Análisis Lineal del Código: `config_manager.py`

Este archivo, `config_manager.py`, no ha sido mencionado en los otros módulos, pero su inclusión es una práctica de desarrollo de software muy recomendada y elegante. Su propósito es externalizar la configuración del código fuente. En lugar de tener valores como el índice de la cámara o los timeouts escritos directamente en los archivos `.py` (como en `constants.py`), este módulo permite que se lean de un archivo de texto externo, típicamente llamado `config.ini`. Esto ofrece una ventaja enorme: un usuario final o un administrador del sistema puede cambiar parámetros clave sin necesidad de editar el código Python y sin riesgo de romper la aplicación.

El script comienza importando la librería `configparser`, que es la herramienta estándar de Python para leer y escribir archivos de configuración con formato `.ini`. También importa `os` para manejar rutas de archivos y `tkinter.messagebox` para poder notificar al usuario si ocurre un error al leer el archivo de configuración.

La variable global `config_global` se define para mantener el objeto de configuración una vez que ha sido cargado, haciéndolo accesible a las otras funciones del módulo.

La primera función, y una de las más importantes, es `crear_config_por_defecto(nombre_archivo)`. Esta es una función de "auto-configuración". Su trabajo es crear un archivo `config.ini` con valores predeterminados y comentarios explicativos si el archivo no existe en la primera ejecución. Está estructurado en secciones como `[General]`, `[Sensores]`, `[Seguridad]`, etc., lo que hace que el archivo sea muy fácil de entender y editar para un humano. Esta función es crucial para la experiencia del usuario, ya que no le obliga a crear el archivo desde cero.

La función principal del módulo es `cargar_configuracion(nombre_archivo)`. Esta función se encarga de leer el archivo `config.ini`. Utiliza un bloque `try...except` para manejar el caso en que el archivo no exista. Si no lo encuentra, llama a `crear_config_por_defecto()` para generarlo y luego intenta leerlo de nuevo. Este enfoque hace que la aplicación sea muy robusta. El contenido del archivo es parseado y cargado en la variable global `config_global`.

La función más utilizada por otros módulos sería `get_config_valor(seccion, clave, tipo=str, defecto=None)`. Esta es una función "getter" segura y versátil. En lugar de que otros módulos accedan directamente al objeto de configuración (lo cual podría ser propenso a errores), llaman a esta función. Le indican la `seccion` (ej. 'Facial') y la `clave` (ej. 'indice_camara') que quieren leer. Además, pueden especificar el `tipo` de dato que esperan (`int`, `float`, `bool`) y un valor por `defecto`. La función se encarga de usar el método correcto de `configparser` (como `getint`, `getfloat`, `getboolean`), manejar cualquier error si la clave no existe o el valor es incorrecto, y devolver el valor por defecto en caso de fallo. Esto previene que la aplicación se caiga por un error en el archivo de configuración y hace que el código en otros módulos sea mucho más limpio.

Finalmente, el bloque `if __name__ == '__main__':` demuestra cómo usar el módulo. Carga la configuración y luego usa `get_config_valor()` para leer y mostrar varios parámetros. Sirve como una excelente herramienta de prueba para verificar que el módulo puede leer y convertir correctamente los tipos de datos del archivo `config.ini`.

En resumen, este módulo reemplaza eficazmente a `constants.py`, moviendo los parámetros fijos del código a un archivo de configuración externo, lo que representa un gran paso adelante en la madurez y mantenibilidad del proyecto.