Universidad de los Andes > FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Tarea 2

Vision Artificial 2023 10

Alumno:

Raúl Duhalde Errázuriz

Professor:

Jose Delpiano

Deadline:

Jueves 23 de Marzo 2023 at 23:59

Índice

1. Introducción

En esta tarea, se busca desarrollar un efecto de video como el que se muestra en este enlace http://vimeo.com/7878518. El efecto consiste en un cambio de variable en el tiempo del tipo $t' = t - c \cdot x$, donde c es una constante. Es decir, las líneas tienen un retardo según la posición en el eje vertical. Las de más abajo se muestran más tarde.

En el desarrollo de la tarea, se realizará un análisis y diseño previo para explicar las decisiones de diseño y las herramientas utilizadas. Además, se detallarán los requisitos necesarios para el desarrollo del efecto.

Luego, se explicará el proceso de desarrollo del efecto, desde la lectura del video de entrada hasta la escritura del video de salida.

Finalmente, se presentarán las conclusiones de la tarea y se mostrarán las referencias utilizadas para su desarrollo.

2. Diseño y análisis

2.1 Decisiones de diseño:

Se han tomado las siguientes decisiones de diseño:
La resolución de la imagen del vídeo original se mantiene en el video de salida.
Se segmenta el video en pequeñas secciones y se aplican los cambios de variable de tiempo en cada sección.
Se escala la resolución del video a una tasa específica antes de aplicar el efecto.
La salida es un archivo mp4, ya que es un formato de video ampliamente utilizado.

2.2 Uso de herramientas:

Se utilizó el lenguaje de programación Python, junto con la biblioteca de OpenCV para procesar el video. Además, se utilizó el módulo time para medir el tiempo de ejecución del programa.

3. Requisitos

Para el desarrollo de este efecto de video se necesitarán los siguientes requisitos: Lenguaje de programación: Python 3.6 o superior.

Librería OpenCV: es necesario tener instalada la librería OpenCV de Python para la lectura y escritura de videos.

Numpy: se utilizará la librería Numpy de Python para la manipulación de matrices y operaciones matemáticas.

Una cámara o un video de entrada: para aplicar el efecto, es necesario tener un video o una cámara conectada para capturar imágenes en tiempo real.

4. Desarrollo

Lectura del video de entrada: se utiliza la función "VideoCapture" de OpenCV para leer el video de entrada y almacenar cada cuadro en una lista de cuadros.

Segmentación del video: se divide el vídeo en segmentos de un número determinado de cuadros para aplicar el efecto de manera más precisa.

Procesamiento de cada segmento: para cada segmento, se recorre cada cuadro y se aplica la transformación $t' = t - c \cdot x$ para cada línea de píxeles en el cuadro, donde t es el tiempo original, c es la constante definida anteriormente y x es la posición vertical de la línea de píxeles en el cuadro.

Escritura del video de salida: se utiliza la función "VideoWriter" de OpenCV para escribir el vídeo de salida con el efecto aplicado.

Además, durante el proceso de desarrollo, se utilizó la herramienta Chat GPT para reducir y simplificar el código y mejorar los comentarios del mismo. Se introdujo el prompt inicial de "Reducir código y agregar comentarios" y se fueron pidiendo cambios y correcciones a la herramienta. De esta manera, se logró una mayor claridad y legibilidad en el código, sin que parezca que se ha copiado directamente de la herramienta.

En resumen, el uso de herramientas y la aplicación de la lógica de segmentación permitió desarrollar un efecto de video interesante y eficiente. Además, la utilización de Chat GPT para reducir y simplificar el código y mejorar los comentarios del mismo fue una herramienta valiosa en el proceso de desarrollo.

5. Conclusión

En conclusión, el desarrollo de este efecto de video fue posible gracias a la utilización de las herramientas y librerías adecuadas, como Python y OpenCV. La lógica de segmentación permitió aplicar el efecto de manera más precisa y eficiente. Además, la constante c se puede ajustar para obtener diferentes variaciones del efecto.

6. Links

link de mi video original:
https://drive.google.com/drive/folders/1Oj9dRer3Y3TGrG-cN1GG7OJI5xtOHYuF?usp=sharing

link de mi video del después:
https://drive.google.com/drive/folders/1r2bhUNGBmQ2MMTe7l08AEB1LMFEq0vmm?usp=sharing

video original del efecto: http://vimeo.com/7878518

openCV: https://opencv.org/

7. Código

```python
#Raul Duhalde Errazuriz
#Tarea 2
import cv2
import numpy as np
from time import perf_counter

#abrir el archivo de video
cap = cv2.VideoCapture('visionartificial1.mp4')

#obtener los cuadros por segundo y el número total de cuadros
fps = cap.get(cv2.CAP_PROP_FPS)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

#definir la constante para el filtro de la variable de tiempo
c = 0.05

#definir el códec y crear un objeto VideoWriter
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter('filtered_visionartificial1.mp4', fourcc, fps, (int(cap.get(3)), int(cap.get(4))))

#definir parámetros de segmentación
segment_count = fps * 3
scale_fact = 1
segment_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT) * scale_fact / segment_count)
frames = []

#comenzar a procesar cada cuadro
t1 = perf_counter()
while(cap.isOpened()):
    ret, new_frame = cap.read()

    if new_frame is None:
        break
```

```python
#escalar el cuadro si es necesario
    if scale_fact != 1:
        new_frame = cv2.resize(new_frame, (int(new_frame.shape[1]*scale_fact),int(new_frame.shape[0]*scale_fact)))

#  agregar el cuadro a la lista de cuadros
    frames.append(new_frame)

# aplicar la lógica de segmentación y concatenación una vez que tenemos suficientecuadros
    if len(frames) >= segment_count:
        segments = []
        for i, frame in enumerate(frames):
            segments.append(frame[i*segment_height:(i+1)*segment_height])

        after_frame = np.concatenate(segments, axis=0)

# Aplicar el filtro de la variable de tiempo
        t = i / fps
        x = after_frame.shape[0] - 1
        t_new = t - c * x
# cambio de variable en el tiempo del tipo t' = t - c · x donde t es el tiempo original,
# c es la constante del filtro y x es la posición en la dirección vertical del cuadro segmentado.
# matriz de transformación para aplicar el cambio de variable en el tiempo
        M = np.float32([[1, 0, 0], [0, 1, 0]])
        M[0, 2] = -c * t_new * fps

# aplicar la transformación a través de la función warpAffine de OpenCV
        filtered_frame = cv2.warpAffine(after_frame, M, (after_frame.shape[1], after_frame.shape[0]))

#escribir el cuadro procesado en el archivo de video de salida
        out.write(filtered_frame)
# mostrar el cuadro procesado
        cv2.imshow('frame', filtered_frame)

        # Esperar a que se presione la tecla 'q' para salir
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

```python

#actualizar el tiempo y eliminar el cuadro más antiguo
        t1 = perf_counter()
        frames.pop(0)

cap.release()
out.release()
cv2.destroyAllWindows()
```