1 | blur: 1 | posterize: 4 | charcoal: 3		@m°z;>§@H

Tabla 37: Ejemplo 2 de solución con y sin función negate

Imagen original	Función de transformación	Imagen transformada	Texto reconocido

	paint: 1 | threshold: 60% | negate | normalize | blur: 2 | type grayscale | level: 68% | equalize		81f5c1
	paint: 1 | threshold: 60% | normalize | blur: 2 | type grayscale | level: 68% | equalize		6)Lf5(,1

Tabla 38: Ejemplo 3 de solución con y sin función negate

Se observan distintos escenarios en cada tabla. En la primera, se ve a simple vista que la supresión de la transformación *negate* generó diferencias en los contrastes entre las letras y el fondo, principalmente se observa en la letra "c" de la imagen. En la segunda tabla es evidente que el resultado de la transformación sin *negate* es muy diferente al original, en vez de obtenerse letras sólidas de un color distinto al fondo de la imagen, se obtienen letras con los contornos delineados pero internamente del mismo color del fondo. Los resultados presentados en la última tabla son distintos a los anteriores, intuitivamente se tiende a suponer que en ambas imágenes se deberían reconocer los mismos caracteres pero no es así; la transformación original le permite al OCR reconocer todos los caracteres y la segunda tiene varios errores.

Capitulo 6 Conclusiones y trabajo futuro

6.1 Conclusiones finales

Teniendo en cuenta los objetivos planteados al inicio de este trabajo, podemos afirmar que las metas propuestas se lograron exitosamente.

Este trabajo ha presentado un enfoque novedoso para el problema de mejorar la efectividad en el reconocimiento de caracteres de motores de OCR. La propuesta consiste en utilizar un AE para determinar una secuencia de conversiones a realizar sobre una imagen de baja calidad o con interferencias, para mejorar su reconocimiento. El AE propuesto permitió encontrar soluciones capaces de reconocer caracteres en imágenes con ruido y distorsiones considerables, superando ampliamente los niveles de exactitud alcanzados por la utilización directa del motor OCR y por la utilización de una búsqueda aleatoria para determinar las conversiones. Asimismo, el análisis experimental permitió comprobar que las soluciones obtenidas presentan buenas propiedades de generalización, permitiendo el reconocimiento de caracteres en imágenes similares a la utilizada para entrenar al algoritmo. Las experiencias demostraron que el algoritmo es útil para romper o evaluar la efectividad de los captchas. Si bien ese no fue su objetivo, se observó que en muchos sistemas de captcha utilizados en internet actualmente, el algoritmo encontró soluciones que permitían reconocer correctamente los caracteres de las imágenes en porcentajes muy altos; demostrando así que esos captchas son poco efectivos.

Las consideraciones referentes a las conclusiones, surgen tanto del proceso de investigación del conjunto de herramientas y tecnologías utilizadas para llevar a cabo el trabajo, así como de los resultados obtenidos luego de poner en marcha la herramienta desarrollada. En primer lugar, una parte significativa del trabajo, tuvo lugar en el relevamiento de herramientas de reconocimiento de caracteres. El conjunto de herramientas relevadas mostró resultados muy diversos en su eficiencia para reconocer texto mecanografiado. Dicha evaluación evidenció el notorio avance de las herramientas de reconocimiento de utilización libre, ya que para el caso de la herramienta que fue seleccionada en el relevamiento para su uso en el proyecto, en ciertos conjuntos de imágenes el porcentaje de acierto fue superior al 80% de reconocimiento completo del texto de la imagen. Otro resultado sumamente interesante y que evidencia la dificultad de los objetivos planteados, quedó expuesto al intentar reconocer imágenes con niveles altos de interferencia o ruido, en estos casos, ninguna de las herramientas relevadas, logró reconocer por completo el texto de alguna de las imágenes, independientemente de los diferentes tipos de perturbación y la complejidad de los mismos. Dados estos resultados se volvió crucial el proceso de transformación de la imagen previa a la aplicación de la herramienta OCR a modo de reducir las perturbaciones que posee la imagen.

El proceso de selección de herramientas de transformación de imágenes fue una de las tareas más trabajosas y complejas del proceso de investigación, por dos razones principales: existía un enorme conjunto de funciones de transformación de imágenes para relevar que superaban las 200 funciones y por otra parte, no existían criterios del todo claros para establecer el desempeño de una función para transformar una imagen de modo de mejorar su posterior reconocimiento. Los resultados de esta etapa evidenciaron que existía una gran cantidad de funciones del conjunto relevado, que de acuerdo al criterio de evaluación adoptado no resultaban provechosas y fueron descartadas. El conjunto resultante de esta etapa abarcaba un conjunto de 40 funciones. En principio hubiera sido beneficioso que el propio AE realizara naturalmente la selección de funciones utilizadas, pero un

simple estudio del espacio de soluciones para las 40 funciones, determino que por su enorme dimensión, era necesario reducir dicho conjunto de funciones previo a la aplicación del algoritmo. La reducción debió utilizar una serie de métodos experimentales que abarcaron la eliminación de funciones de uso poco frecuente y prescindible para un conjunto de soluciones estudiadas. Posteriormente a esta etapa y a través de pruebas primarias de reconocimiento, utilizando el conjunto de herramientas de transformación de imágenes relevadas, se pudo comprobar que el criterio de selección de las funciones fue acertado dado que se obtuvieron resultados promisorios. Asimismo se corroboró que el comportamiento del proceso de reconocimiento fue correcto para los diferentes conjuntos de imágenes independientemente de las características del conjunto, lo que podría significar ampliar el conjunto de casos de prueba con nuevos grupos de imágenes con buenas expectativas de que el proceso de reconocimiento mantenga un buen desempeño.

La siguiente etapa que involucró el proceso formal de calibración de parámetros, también permitió obtener conclusiones referentes a la posible generalización del algoritmo a cualquier conjunto de imágenes. En este caso, se observaron diferencias en el proceso de calibración para cada uno de los conjuntos pertenecientes a los casos de prueba, no siendo tan fácil determinar una única calibración con buen desempeño para todos los conjuntos. A partir de dichos resultados, parece de importancia realizar una nueva calibración de parámetros en el caso de utilizar nuevos conjuntos de imágenes, sobre todo si las características del nuevo conjunto son muy diferentes a los ya estudiados.

Posteriormente a las pruebas de calibración ya se pudieron obtener los primeros resultados propios del desempeño del algoritmo de reconocimiento en diversos contextos. El primer resultado interesante surge al comparar los resultados del algoritmo con los de un proyecto académico de reconocimiento de captchas (Breaking a visual captcha). En este caso los resultados superaron a los del proyecto “Breaking a visual captcha” ya que el AE reconoció el 100% de los textos de las imágenes mientras que el otro proyecto no fue capaz de reconocer el texto de dos de las imágenes de su conjunto de prueba. Siguiendo con la evaluación de desempeño del algoritmo, resulta importante medir su capacidad con respecto al desempeño de un algoritmo que obtenga resultados de forma aleatoria. En este caso el desempeño del algoritmo fue ampliamente superior, logrando no sólo encontrar soluciones mucho más rápidamente, sino que en algunos casos logró encontrar soluciones que la forma aleatoria nunca logró determinar. Por último un resultado interesante tuvo que ver con los resultados obtenidos en la etapa del reconocimiento de un conjunto de imágenes de características de distorsión similares, a partir de calibrar el algoritmo con una muestra de dicho conjunto. En esta evaluación se lograron resultados muy auspicios cuando las características de distorsión del conjunto de imágenes eran similares unas con otras.

Es importante notar que si bien el tiempo de ejecución en algunos casos puede parecer un inconveniente, no lo es. El proceso de ejecución del AE se hace solamente una vez y la solución que se obtiene (las funciones a aplicar, sus parámetros y orden) se aplica todas las veces que sea necesario con bajo tiempo de ejecución. Por ejemplo, si se desean optimizar los niveles de reconocimiento del texto contenido en un libro con cierto deterioro, alcanza con entrenar el AE con un par de fragmentos de páginas representativos del “deterioro” y luego se aplica la solución obtenida a todo el libro.

6.2 Dificultades encontradas durante el desarrollo del proyecto

Una de las principales dificultades encontradas en el transcurso del trabajo, tuvo que ver con la falta de antecedentes respecto al uso de algoritmos evolutivos para trabajar con imágenes. A partir de esta situación, la estrategia innovadora empleada en este trabajo elevaba el valor del proyecto, pero

por otro lado dificultaba los sucesivos pasos en el desarrollo de la solución ya que se debió recurrir mayormente a la continua experimentación para avanzar en la tarea.

La falta de información también se dio lugar en el estudio de los motores OCR. En este caso, se encontró muy poca información de las características de los mismos, y debieron ser sujetos a diversos experimentos para determinar cuáles resultaban adecuados para las características del trabajo.

6.3 Trabajo futuro

Las líneas de trabajo actual y futuro se orientan a mejorar el procedimiento algorítmico, a profundizar en el diseño de versiones paralelas del AE y a la mayor experimentación con otro tipo de conjuntos de imágenes.

El aspecto relacionado con la mejora algorítmica se orienta a perfeccionar el funcionamiento del AE propuesto en procura de mejorar los tiempos de ejecución del algoritmo. Para ello es necesario identificar los cuellos de botella en la ejecución del algoritmo de modo de mejorar los tiempos de ejecución. En particular, el problema parece situarse en el costo de evaluación que conllevan las funciones de transformación sobre la imagen en cada iteración del algoritmo. Una posible solución es incorporar un diccionario que permita evitar evaluar una misma solución en forma repetida. Otra alternativa puede ser diseñar estrategias que permitan descartar soluciones sin necesidad de aplicarlas y validar su resultado. Respecto al segundo punto, se propone profundizar en el estudio de modelos de AE paralelos y analizar alternativas para mejorar la eficiencia computacional de los AE propuestos. Asimismo, se debe evaluar una posible reestructura en la implementación para mejorar los tiempos de ejecución.

Dado que este trabajo se basó puntualmente en el estudio de imágenes de tipo CAPTCHA sería muy interesante el estudio de otros tipos de imágenes, en particular, al estudio de imágenes de textos deteriorados escaneados, como pueden ser periódicos o libros los cuales poseen hojas deterioradas por el paso del tiempo.

Por último, un trabajo interesante es publicar un sitio web que permita interactuar con el algoritmo, o sea, que usuarios de internet puedan subir imágenes para luego con el AE buscar soluciones y presentar los resultados. Incluso se podría permitir que los usuarios posean cierto control sobre algunos de los parámetros del algoritmo o las funciones de transformación a utilizar. Registrando el uso del algoritmo, entre otras cosas, se podrían aprender combinaciones de parámetros y funciones con buenos resultados en base a los experimentos que realizan los usuarios. Este trabajo ya fue comenzado, lográndose implementar un sistema distribuido entre todas las PCs de las salas UNIX de la Facultad de Ingeniería, para que desde una página web se pueda subir una imagen y automáticamente se ejecute el AE (por ahora con parámetros fijos) con ella en una PC que no tenga procesos utilizando mucho CPU. Luego desde la misma página web se puede ver la solución encontrada. Si bien partes fundamentales del trabajo ya están implementadas, le faltan muchos detalles como para considerarlo pronto para publicar en internet.

Bibliografía

1. E. Alba, M. Tomassini: *Parallelism and evolutionary algorithms*. IEEE Transactions on Evolutionary Computation, 6(5):443-462, 2002.
2. T. Bäck., D. Fogel y Z. Michalewicz: *Handbook of Evolutionary Computation*, Oxford Univ. Press, 1997.
3. C. Darwin: *The Origin Of Species,* 1859.
4. L. Davis: *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
5. A. E. Eiben. y J. E. Smith: *Introduction to Evolutionary Computing* (Natural Computing Series), 2003.
6. G. Mory y J. Malik: *Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA*, Computer Science Division, University of California, 2003
7. S. V. Rice, G. Nagy y T. Nartker: *An Illustrated Guide to the Frontier*, *Optical Character Recognition.* New York, 1999.
8. D. Rivero, R. Vidal, J. Dorado, J. R. Rabuñal, A. Pazos: *Restoration of old documents with genetic algorithms,* 2003.
9. H. Sakano, H. Kida, N. Mukawa: *Seeing the character images that an OCR system sees - analysis by genetic algorithm,* 1996.
10. H. F. Schantz, *The History of OCR, Optical Character Recognition.* Manchester Center, VT: Recognition Technologies Users Association, 1982.
11. S. N. Sivanandam: *Introduction to Genetic Algorithms*, 1990.
12. R. Smith: An *Overview of the Tesseract OCR Engine*, Proc. 9th International Conference on Document Analysis and Recognition, pp.629-633, 2007
13. R. M. Stair, y G W. Reynolds: *Principles of Information Systems*, A Managerial Approach, 5th ed. Boston: Course Technology—ITP, 2001.
14. C. Y. Suen: *Future Challenges in Handwriting and Computer Applications*, 3rd International Symposium on Handwriting and Computer Applications, 1987.
15. C. C. Tappert: *The State of the Art in On-line Handwriting Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence 1990.
16. Web site de LoQuo disponible en http://www.loquo.com/ consultado en Diciembre de 2009.
17. Web site de Aardvark Topsites PHP disponible en http://www.aardvarktopsitesphp.com/ consultado en Diciembre de 2009.
18. Web Site de desarrollo de GOCR disponible en http://jocr.sourceforge.net consultado en Diciembre de 2010.
19. Web Site de GNU disponible en http://www.gnu.org/software/ocrad/ consultado en Diciembre de 2010.
20. Web Site de HOCRLib disponible en http://hocr.berlios.de/ consultado en Diciembre de 2010.
21. Web Site de ImageMagick library disponible en http://www.imagemagick.org/ consultado en Diciembre de 2010.
22. Web site de LANAP Software disponible en http://captcha.biz/demo/captcha-demo.aspx consultado en Diciembre de 2009.
23. Web site de PHP Link Directory disponible en http://www.phplinkdirectory.com/ consultado en Diciembre de 2009.
24. Web Site de Reconocimiento óptico de caracteres disponible en

http://www.intellogist.com/wiki/Optical_Character_Recognition consultado en Diciembre de 2010.

25. Web Site de Tesseract-OCR disponible en http://code.google.com/p/tesseract-ocr/ consultado en Diciembre de 2010

Apéndice I: Relevamiento general de funciones

Se realizo una etapa de evaluación de todas las funciones de la librería ImageMagick. En la tabla se presenta un resumen del procesamiento que realiza cada función, sus parámetros, los resultados observados y una calificación relativa a la utilidad que muestra cada una para este proyecto. Dicha calificación se midió empíricamente, observando los resultados de aplicar cada función y teniendo en cuenta si las transformaciones observadas podrían llegar a ser útiles para remover interferencias o simplificar el trabajo del OCR.

Nombre	Descripción	Parámetros	Resultados	Calificación
Adjoin	Función que une imágenes en un archivo de multi-imágenes	No tiene.	A partir de la experimentación dicha función no parece tener aplicaciones favorables a las establecidas.	Función no favorable.
Authenticate [value]	Función que descifra a la imagen con una contraseña.	Contraseña para descifrar la imagen.	A partir de la experimentación dicha función no tiene aplicaciones favorables a las establecidas.	Función no favorable.
Background [color]	Función que establece un color de fondo para la imagen.	Color a establecer como fondo para la imagen.	A partir de la experimentación dicha función puede tener aplicaciones favorables a las establecidas.	Función que puede ser favorable
Bias [value]	Función que deforma la imagen.	Valor utilizado para torcer la imagen	A partir de la experimentación dicha función no tiene aplicaciones favorables a las establecidas.	Función no favorable.
Black point compensation	Función modifica las características de colores de la imagen	No tiene.	A partir de la experimentación dicha función actuó dejando de manera más difusa las imágenes por lo que no parece favorable a las establecidas.	Función no favorable.
Blue primary	Función que	Valor que pondera	A partir de la	Función no

[point]	modifica las características de colores de la imagen.	las alteraciones en los colores de la imagen.	experimentación dicha función actuó dejando de manera más difusa las imágenes por lo que no parece favorable a las establecidas.	favorable.
Bordercolor [color]	Función que modifica las características de los bordes de la imagen.	Color con que cubre los bordes de la imagen.	A partir de la experimentación con dicha función, se pudo observar que la función puede actuar a modo de completar el trazo de las líneas de aquellos caracteres de líneas difusos. De este modo, esta función, puede realizar transformaciones favorables a las establecidas.	Función que puede resultar favorable.
Caption [string]	Función que asigna un título a la imagen.	Texto a asignar como título a la imagen.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Channel [type]	Función que modifica los tipos de canales de la imagen.	Tipo de canal a modificar.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Colors [value]	Función que define la cantidad máxima de colores en la imagen.	Numero de colores.	A partir de la experimentación no se lograron cambios significativos de transformación	Función no favorable.

			sobre la imagen por lo que no parece favorablemente a lo establecido.	
Colorspace [type]	Función que modifica el espacio de colores de la imagen.	Espacio de colores.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Comment [string]	Función que agrega un comentario a la imagen.	Comentario para agregar.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Depth [value]	Función que varía la profundidad de la imagen.	Profundidad de la imagen.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Dither	Función que aplica difusión a la imagen.	No tiene.	A partir de la experimentación dicha función, se pudo observar que La función puede actuar a modo de completar el trazo de las líneas de aquellos caracteres de líneas difusos. De este modo, esta función, puede realizar transformaciones favorables a las	Función que puede resultar favorable.

			establecidas.	
Fill [color]	Función que pinta el gráfico primitivo de la imagen.	Color utilizado.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Fuzz [distance]	Función que considera iguales colores que se encuentran a esa distancia en la imagen.	Distancia entre los colores.	A partir de la experimentación se noto que pueden ser eliminados ruidos actuado favorablemente a lo establecido.	Función que puede resultar favorable.
Label [string]	Función que asigna una etiqueta a la imagen.	Texto de la etiqueta.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Mattecolor [color]	Función que establece el color del marco de la imagen.	Color del marco.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Red Primary [point]	Función que modifica el punto rojo cromático primario de la imagen.	Punto del color.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Sampling Factor [geometry]	Función que establece el factor	Factor de muestreo.	A partir de la experimentación	Función no favorable.

	de muestreo a usar con las codificaciones JPEG, YUV, MPEG.		no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	
Tile Offset [geometry]	Función que especifica un desplazamiento cuando se realiza una operación de llenado de una figura con otra imagen.	Valor del desplazamiento.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Tree Depth [color]	Función que establece el color del árbol de profundidad de la imagen.	Color a establecer.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Transparent [color]	Función que transparenta determinado color de la imagen.	Color a transparentar.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Undercolor [color]	Función que especifica propiedades de color a la hora de imprimir texto en la imagen.	Color del fondo tras los caracteres.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.

Verbose	Función que imprime información detallada de la imagen.	No tiene.	No aplica.	Función que no aplica.
Weight [type]	Función que envía un texto con determinado tamaño a la imagen.	Tamaño de la fuente.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
White Point [point]	Función para definir que color se utilizará como blanco, por ejemplo en operaciones relacionadas con el balance de blancos.	Coordenadas del punto.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Adaptive Blur [point]	Función que cambia la claridad a modo de nubosidad de los píxeles decrementando el efecto hacia los bordes de la imagen.	Factor de acrecentamiento de la nubosidad.	A partir de la experimentación dicha función, se pudo observar que la función puede actuar a modo de completar el trazo de las líneas de aquellos caracteres de líneas difusos. De este modo, esta función, puede realizar transformaciones favorables a las establecidas. Se observo que la transformación es sustancialmente más lenta que en el caso de la mayoría de las funciones relevadas.	Función que puede resultar favorable.
Adaptive Resize [geometry]	Función que modifica las	Valor de cambio de tamaño.	A partir de la experimentación	Función que puede resultar

	dimensiones de la imagen.		dicha función, se pudo observar que la función actúa sobre la imagen variando sus proporciones. No está claro si estos cambios pueden resultar provechosos para el reconocimiento, por ello no sería aconsejable desecharla.	favorable
Adaptive Sharpen [geometry]	Función que agudiza la imagen de forma adaptativa dependiendo del contexto.	Factor de enfoque.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Annotate [geometry] [text]	Función que escribe un texto en un sector de la imagen	Lugar y texto a escribir sobre la imagen.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Black-threshold [value]	Función que torna de color negro la imagen dependiendo del parámetro.	Factor para la conversión.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Version	Función que muestra información acerca de la versión	No tiene.	No tiene.	Función que no aplica.
Separate	Función que separa los canales de color dejando la imagen en	No tiene.	A partir de la experimentación se lograron ciertas transformaciones	Función que puede resultar favorable.

	escala de grises.		que podrían resultar favorables a lo establecido.	
Mosaic	Función que aplica un filtro de mosaico sobre la imagen.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Morph [value]	Función que aplica un filtro sobre la imagen, no queda explícitamente claro el tipo de filtro.	Factor de aplicación del filtro sobre la imagen.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Map [filename]	Función que realiza una transformación de colores de la imagen a través de un conjunto de colores de un segundo archivo.	Archivo que provee el conjunto de colores.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Fx [expression]	Función que aplica una expresión matemática a los canales de color de la imagen	Expresión matemática.	A partir de la experimentación con diversas expresiones matemáticas de distintos tipos y con distintos factores no se alcanzo una transformación favorable a lo establecido.	Función que no resulta favorable.

Flatten	Función que aplica un filtro de "aplanamiento" sobre la imagen.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Crop [geometry]	Función que retira una porción rectangular de la imagen.	Porción a quitar de la imagen.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable
Composite [filename]	Función que realiza una composición de imágenes.	Imagen utilizada para la composición.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Combine[filename]	Función que realiza una combinación de imágenes.	Imagen utilizada para la combinación.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Coalesce [filename]	Función que une una secuencia de imágenes.	Secuencia de imágenes utilizada para la unión.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.

Append[filename]	Función que concatena una secuencia de imágenes.	Secuencia de imágenes utilizada para la concatenación.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
White Threshold [value]	Que fuerza píxeles a tomar el color blanco.	Valor por el cual se opta los píxeles que serán forzados a blanco y los que no.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Wave [geometry]	Función que altera las proporciones de la imagen a través de una curva.	Factor de incidencia de la curva sobre la imagen.	A partir de la experimentación no se lograron transformaciones que resulten favorables a lo establecido.	Función no favorable.
Vignette [geometry]	Función que altera las los bordes de la imagen al estilo de viñetas.	Factor de incidencia de la proporción del borde modificado.	A partir de la experimentación no se lograron transformaciones que resulten favorables a lo establecido.	Función no favorable.
Unsharp [geometry]	Función que desagudiza los bordes y detalles de la imagen.	Valor que define al filtro.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.

Unique-colors	Función que descarta todos menos un píxel de cada color existente en la imagen	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Type [type]	Función que cambia el tipo de la imagen con respecto a sus colores, por ejemplo, en escala de grises.	Tipo de la imagen.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Trim []	Función que recorta los bordes de la imagen.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Transverse []	Función que voltea la imagen horizontalmente y la gira 270 grados.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.

Transpose []	Función que voltea la imagen verticalmente y la gira 90 grados.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Transparent [color]	Función que establece al color dado como transparente.	Color que pasará a ser transparente.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
transform []	Función que aplica una transformación afín.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Tint [value]	Función que tiñe una imagen con el color dado.	Color de la tintura.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	No favorable.
Tile [filename]	Función que repite una imagen para completar otra.	Nombre de archivo de la imagen.	A partir de la experimentación no se lograron cambios significativos de transformación	Función no favorable.

			sobre la imagen por lo que no parece favorablemente a lo establecido.	
Thumbnail [geometry]	Función que crea un pequeño boceto de la imagen.	Medidas del boceto.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Threshold [value]	Función que aplica un umbral sobre blanco y negro de forma tal que maximiza los valores de los píxeles que superan dicho umbral y minimiza al resto.	Umbral para la aplicación del filtro.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Swirl [degrees]	Función que produce un efecto de remolino en la imagen en torno al centro.	Grados que definen al remolino	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	No favorable.
Strip []	Función que le quita a la imagen los perfiles y comentarios que pueda tener.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Strip []	Función que le quita a la imagen	No tiene.	A partir de la experimentación	Función no favorable.

	los perfiles y comentarios que pueda tener.		no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	
Spread [radius]	Función que dispersa los píxeles de la imagen una distancia aleatoria.	Máximo para el desplazamiento	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Splice [geometry]	Función que une el color de fondo con la imagen.	Coordenadas que definen como será la unión.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Sparse color [method args]	Función que colorea una imagen a partir de los puntos y colores especificados.	Identificador del método a aplicar y sus argumentos.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Solarize [threshold]	Función que transforma todos los píxeles que superen un umbral por su negado.	Umbral para la aplicación del filtro	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no	Función no favorable.

			parece favorablemente a lo establecido.	
Sketch [geometry]	Función que simula un efecto de dibujo con lápiz en la imagen	Cantidad y características del efecto a aplicar.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Sigmoidal Contrast [geometry]	Función que modifica el contraste de la imagen sin saturar las luces o las sombras.	Intensidad del filtro.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Shear [threshold]	Función que desplaza un borde de la imagen sobre el eje X o el Y.	Coordenadas que definen el desplazamiento.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Shave [geometry]	Función que recorta una imagen desde sus bordes.	Coordenadas que definen el corte.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Sharpen [geometry]	Función que agudiza los	Umbral para la aplicación del	A partir de la experimentación	Función no favorable.

	bordes y detalles de la imagen.	filtro.	no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	
Shadow [geometry]	Función que genera una sombra en la imagen.	Umbral para la aplicación del filtro.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Shade [degrees]	Función que genera un efecto de sombra sobre los bordes de la imagen.	Intensidad del efecto.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Set [property value]	Función que modifica una propiedad de la imagen.	Identificador de la propiedad y su valor.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Sepia Tone [threshold]	Función que simula una imagen en tonos sepia	Umbral para la aplicación del filtro.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen	Función no favorable.

			por lo que no parece favorablemente a lo establecido.	
Selective Blur [geometry]	Función que aplica un filtro de niebla selectivamente con respecto a un umbral de contraste	Valor que define la forma de aplicar el filtro.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Segment [values]	Función que segmenta una imagen analizando sus histogramas de color.	Valores define el grado de segmentación.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Scale [geometry]	Función que escala la imagen.	Valores que definen la escala.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Sample [geometry]	Función que escala la imagen aplicando muestreo de píxeles.	Valores que definen la escala.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Rotate [degrees]	Función que rota la imagen.	Valor para la rotación en grados.	a partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido en el comienzo	Función que puede resultar favorable.
Roll [geometry]	Función que desplaza la	Valores que definen al	A partir de la experimentación	Función no favorable.

	imagen de forma horizontal o vertical.	desplazamiento.	no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	
Resize [geometry]	Función que cambia el tamaño de la imagen.	Valores que definen al tamaño.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Resample [geometry]	Función que cambia la resolución de la imagen.	Valores que definen la resolución.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Repage [geometry]	Utilizada para definir el área de trabajo en la imagen.	Valores que definen el área.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Render [ ]	Función que genera una imagen vectorial a partir de una matricial.	No tiene.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Region [geometry]	Función utilizada para aplicar las opciones solo a una porción de la imagen.	Valores que definen la región.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Recolor [matrix]	Función con la que se pueden aplicar distintas transformaciones	Matriz con los valores que definen las transformaciones.	A partir de la experimentación no se lograron cambios sobre la	Función no favorable.

	sobre los colores de la imagen.		imagen que resulten favorables a lo establecido.	
Random threshold [low, high]	Función que aplica un umbral con un valor aleatorio dentro del rango dado.	Rango que se utilizará para determinar el valor del umbral.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Raise [value]	Función que crea un efecto 3D oscureciendo y aclarando los bordes detectados en la imagen.	Intensidad del filtro a aplicar.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Radial Blur [angle]	Función que aplica un filtro de niebla en forma radial sobre la imagen.	Ángulo para el efecto de niebla.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Quantize [colorspace]	Función que reduce los colores de la imagen al espacio dado.	Nombre del espacio de colores.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Print [string]	Función que imprime una cadena de caracteres.	Cadena de caracteres a imprimir.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Posterize [levels]	Función que limita el número de niveles de	Numero de niveles de colores de la imagen.	A partir de la experimentación se lograron ciertas	Función que resulta favorable.

	colores de la imagen		transformaciones que claramente resultan favorables a lo establecido.	
Polaroid [angle]	Función que modifica la imagen dejándola similar a una fotografía de tipo polaroid.	ángulo de inclinación de la imagen generada	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Paint [radius]	Extiende los trazos de color de la imagen.	Factor que aumenta o disminuye la dilución de los colores.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Opaque [color]	Función que pinta de un color de la imagen.	Color utilizado por la función.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Normalize	Función que reduce el espectro de colores de la imagen.	No tiene.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Noise [radius]	Que agrega o quita ruido a la imagen.	Factor que modifica el efecto del ruido sobre la imagen.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.

Negate	Función que modifica cada píxel por su valor opuesto en la imagen.	No tiene.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Motion Blur [geometry]	Función que aplica un filtro de niebla sobre la imagen.	Factor de ajuste del filtro aplicado.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Monochrome	Función que modifica la imagen a la escala blanco y negro.	No tiene.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Modulate [value]	Función que modifica el brillo de la imagen.	Factor de ajuste inversamente proporcional al brillo resultante de la transformación.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Median [radius]	Función aplica un filtro basado en una distribución mediana que altera los colores de la imagen.	Valor que ajusta el efecto de la función.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Linear Stretch [value]	Función ajusta el contraste de la imagen a partir del estrechamiento de márgenes de	Factor de ajuste del estrechamiento de la saturación.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan	Función que resulta favorable.

	saturación.		favorables a lo establecido.	
Level [value]	Ajusta el contraste de la imagen.	Factor de ajuste para el contraste de la imagen.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
lat [geometry]	Función que realiza un filtro que altera el brillo de la imagen variando la zona de afectación	Factor utilizado para variar la zona en que actúa el filtro.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
lat [geometry]	Función que realiza un filtro que altera el brillo de la imagen variando la zona de afectación.	Factor utilizado para variar la zona en que actúa el filtro.	Partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Implode [amount]	Función que realiza un filtro que modifica la imagen aumentando desde el centro la decoloración de sus píxeles.	Factor utilizado para aumentar o disminuir el efecto del filtro.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Identify	Función despliega las características de la imagen.	No tiene.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función no aplica.
Geometry [geometry]	Función que modifica el tamaño de la imagen.	Factor utilizado para modificar el tamaño.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Gaussian-blur [geometry]	Función que reduce el ruido de	Valor utilizado para modificar el	A partir de la experimentación	Función que puede resultar

	la imagen.	efecto del filtro	se lograron ciertas transformaciones que podrían resultar favorables a lo establecido. a manera de eliminar ruido de la imagen. Cabe considerar que el tiempo de que toma ejecutar esta función varía dependiendo del valor del parámetro y para valores altos se torna excesivamente largo.	favorable.
Gamma [value]	Función que modifica la gama de colores de la imagen.	Valor utilizado para la transformación de la gama de colores.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Frame [geometry]	Función que rodea a la imagen con un marco.	Factor que modifica las dimensiones del marco.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Flop	Función que invierte verticalmente la imagen.	No tiene.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Floodfill [geometry] [color]	Función que completa con color la imagen.	Factor utilizado para calibrar el color y color de la transformación.	A partir de la experimentación no se lograron cambios sobre la	Función no favorable.

			imagen que resulten favorables a lo establecido.	
Flip	Función que invierte la imagen.	No tiene.	A partir de la experimentación no se lograron cambios sobre la imagen que resulten favorables a lo establecido.	Función no favorable.
Extract [geometry]	Función que extrae un sector de la imagen.	Factor utilizado para seleccionar el sector a extraer.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido. A manera de eliminar ruido situado en los bordes de la imagen extrayendo únicamente el sector donde se sitúan los caracteres.	Función que puede resultar favorable.
Extent [geometry]	Función que modifica el tamaño de la imagen.	Factor utilizado para el cambio de tamaño de la imagen.	A partir de la experimentación se lograron ciertas transformaciones de tamaño que podrían resultar favorables a lo establecido. a manera de eliminar ruido situado en los bordes de la imagen.	Función que puede resultar favorable.
Equalize	Función aplica un filtro de ecualización sobre la imagen.	No tiene.	A partir de la experimentación se lograron ciertas transformaciones que podrían	Función que puede resultar favorable.

			resultar favorables a lo establecido.	
Enhance	Función aplica un filtro digital para reducir ruido sobre la imagen.	No tiene.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Emboss [radius]	Función que aplica un filtro de grabado sobre la imagen	Factor que utiliza para el grabado.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable
Edge [radius]	Función que aplica un filtro para detectar bordes en la imagen	Factor que utiliza para detectar los bordes de la imagen.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Despeckle	Función que intenta reducir la cantidad de ruido en forma de puntos en la imagen.	No tiene.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Cycle [amount]	Función que desplaza el espacio de colores según el valor dado.	Desplazamiento del espacio de colores.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Convolve [coefficients]	Función que tuerce la imagen.	Coeficiente utilizado para torcer la imagen.	A partir de la experimentación no se lograron	Función no favorable.

			cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	
Contrast stretch [geometry	Función que modifica el contraste de la imagen teniendo acortando el rango de intensidad.	Estrechamiento del rango de intensidad.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Contrast	Función que modifica el contraste de la imagen.	No tiene.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.
Colorize [value]	Función que da colorea la imagen.	Factor utilizado para la coloración de la imagen.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Chop [geometry]	Función que remueve píxeles del interior de la imagen.	Factor utilizado para el ajuste de la herramienta.	A partir de la experimentación no se lograron cambios significativos de transformación sobre la imagen por lo que no parece favorablemente a lo establecido.	Función no favorable.
Charcoal [radius]	Función que realiza un efecto de filtro de tipo “carbón” sobre la imagen.	Radio de utilizado como factor de ajuste para el efecto.	A partir de la experimentación se lograron ciertas transformaciones que podrían	Función que puede resultar favorable.

			resultar favorables a lo establecido.	
Border [geometry]	Función añade un borde alrededor de la imagen	Dimensión del borde.	A partir de la experimentación dicha función no aplica favorablemente a lo establecido.	Función que no aplica.
Blur [geometry]	Función que nubla o hace menos definida a la imagen.	Factor para calibrar la potencia del filtro.	A partir de la experimentación se lograron ciertas transformaciones que claramente resultan favorables a lo establecido.	Función que resulta favorable.
Threshold [value]	Función que establece un umbral debajo del cual los colores son transformados al negro.	Factor para la conversión.	A partir de la experimentación se lograron ciertas transformaciones que podrían resultar favorables a lo establecido.	Función que puede resultar favorable.