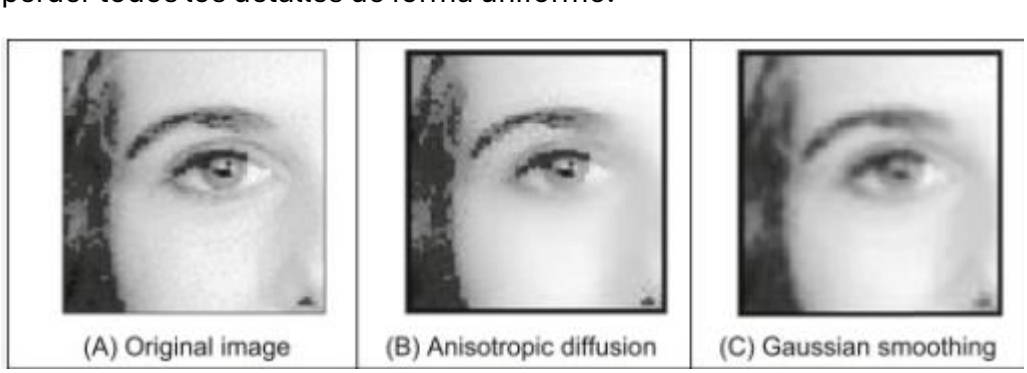
Visión por computadora – Lab 4

Investigación

El filtro anisotrópico es una técnica para filtrar texturas, procurando mejorar la calidad de las imágenes en superficies que se ven ángulos oblicuos. A diferencia de otros tipos de filtro, que solo consideran una única dirección, el anisotrópico considera varias muestras sobre una dirección de mayor detalle, procurando mantener la nitidez de texturas.

La forma en la que funciona es que toma muestras sobre el eje que posea una mayor textura estirada, como puede ser un camino, de forma que mantiene detalles nítidos. Así mismo, esto a diferencia de un filtro gaussiano que busca suavizar los detalles de la imagen difuminando detalles de una forma general, mientras que uno anisotrópico busca difuminar mientras que mantiene las principales características, suavizando sin perder todos los detalles de forma uniforme.



Solución

Se desarrolló una U-Net para poder convertir imágenes a un filtrado de Perona-Malik. Para ello se utilizaría la base de datos de Berkely. Así mismo, la arquitectura de la U-Net estaría compuesta por una profundida inicial de 3 capas, con 3 convoluciones, un dropout y realizando 3 ascensos con funciones de activación RELU.

Así mismo, la metodología estaría compuesta por los siguientes pasos:

a. Carga de las imágenes
b. Aplicación del filtro anisotrópico
c. Generación de ventanas aleatorias de un tamaño de 32x32 sobre las imágenes filtradas y las imágenes normales de forma que fueran nuestro entrenamiento y nuestro objetivo.
d. Creación de la arquitectura de la U-Net
e. Entrenamiento con una distribución de los datos de la siguiente forma: validación del 15%, 15% de test y 70% de entrenamiento.
f. Prueba con imágenes

Discusión

Se observó que realmente la implementación del filtro anisotrópico no funciona correctamente con esta máquina ya que realmente se parece mucho a un Gaussiano, o es probable que se tuviera que tunear para poder generar una imagen que no pareciera un filtro Gaussiano. Asimismo, se destaca el hecho de que debido a la limitada capacidad de la computadora se debió de trabajar con imágenes de 32x32 y no con 64 o 128 ya que ocupaban demasiada memoria RAM. También, se destaca el hecho de que se tuvieron que generar únicamente 1000 ventanas aleatorias y no las 2500 por imagen ya que no soportaba, se utilizó una RAM de 16GB, con un procesador de AMD Ryzen 7 6800H y una tarjeta gráfica RTX 3500Ti.

Para al arquitectura se tuvieron que utilizar 5 capas de convolución y upsampling ya que no se daba a basto ni con 3 ni con 4, teniendo un error absoluto medio de 44% y una pérdida del 22%. Por ello es que se aumentó a una capa más, por más que se tunearan los hiperparámetros.

Dentro del entrenamiento se trabajaron con batches de 8 dada la limitada capacidad de la RAM y también utilizando un learning rate mínimo de 10^-6 aunque iniciaba en 10^-4. Aproximadamente se tardó 10 minutos en entrenarse obteniendo unos resultados más que satisfactorios ya que se obtuvo una pérdida de menos del 0.0002% y dentro de la validación igual obteniendo un error absoluto medio menor al 0.01% tanto en la validación como en el entrenamiento.

Finalmente, al momento de evaluar las imágenes y las predicciones se observa que realmente no existen diferencias significativas entre las imágenes predichas contra las imágenes con el filtrado original. En general se obtuvo un desempeño excelente y como recomendación se podría desarrollar tal vez una que utilice una mayor cantidad de batches, y si es posible, optimizar la U-Net para ver si se puede mejorar el desempeño con una menor cantidad de capas de convolución.

Repositorio

https://github.com/JusSolo/lab-4-vision.git