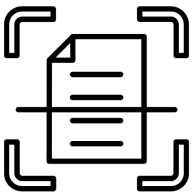
UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

TRABAJO FINAL MODELIZACIÓN

GRADO EN MATEMÁTICAS E INFORMÁTICA

QuadFix:
Rectificador de Imágenes

Autor: Carmen Toribio, Marcos Carnerero, María Moronta

Dirigido por: Jonatan Sanchez

Madrid, 9 de junio de 2025

QuadFix:
Rectificador de Imágenes
Autor: Carmen Toribio, Marcos Carnerero, María Moronta
Dirigido por: Jonatan Sanchez
Trabajo Final Modelización, 9 de junio de 2025

Si deseas citar este trabajo, la entrada completa en BibTeX es la siguiente:

```
@mastersthesis{citekey,
title = {QuadFix:
Rectificador de Imágenes},
author = {Toribio, Carmen, Carnerero, Marcos, Moronta, María y Sanchez, Jonatan},
school = {Escuela Técnica Superior de Ingenieros Informáticos},
year = {2025},
type = {Trabajo Final Modelización}
}
```

Índice general

Índice de figuras

1

Introducción

Hoy en día, es muy común usar dispositivos móviles para tomar fotos de **objetos planos**, como folios, pizarras o carteles. Sin embargo, estas imágenes suelen tener **distorsiones** por el ángulo desde el que se toman, lo que hace que un documento se vea como una figura irregular. Este proyecto soluciona ese problema utilizando una técnica llamada **rectificación proyectiva**, que permite recuperar la **forma original** de esos objetos planos.

Para solucionar este problema, desarrollamos una herramienta llamada **QuadFix**, que combina conceptos de **geometría proyectiva** con un enfoque interactivo. El usuario simplemente selecciona las **cuatro esquinas** del objeto en la imagen y el sistema aplica una transformación llamada **homografía** para generar una nueva imagen con una vista frontal, manteniendo las **proporciones** del objeto.

Para crear esta herramienta usamos **Python** y algunas librerías importantes. **NumPy** nos ayudó a hacer cálculos matriciales de manera eficiente, lo que fue clave para resolver el sistema de ecuaciones que permite la transformación. Por otro lado, **PySide6 (Qt)** nos permitió construir una **interfaz gráfica** sencilla y fácil de usar, que incluye funciones como la selección de puntos, el ajuste de proporciones y la **visualización en tiempo real** de los cambios.

El desarrollo se dividió en tres partes principales. Primero, implementamos la **rectificación básica**, que corrige cualquier cuadrilátero irregular y lo transforma en un cuadrado usando homografías. Luego, extendimos el sistema para manejar **proporciones específicas**, como los rectángulos con una relación de aspecto particular, por ejemplo el formato **DIN A4**, que tiene una proporción aproximada de $\sqrt{2} : 1$. Por último, diseñamos una **interfaz** que acompaña al usuario durante todo el proceso, desde cargar la imagen hasta guardar el resultado.

A lo largo de este documento se describen con detalle los **fundamentos teóricos** que sustentan la rectificación proyectiva, el **desarrollo técnico** dividido en sus distintas fases, y se presentan **ejemplos prácticos** que ilustran la funcionalidad y utilidad de la herramienta.

2

Fundamentos Teóricos

El presente capítulo establece los **fundamentos teóricos** sobre los cuales se construye el desarrollo del sistema de **rectificación de imágenes** propuesto. Para comprender el funcionamiento y las decisiones de diseño adoptadas, es imprescindible revisar conceptos clave de la **geometría proyectiva**, las **homografías** y las **transformaciones métricas**. Estas herramientas matemáticas permiten modelar y revertir las deformaciones por perspectiva que se producen en fotografías de objetos planos, facilitando así su representación frontal sin distorsiones. A través de este marco teórico se justifica tanto la **viabilidad del proceso de rectificación** como las **simplificaciones prácticas** implementadas en el sistema, que buscan un equilibrio entre rigor matemático, facilidad de uso y eficiencia computacional.

2.1. Rectificación de Imágenes

La **rectificación proyectiva** es el proceso mediante el cual se elimina la distorsión causada por la perspectiva en una imagen, con el fin de obtener una representación frontal y sin deformaciones de un objeto plano. Este proceso es fundamental cuando se trabaja con imágenes tomadas desde ángulos no perpendiculares, como ocurre frecuentemente al fotografiar documentos, pizarras o carteles con dispositivos móviles.

Desde un punto de vista teórico, una imagen tomada con una cámara convencional es una proyección central del mundo tridimensional sobre un plano bidimensional. Debido a esta proyección, las líneas paralelas en el mundo real pueden aparecer convergentes en la imagen (fenómeno conocido como *perspectiva*), y las formas geométricas regulares se ven deformadas, transformándose en cuadriláteros irregulares.

La rectificación consiste en encontrar una transformación planar —específicamente, una homografía— que deshaga esta proyección, reconstruyendo la vista frontal del objeto plano y restaurando sus propiedades métricas, como la forma y las proporciones.

Matemáticamente, si el objeto plano en el mundo real puede representarse mediante un sistema de coordenadas en un plano, la imagen proyectada corresponde a una transformación de ese plano mediante una homografía H. Si $\mathbf{x} = (x, y, 1)^T$ es un punto en la imagen distorsionada, existe una homografía H tal que

$$\mathbf{x}' = H\mathbf{x}, \tag{2.1}$$

donde $\mathbf{x}'$ corresponde al punto en la imagen rectificada, idealmente en la posición que tendría si la cámara estuviera perpendicular al objeto. Para calcular esta homografía, se requieren al menos cuatro puntos no colineales en la imagen original y sus correspondientes puntos en la imagen deseada (normalmente, los vértices de un rectángulo que representa la forma real del objeto). Esta correspondencia permite resolver un sistema de ecuaciones lineales que determina los parámetros de H, pero este tema se tratará con más detalle en el punto a continuación.

El proceso de rectificación implica entonces tres pasos fundamentales:

1. **Identificación de correspondencias:** Selección de los cuatro puntos en la imagen original que delimitan el objeto plano.

2. **Cálculo de la homografía:** Resolución del sistema lineal para encontrar la matriz H que mapea esos puntos a un rectángulo con las proporciones reales.

3. **Aplicación de la transformación:** Uso de H para transformar toda la imagen, corrigiendo la perspectiva y generando la imagen rectificada.

Es importante destacar que esta transformación preserva la linealidad (es decir, las líneas rectas permanecen rectas), pero no garantiza la conservación de distancias o ángulos, ya que una homografía general sólo preserva propiedades proyectivas. Sin embargo, en aplicaciones como la nuestra, donde se busca que el objeto rectificado mantenga sus **proporciones reales**, es necesario realizar ajustes adicionales para **restaurar la métrica euclidiana**. Para ello, se pueden emplear técnicas que estiman la *línea del infinito* y los *puntos de fuga*, permitiendo recuperar propiedades métricas del plano original.

Durante la aplicación de una transformación proyectiva, es necesario reasignar el valor de cada píxel en la imagen rectificada, ya que las coordenadas resultantes suelen ser valores reales y no enteros, que no corresponden directamente a posiciones exactas en la imagen original. Para resolver este problema se utilizan técnicas de **interpolación**, cuyo objetivo es estimar el valor del píxel en una posición no entera a partir de los valores de píxeles vecinos con coordenadas enteras.

Uno de los métodos más simples y eficientes es la **interpolación por vecino más cercano** (nearest neighbor). Este método consiste en asignar al píxel transformado el valor del píxel original que se encuentra en la posición entera más próxima a las coordenadas reales calculadas.

Por otro lado, la rectificación de imágenes es sumamente útil en múltiples aplicaciones:

- **Digitalización de documentos:** Fotografías tomadas con smartphones que deforman folios o pizarras pueden corregirse para obtener imágenes limpias y planas.
- **Fotogrametría:** Permite reconstruir modelos 3D a partir de imágenes 2D mediante la corrección de perspectivas.
- **Diseño gráfico:** Facilita la corrección de perspectivas en carteles, logotipos u otros elementos visuales deformados.

En resumen, la rectificación proyectiva es una herramienta matemática y computacional esencial para convertir imágenes deformadas por perspectiva en representaciones frontales precisas, facilitando la digitalización, el análisis y la manipulación de objetos planos.

2.2. Geometría Proyectiva y Homografías

Como se mencionó anteriormente, la herramienta fundamental para la rectificación de imágenes es la **homografía**, una transformación proyectiva que proviene de la **geometría proyectiva**. Esta rama de las matemáticas estudia las propiedades de las figuras que permanecen invariantes bajo transformaciones proyectivas, como las perspectivas que vemos en fotografías.

A diferencia de la geometría euclidiana —que conserva distancias, ángulos y paralelismo—, la geometría proyectiva preserva propiedades más generales, como la **incidencia** (puntos alineados sobre una misma recta), la **colinealidad** y la **razón doble** (una proporción invariante entre cuatro puntos colineales bajo transformaciones proyectivas).

En este contexto, una **homografía** es una función que relaciona dos planos proyectivos mediante una matriz 3×3. Esta matriz actúa sobre puntos expresados en coordenadas homogéneas, lo que permite manejar de forma compacta transformaciones que incluyen proyecciones y efectos de perspectiva.

Sea un punto en coordenadas homogéneas $\mathbf{x} = (x, y, w)^T$ en el plano original, la homografía H produce el punto transformado:

$$\mathbf{x}' = H\mathbf{x} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix}. \tag{2.2}$$

Esta transformación tiene la propiedad fundamental de preservar la **linealidad**, es decir, las líneas rectas en el plano original se mapean en líneas rectas en el plano transformado. Sin embargo, no se garantiza la conservación de distancias, ángulos ni proporciones, lo que implica que la imagen resultante puede estar deformada desde un punto de vista métrico.

Cálculo de la homografía

En nuestro sistema, la matriz de homografía H se calcula directamente a partir de información geométrica obtenida de la imagen, mediante un proceso basado en álgebra lineal. El procedimiento consta de los siguientes pasos principales:

1. **Selección de puntos de entrada:** El usuario proporciona cuatro puntos no colineales que definen el contorno del objeto plano a rectificar (por ejemplo, los vértices de un folio o cartel en una imagen tomada en perspectiva).

2. **Cálculo de puntos de fuga:** Se construyen las rectas que corresponden a los lados opuestos del cuadrilátero, y se obtienen sus intersecciones en coordenadas homogéneas. Estas intersecciones definen los *puntos de fuga* $\mathbf{v}_1$ y $\mathbf{v}_2$, que caracterizan la dirección de las líneas paralelas en el espacio original:

$$\ell_1 = \mathbf{x}_1 \times \mathbf{x}_2, \quad \ell_2 = \mathbf{x}_3 \times \mathbf{x}_4, \quad \Rightarrow \quad \mathbf{v}_1 = \ell_1 \times \ell_2,$$

$$\ell_3 = \mathbf{x}_2 \times \mathbf{x}_3, \quad \ell_4 = \mathbf{x}_4 \times \mathbf{x}_1, \quad \Rightarrow \quad \mathbf{v}_2 = \ell_3 \times \ell_4.$$

   Aquí, $\times$ denota el producto cruzado en coordenadas homogéneas.

3. **Construcción del sistema lineal:** Se plantea un sistema de ecuaciones para determinar los coeficientes de la matriz H, tal que:

   - $\mathbf{v}_1$ y $\mathbf{v}_2$ se transformen en las direcciones canónicas $[1, 0, 0]^T$ y $[0, 1, 0]^T$.
   - Dos vértices del cuadrilátero (por ejemplo, $\mathbf{x}_1$ y $\mathbf{x}_3$) se fijen en coordenadas específicas del plano destino, como $(0, 0)$ y $(1, 1)$, lo cual establece la escala y orientación de la imagen rectificada.

   La resolución de este sistema proporciona directamente la matriz de homografía H.

4. **Ajuste de razón de aspecto:** Para garantizar que la imagen rectificada refleje fielmente las proporciones reales del objeto (por ejemplo, un folio DIN A4 o un cartel cuadrado), se aplica una transformación de escala anisotrópica mediante la siguiente matriz:

$$S = \begin{pmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

donde los factores b y c se escogen de forma que la razón c/b coincida con la razón de aspecto deseada (proporción altura/ancho). La homografía final se obtiene entonces como:

$$H' = S \cdot H.$$

Este enfoque directo resulta más eficiente y fácil de controlar en la práctica, ya que permite obtener una transformación completa sin descomponerla en pasos intermedios. Refleja fielmente la implementación del sistema y proporciona una base sólida para una rectificación proyectiva robusta.

2.3. Preservación Métrica y Razón de Aspecto

Aunque una homografía general garantiza la preservación de propiedades proyectivas como la colinealidad y la incidencia, no conserva en general la métrica euclidiana, es decir, las distancias y ángulos reales. Sin embargo, en muchas aplicaciones prácticas (entre ellas, la nuestra), es fundamental que la imagen rectificada mantenga las proporciones del objeto.

Para ello, es necesario considerar conceptos como los **puntos de fuga**, también conocidos como puntos en el infinito, y la **línea del infinito** en el plano proyectivo. Las líneas paralelas en el mundo real convergen en puntos de fuga en la imagen, debido a la perspectiva. Su identificación permite localizar la línea del infinito, fundamental para restaurar propiedades métricas como los ángulos rectos y las proporciones reales.

En teoría, si se conoce la línea del infinito, se puede aplicar una transformación adicional denominada **homografía métrica** que recupera la métrica euclidiana. Sin embargo, dado que en nuestro proyecto el usuario define manualmente las cuatro esquinas del objeto, adoptamos un enfoque práctico basado en la especificación directa de la razón de aspecto deseada. El usuario introduce la proporción altura/ancho (*aspect ratio*) real del objeto, por ejemplo:

- Cuadrado: 1 : 1

- Folio DIN A4 (vertical): $\approx \sqrt{2} : 1 \approx 1{,}4142 : 1$

Esta razón se incorpora en la homografía final mediante una **homotecia**, que es una transformación afín que realiza un escalado diferencial en las direcciones del plano, ajustando así las dimensiones de la imagen rectificada para que cumpla con la proporción deseada. Matemáticamente, esta homotecia se representa mediante la matriz de escala diagonal que se encuentra a continuación

$$S = \begin{pmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

donde los factores b y c se eligen para que la razón b/c coincida con la proporción especificada. Por ejemplo, si $r = \text{alto}/\text{ancho}$ es la razón deseada, se puede tomar $b = 1$ y $c = r$ (o viceversa según normalización).

Otro concepto relevante dentro de la geometría proyectiva es la **razón doble**, un *invariante proyectivo* que se mantiene constante bajo transformaciones proyectivas y que, en teoría, puede utilizarse para recuperar información métrica (como proporciones o distancias relativas) sin necesidad de conocer previamente las dimensiones reales del objeto. No obstante, en este proyecto hemos decidido no emplearla por varias razones:

- **Simplicidad y usabilidad:** Incluir el cálculo de la razón doble implicaría pedir al usuario que identifique puntos adicionales con precisión (por ejemplo, cuatro puntos colineales con una distribución significativa), lo cual complicaría considerablemente tanto la interfaz como la experiencia de uso. Nuestro enfoque busca minimizar la intervención técnica por parte del usuario.

- **Precisión práctica:** En la mayoría de los casos de uso contemplados, como la rectificación de documentos, el usuario ya conoce la **razón de aspecto** del objeto (pues están escritas las proporciones más básicas en un apartado de la interfaz de usuario llamado “Instrucciones”). Esta proporción es más intuitiva, fácil de introducir y menos propensa a errores que intentar estimar una razón doble sobre una imagen deformada.

- **Eficiencia computacional:** La estimación de la razón doble requiere identificar características adicionales (como líneas paralelas u ortogonales), lo que implicaría procesos más complejos y costosos en términos de cómputo. En cambio, aplicar una **homotecia** (un escalado uniforme o anisotrópico) sobre la imagen rectificada a partir de una razón de aspecto conocida es inmediato y computacionalmente eficiente.

Este método es robusto y fácil de usar, pues requiere solo cuatro puntos y una razón de aspecto conocida. No obstante, si la razón introducida no es correcta, la imagen puede resultar estirada o comprimida. Para minimizar este riesgo, la interfaz ofrece valores predefinidos comunes, como *Cuadrado* o *DIN A4*.

En resumen, la combinación de homografías con ajustes métricos mediante homotecias ofrece un **equilibrio entre precisión y simplicidad**, permitiendo rectificar imágenes que conservan las proporciones reales del objeto con un procedimiento práctico y efectivo.

3

Desarrollo del Proyecto

El desarrollo del proyecto se llevó a cabo en tres fases bien diferenciadas, con el objetivo de construir progresivamente una herramienta robusta, funcional y orientada a la experiencia del usuario. A través de una metodología incremental, se fueron implementando las funcionalidades principales desde una primera versión básica hasta llegar a una aplicación completa y adaptativa.

3.1. Fase 1: Rectificación Básica

Objetivo: Implementar la funcionalidad base que permitiera rectificar imágenes de superficies cuadradas a través de transformaciones proyectivas.

Tareas Realizadas:

- **Creación de la ventana principal:**
  Se diseñó una interfaz gráfica básica utilizando la biblioteca `PySide6`, que permite combinar Python con Qt para el desarrollo de interfaces modernas. Esta ventana principal muestra la imagen cargada por el usuario y permite interactuar con ella. Para ello, se implementó la clase `ClickArea`, para gestionar la interacción con la imagen y permitir seleccionar puntos sobre ella mediante clics del ratón.

- **Captura y gestión de coordenadas:**
  Para estructurar y manipular las coordenadas de manera eficiente, se desarrolló la clase `Coordinates`. Esta clase encapsula toda la lógica relacionada con el almacenamiento y validación de los puntos seleccionados por el usuario. Entre sus funcionalidades principales se incluye la capacidad de **almacenar hasta cuatro puntos**, que son aquellos seleccionados mediante clics.

Además, la clase permite **eliminar un punto** si el usuario hace clic nuevamente sobre él, facilitando la corrección y modificación de la selección. Finalmente, incorpora una **validación automática** que asegura que se hayan seleccionado exactamente cuatro puntos antes de proceder con la transformación, garantizando así la integridad de los datos para las operaciones posteriores.

- **Cálculo de los puntos de fuga:**
  El cálculo de los puntos de fuga se implementó en el módulo `infinity_line.py`, en la función `calculate_vanish_points`. Esta función recibe como entrada cuatro puntos que definen un cuadrilátero sobre la imagen y calcula las intersecciones de los pares de lados opuestos. En concreto, se obtienen dos puntos de fuga mediante la intersección de las siguientes rectas:

  - La recta que une los puntos p_1 y p_2 con la que une p_3 y p_4.
  - La recta que une los puntos p_1 y p_3 con la que une p_2 y p_4.

  Estas intersecciones se resuelven mediante álgebra lineal usando la regla de Cramer en **coordenadas cartesianas**, sin utilizar coordenadas homogéneas. El resultado se devuelve como puntos con $w = 1$, pero los cálculos no son proyectivos en sentido estricto.

  En el caso de que las líneas sean paralelas (es decir, cuando el determinante del sistema es cero), la función lanza un **error**, ya que no se maneja actualmente la existencia de puntos de fuga en el infinito.

  Los dos puntos de fuga obtenidos son utilizados posteriormente para definir la recta del infinito en otros módulos. Aunque no se calcula explícitamente la recta proyectiva como producto vectorial (e.g. con `np.cross(p, q)` en coordenadas homogéneas), esta operación puede incorporarse en el futuro para lograr una interpretación más rigurosa desde la geometría proyectiva.

- **Transformación proyectiva:**
  En el módulo `projective_transform.py`, se desarrollaron dos funciones clave:

  - `calculate_homography`: Calcula la **matriz de homografía** que permite transformar un cuadrilátero irregular (producto de la perspectiva) en un cuadrado perfecto en la nueva imagen. La función utiliza una base de referencia estándar y resuelve el sistema mediante **álgebra lineal**, sin descomponer la transformación en pasos intermedios explícitos como H_1, H_2, H_3.
    La homografía se calcula mapeando:

- Los **puntos de fuga**, obtenidos como intersecciones de lados opuestos del cuadrilátero, a direcciones canónicas $[1, 0, 0]$ y $[0, 1, 0]$.
- Dos **esquinas del cuadrilátero** (específicamente p_0 y p_3) a posiciones fijas que determinan la escala y orientación del plano rectificado.

Finalmente, se aplica una **matriz de escala diagonal** (escalado anisotrópico) para ajustar la **razón de aspecto** del plano rectificado. Este escalado no representa una transformación afín completa, sino una modificación de la proporción entre ejes horizontal y vertical.

- `warp_perspective_qpixmap`: Aplica la homografía calculada a la imagen original para obtener una imagen **rectificada**. Esta función implementa la transformación inversa para cada píxel del plano destino: se aplica la **inversa de la homografía** para mapear cada píxel en la imagen de salida a su posición correspondiente en la imagen original. Este enfoque evita **huecos** o **solapamientos**, al garantizar que cada píxel destino proviene de una ubicación definida en la imagen fuente.

  Para determinar el valor del color correspondiente a cada coordenada mapeada, se usa el método de **interpolación por vecino más cercano** (*nearest neighbor*), que selecciona el valor del píxel más próximo en la imagen original. Aunque este método es simple y rápido de implementar, puede generar artefactos visuales (escalonamientos) y resulta **computacionalmente ineficiente** en imágenes de gran tamaño. La interpolación más avanzada (bilineal o bicúbica) podría considerarse en futuras versiones para mejorar la calidad visual del resultado.

- **Integración y pruebas:**
  Una vez implementadas todas las funciones anteriores, se integraron en el flujo de trabajo principal de la aplicación. El usuario podía:

  1. Cargar una imagen.
  2. Seleccionar 4 puntos sobre la imagen.
  3. Hacer clic en el botón “Rectificar”.

  La aplicación realizaba la transformación y mostraba la imagen corregida. Se llevaron a cabo pruebas con imágenes que contenían superficies cuadradas en distintas perspectivas, confirmando la efectividad de la transformación proyectiva.

3.2. Fase 2: Adaptación a Otras Proporciones

Objetivo: Generalizar la herramienta para que pueda rectificar no solo cuadrados, sino también rectángulos con proporciones específicas. Para ello, se añadió la posibilidad de que el usuario introduzca manualmente la razón de aspecto deseada a través de la interfaz gráfica, permitiendo así adaptar la rectificación a cualquier proporción que se requiera.

Tareas Realizadas:

- **Introducción de la razón de aspecto:**
  Se añadió un nuevo widget llamado `AspectRatioWidget` dentro del archivo `menu.py`, que permite al usuario introducir manualmente una razón de aspecto (alto/ancho). Esta se utiliza para adaptar la referencia proyectiva a las proporciones deseadas.
- **Ajuste de la homografía:**
  La función `calculate_homography` fue extendida para admitir una razón de aspecto como parámetro adicional. Esta razón permite ajustar la base estándar, originalmente un cuadrado, para transformarla en un rectángulo con las proporciones deseadas.

  De este modo, se garantiza que la homografía resultante respete las proporciones especificadas por el usuario. Además, se incorpora un factor de escala que mantiene el tamaño visual coherente dentro de la imagen transformada, asegurando que la rectificación no distorsione la apariencia del resultado final.
- **Pruebas con rectángulos:**
  Se realizaron múltiples pruebas con diferentes proporciones. Los resultados confirmaron que la herramienta puede adaptarse a una amplia variedad de formatos manteniendo una alta precisión en la rectificación.

3.3. Fase 3: Interfaz de Usuario

Objetivo: Mejorar la experiencia del usuario con una interfaz más atractiva, intuitiva y coherente con las convenciones modernas de diseño.

Tareas Realizadas:

- **Diseño de la interfaz:**
  Se reestructuró la ventana principal para incluir un menú lateral (`Menu` en `menu.py`). Este menú contiene:

- Título de la aplicación con su logo (que puede verse en la Figura 3.1).
- Instrucciones de uso, en un apartado desplegable (apreciable en la Figura 3.2).
- Campo para introducir la razón de aspecto.
- Botones para cargar imagen, aplicar la transformación, borrar todos los puntos seleccionados y descargar la imagen una vez ha sido rectificada.

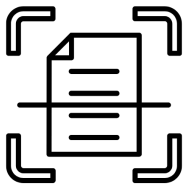


Figura 3.1. Logo de la aplicación

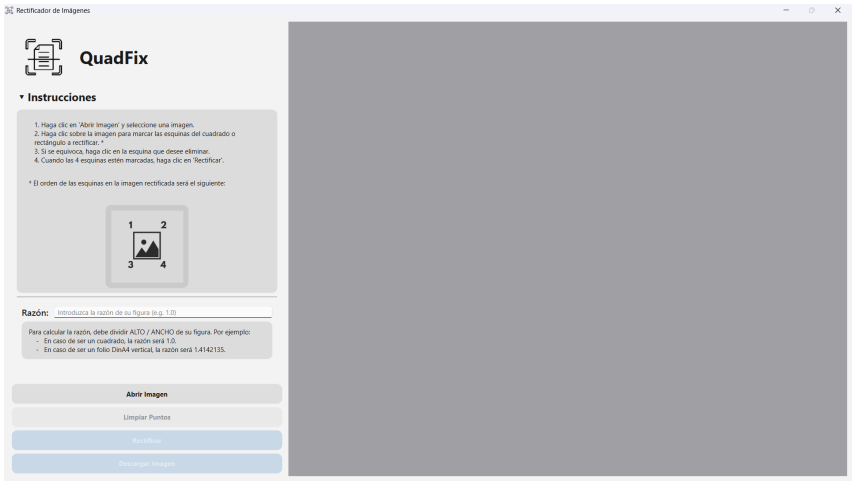


Figura 3.2. Pantalla principal con el apartado "Instrucciones" desplegado

También se definieron dos tipos de botones personalizados, como puede apreciarse en la Figura 3.2 (`PrimaryButton`, de color azul, y `SecondaryButton`, de color negro, en `buttons.py`) para mejorar la coherencia visual y la jerarquía de acciones.

- **Modo claro y oscuro:**
  La aplicación detecta el tema del sistema operativo mediante la función `is_dark_mode` (archivo `dark_mode.py`) y adapta automáticamente los iconos, los colores del fondo, texto y contenedores y los estilos de los widgets para mantener la legibilidad. En las Figuras 3.3 y 3.4 pueden verse los dos modos.

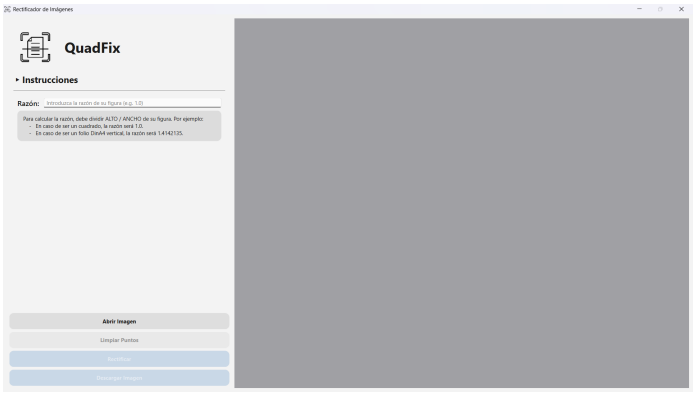


Figura 3.3. Modo Claro de la aplicación

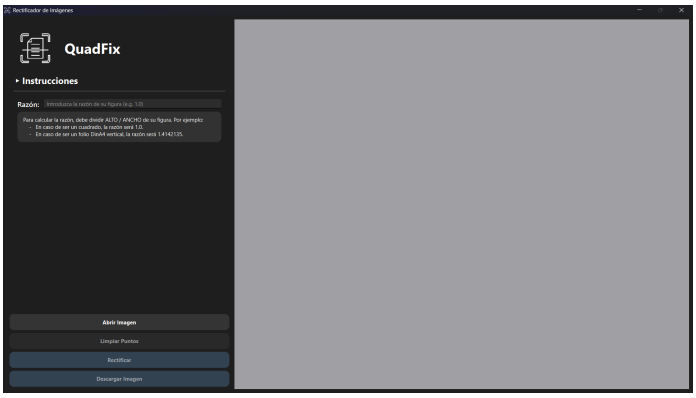


Figura 3.4. Modo Oscuro de la aplicación

- **Feedback visual al usuario:**
  Se incorporaron diversos mecanismos para mejorar la comunicación con el usuario durante la interacción con la aplicación. Entre estos, destaca un *widget* de carga, implementado como `OverlayWidget` en el módulo `click_area.py`, que se muestra mientras la imagen está siendo procesada.

  Además, se añadieron mensajes de error que alertan al usuario en caso de que el número de puntos seleccionados sea incorrecto, si la imagen no ha sido cargada correctamente, o si la descarga de la imagen no ha podido completarse. Por último, se implementaron confirmaciones visuales que informan al usuario sobre el éxito de las transformaciones realizadas o la descarga correcta de la imagen.

- **Optimización y detalles finales:**
  Se ajustaron los márgenes, tipografías, colores y alineaciones para garantizar un

diseño limpio y profesional. Además, se añadió una función para guardar la imagen rectificada en el sistema del usuario, mejorando la utilidad práctica de la herramienta.

3.4. Conclusión del Desarrollo

A lo largo de las tres fases, el proyecto evolucionó desde una funcionalidad básica centrada en la geometría proyectiva hasta convertirse en una aplicación visualmente pulida, práctica y adaptable a distintos contextos. Cada fase aportó un avance significativo:

- **Fase 1:** Estableció las bases matemáticas y funcionales.
- **Fase 2:** Amplió la versatilidad con soporte para diferentes formatos.
- **Fase 3:** Cerró el desarrollo con una interfaz moderna y experiencia de usuario cuidada.

El resultado final es una herramienta que combina precisión matemática con una usabilidad óptima, apta para tareas de corrección de perspectiva en distintos formatos de imagen.

4

Ejemplos de uso

Por hacer aún

5

Conclusiones

El proyecto *Rectificador de Imágenes* (QuadFix) ha logrado su objetivo principal: crear una herramienta intuitiva y eficiente para rectificar imágenes que contienen rectángulos mediante transformaciones proyectivas. A lo largo del desarrollo, se implementaron algoritmos avanzados de visión por computadora y se diseñó una interfaz gráfica amigable que facilita la interacción con el usuario. Además, se incorporaron funcionalidades importantes como el ajuste dinámico de proporciones y el soporte para modos visuales claros y oscuros, lo que contribuye a mejorar tanto la adaptabilidad como la experiencia general del usuario.

5.1. Logros Principales

En cuanto a los resultados obtenidos, el sistema es capaz de calcular con precisión la matriz de homografía necesaria para transformar un cuadrilátero seleccionado por el usuario en una figura regular, ya sea un cuadrado o un rectángulo con una proporción definida. La detección de puntos de fuga y la aplicación de la transformación proyectiva se ejecutan correctamente incluso en imágenes que presentan perspectivas complejas, garantizando la robustez del proceso.

La interfaz desarrollada facilita una interacción fluida, permitiendo al usuario seleccionar los puntos de interés, ajustar las proporciones deseadas y guardar los resultados obtenidos. Además, la inclusión de instrucciones claras y mecanismos de retroalimentación visual, como la indicación de carga durante el procesamiento, contribuyen a una experiencia de usuario más satisfactoria y comprensible.

El sistema también demuestra una notable adaptabilidad, dado que soporta diversas proporciones, incluyendo valores típicos como 1.0 para cuadrados y aproximadamente $\sqrt{2}$ para el formato DIN A4. Adicionalmente, la integración con el tema del sistema

operativo permite que la aplicación se adapte automáticamente al modo claro u oscuro, proporcionando una apariencia coherente con la configuración del usuario.

Finalmente, en términos de rendimiento, la aplicación presenta tiempos de procesamiento aceptables, gracias a la utilización de un método de interpolación sencillo.

5.2. Limitaciones Actuales

No obstante, existen algunas limitaciones en la implementación actual que deben ser consideradas. En primer lugar, la interpolación basada en el vecino más cercano, aunque simple y rápida, puede generar efectos de aliasing. En imágenes, esto suele manifestarse como bordes o líneas "dentadas" o "escalonadas" en lugar de ser suaves, lo que afecta la calidad visual del resultado final.

Por otro lado, se eligió el método del vecino más cercano por su simplicidad de implementación inicial, pero limita la calidad visual. Futuras versiones priorizarán métodos más avanzados (bilineal/bicúbica) para mejorar la resolución de las imágenes rectificadas.

Además, el manejo de errores aún es limitado. Si el usuario selecciona puntos en un orden incorrecto o si la geometría de los puntos es inválida, como en el caso de líneas casi paralelas, la rectificación se llevaría a cabo de forma errónea sin ofrecer indicaciones claras para corregir la selección, lo que puede dificultar el uso por parte de usuarios menos experimentados.

5.3. Mejoras Futuras

Para superar estas limitaciones y ampliar las capacidades del proyecto, se proponen diversas mejoras para futuras versiones. En términos de rendimiento, se contempla la implementación de métodos de interpolación más avanzados, como la interpolación bilineal o bicúbica, que mejorarán la calidad visual de la imagen rectificada.

Además, se ha identificado que el método actual de aplicar la homografía de forma **pixel por pixel** es **computacionalmente ineficiente**, especialmente para imágenes de alta resolución. Esta ineficiencia afecta el tiempo total de procesamiento y la escalabilidad de la aplicación.

Como propuesta de mejora, se sugiere la **paralelización de los cálculos**, aprovechando técnicas como la aceleración por GPU mediante tecnologías como `CUDA` o `OpenCL`, así como el uso de bibliotecas optimizadas para procesamiento de imágenes, como `OpenCV`.

También se recomienda implementar métodos de interpolación que utilicen operaciones matriciales en lugar de bucles explícitos, lo que puede traducirse en una reducción significativa del tiempo de cómputo y en una mejor calidad visual.

Desde el punto de vista funcional, se planea incorporar la **detección automática de esquinas** mediante algoritmos, facilitando la selección inicial de puntos por parte del usuario. Además, se contempla ofrecer herramientas para el ajuste manual post-transformación, permitiendo rotaciones y escalados finos para corregir imperfecciones.

También se espera ampliar el soporte para **rectificaciones de polígonos irregulares**, como trapezoides, incrementando la versatilidad del sistema.

En lo que respecta a la **experiencia de usuario**, se propone implementar mejoras que faciliten la interacción y el control sobre el proceso de rectificación. Entre estas mejoras destacan la incorporación de una funcionalidad de deshacer que permita retroceder en los pasos realizados durante la rectificación, la posibilidad de aplicar diversos filtros a las imágenes para mejorar su apariencia o destacar detalles específicos, y la integración de herramientas adicionales para ajustar manualmente aspectos como el brillo, contraste y saturación. Finalmente, se considera ampliar las opciones de exportación, incluyendo formatos profesionales como PDF o SVG.

Estas mejoras tienen como objetivo ofrecer una experiencia más fluida, flexible y personalizada, adaptándose a las necesidades de cada usuario.

5.4. Conclusión Final

En resumen, QuadFix demuestra que es posible combinar fundamentos matemáticos sólidos, basados en geometría proyectiva, con herramientas modernas de programación y procesamiento de imágenes, para crear una aplicación utilizable en nuestro día a día. Aunque la versión actual cumple con los requisitos básicos planteados, las mejoras propuestas tienen el potencial de transformar la herramienta en un producto profesional apto para ámbitos como el diseño, la arquitectura o la fotogrametría.

Además, el proyecto sirve como una base prometedora para la exploración de técnicas más avanzadas en visión por computadora, tales como el uso de redes neuronales para la rectificación automática o la integración con sistemas de realidad aumentada, abriendo nuevas vías para la innovación y aplicación en este campo.