Reconstrucción de Videos y Filtro de Cámara Lenta mediante Interpolación Polinomial

Silvio Vileriño *(svilerino@gmail.com)*, Sacha Kantor *(sacha.kantor+exactas@gmail.com)*
Juan Grandoso *(juan.grandoso@gmail.com)* y Nahuel Lascano *(laski.nahuel@gmail.com)*

Resumen

La elaboración de órdenes es uno de los problemas más frecuentes que se da en nuestra sociedad. Virtualmente todos los ámbitos conocidos (deportes, laboral, lúdico, salarial, etc) enfrentan este problema, que será más o menos complejo según el contexto. A lo largo de este trabajo estudiamos este problema en el ámbito de las páginas web: ¿cuándo una página debería estar por encima de otra?

Palabras Clave

PageRank, Autovectores, Matriz Estocástica, Motores de Búsqueda

Índice

1. Introducción Teórica

En el siguiente trabajo se aborda el desafío de aumentar algorítmicamente la cantidad de frames de un video de manera que el resultado se asemeje al video original. En otras palabras, el objetivo es, a partir de un video, generar otro con mayor cantidad de frames, de modo tal que coincidan en aquellos frames presentes en el video

original y los frames generados se *ajusten* a estos, apuntando idealmente a que el ojo humano no perciba el agregado artificial.

En esta introducción presentaremos una lista no exhaustiva de las diversas situaciones que motivan la resolución de dicho problema y daremos un marco teórico a los métodos propuestos para su solución.

1.1. La motivación

1.1.1. Compresión de video

El crecimiento exponencial de internet ha dado lugar, entre otras cosas[**TP2**], a la mejora de la infraestructura utilizada, lo que en particular repercutió en un aumento generalizado de las velocidades de conexión y del ancho de banda de las mismas. Esto promovió su utilización para compartir contenido cada vez más pesado, en particular videos de todo tipo, desde caseros a profesionales. Además, de la mano con el avance de las tecnologías de captura de video, la resolución de los mismos aumenta cada vez más.

Sin embargo, la inmensa cantidad de usuarios impone un límite al ancho de banda que se le puede dedicar a cada uno, sobretodo para sitios populares como YouTube que sirven miles de videos en cada instante determinado, e impone la necesidad de criterios para reducir la cantidad de paquetes que se le transfieren a cada usuario.

Un abordaje común y ampliamente difundido es la compresión de videos, con o sin pérdida de calidad. En términos generales, consiste en que el servidor envíe una versión comprimida del video (posiblemente precomputada de antemano) y que el usuario use su propio poder de cómputo para descomprimirlo y visualizarlo. De este modo se reduce la cantidad de tráfico en la red a costa de un trabajo mayor de CPU de servidores y usuarios, que en general resulta menos costoso.

Los resultados de este trabajo pueden utilizarse como método de compresión con pérdida. Visto de ese modo, una versión *comprimida* de un video es un nuevo video con un subconjunto de los frames del original. El mecanismo de compresión, entonces, resulta muy sencillo de implementar. Para realizar la descompresión se precisa, entonces, generar los frames faltantes a partir de los recibidos. El objetivo de este trabajo es el estudio de distintos métodos para resolver ese problema.

Pero la compresión de videos no es la motivación principal de los métodos estudiados. Para dicho problema existen variados algoritmos que, sin eliminar cuadros completos, representan cada uno en función de los cambios respecto de los anteriores, obteniendo resultados más fieles (dado que no eliminan por completo la información de ningún cuadro) sin un tamaño mucho mayor[**wiki˙data˙compression˙video**].

1.1.2. Reproducción en cámara lenta

Otra motivación posible y más generalizada resulta de analizar las tecnologías de captura de video. Desde su invención, el principio básico se mantuvo intacto: capturar varias imágenes por segundo y reproducirlas en orden para dar al ojo humano la sensación de movimiento. Un video, entonces, no es más que una secuencia de imagenes (en adelante *frames*) reproducidas a una frecuencia determinada, en general mayor a 12 por segundo (el máximo que el sistema visual humano puede percibir como imágenes separadas[**wiki˙framerate**]). En la época del cine mudo las películas se filmaban con cámaras manuales, lo cual permitía alterar la cantidad de frames por segundo (en adelante *frame-rate*) según la velocidad que se le quisiera dar a la escena: a mayor frame-rate la escena se percibe más lenta y viceversa. Pero al añadirles sonido fue necesario estandarizar el frame-rate, pues el oído humano es mucho más sensible a cambios de frecuencia que el ojo[**wiki˙framerate**]. Desde entonces el estándar ha sido filmar y reproducir a (aproximadamente) 24 cuadros por segundo, tanto películas como demás videos, lo cual se mantuvo prácticamente intacto hasta 2012 con la llegada del Cine en Alta Frecuencia (*HFR* por sus siglas en inglés) de la mano de Peter Jackson en *The Hobbit: An Unexpected Journey*.

A lo largo de la historia y cada vez con mayor frecuencia se han utilizado Cámaras de Alta Velocidad (*HSC*) para generar videos que, reproducidos a 24 *fps*[a] permitan percibir cosas que una cámara normal e incluso el ojo humano no percibiría. Los usos de los mismos son muy variados, y van desde la biomecánica[b] hasta los eventos deportivos[c], pasando incluso por meras curiosidades[d]. Sin embargo los videos resultantes son sumamente pesados por unidad de tiempo, haciéndolos complicados de almacenar, transportar y distribuir. Además, el equipo necesario

a. frames por segundo, unidad estándar del frame-rate
b. https://www.youtube.com/watch?v=VSzpM8vEAFA
c. https://www.youtube.com/watch?v=O0lCJfFtjCQ
d. https://www.youtube.com/watch?v=tw3q4_jZv8M

para realizar las capturas suele ser mucho más costoso que el equipamiento normal. O quizás simplemente no se cuenta con una versión en alta velocidad de un video ya filmado.

Dicho en líneas más generales, es posible que se desee reproducir en cámara lenta un video del cual, por el motivo que fuere, solo se tiene una versión con frame-rate estándar. Una solución es generar computacionalmente los frames faltantes, aprovechando la información existente para crear frames que se acerquen lo más posible a los que hubiese producido una HFC. Lo cual nos lleva nuevamente al objeto de estudio de este trabajo.

Esta es, en particular, la motivación sobre la que más hincapié haremos en el resto del trabajo, más allá de que los mismos métodos pueden ser usados para atacar cualquiera de los problemas que describimos en esta sección.

1.1.3. Suavización de video y Morphing

También es posible que lo que se busque sea generar frames nuevos a partir de un video ya existente pero no con la intención de verlo en cámara lenta sino para que el resultado final sea más "suave" o agradable a la vista. Es un proceso común en los videos animados dibujados a mano[a], dado que el trabajo extra necesario para dibujar cada frame individualmente difícilmente sea apreciado por el espectador final. Hoy en día se utiliza también en animaciones por computadora, por ejemplo para realizar una trancisión fluida entre dos expresiones de una cara o entre dos estados posibles de un cuerpo 3D.

Un objetivo similar es generar una trancisión entre dos fotos o videos que no necesariamente forman parte de umna misma captura, pero que se quiere integrar en un único y, en lo posible, fluido video. Los usos más comunes del *morphing*, como se le llama, consisten en transformar la cara de una persona en la de otra[b].

1.2. Marco teórico

Nos interesa, entonces, transformar videos computacionalmente para que se perciban más lentamente, que el resultado final se vea "fluido" y se acerque lo más posible a lo que se habría capturado usando una Cámara de Alta Velocidad. Lo primero que hace falta es modelar los videos de un modo que nos permita manejarlos usando lenguajes de programación conocidos, sin incluir herramientas de edición complejas que exceden al alcance de este trabajo. Usamos entonces que un video, en su forma "original" (sin compresión) es un conjunto ordenado de imágenes, cada una de la cual representa un frame. En consecuencia, el problema consiste en generar nuevas imágenes "intermedias" para que, al reproducir el video con el mismo frame-rate, se perciba más lento. A esto último lo conoceremos como el efecto de *slowmotion*.

Cada imagen, a su vez, se puede modelar como una matriz de píxeles. Para simplificar el análisis, consideraremos todas las imágenes (y por lo tanto los videos) únicamente en escala de grises[c]. De este modo, un píxel se puede representar con entero entre 0 y 255 inclusive (un byte) que denota la cantidad de luz que hay en ese píxel particular (siendo 0 el negro absoluto y 255 el blanco absoluto).

La generación de estas nuevas imágenes intermedias puede hacerse de diversas maneras. Una de ellas, aplicada en este trabajo, implica generarlas píxel por píxel, utilizando la información que nos brindan los píxeles correspondientes de las imágenes cercanas en el tiempo. Más formalmente, para cada píxel p de cada frame f a generar tomamos como información los píxeles que están en la misma posición que p en las imágenes cercanas a f en el tiempo.

Se nos presenta el problema de cómo utilizar esa información para generar un píxel que resulte en un video final con las propiedades deseadas. En este trabajo estudiamos diferentes métodos y los comparamos estableciendo métricas cualitativas y cuantitativas. Pero para introducir el detalle de los métodos hace falta hacer algunas definiciones previas.

1.2.1. Interpolación

Dado un conjunto de puntos en $\mathbb{R}^2 (x_0, y_0), \ldots, (x_n, y_n)$ con $x_i \neq x_j$ si $i \neq j$, decimos que una función $f : \mathbb{R}^2 \to \mathbb{R}^2$ interpola dichos puntos si $f(x_i) = y_i$ para todo $i = 0, \ldots, n$. En particular, si nos restringimos a considerar polinomios, se puede demostrar que dados $n + 1$ puntos como los descritos existe un único polinomio $P \in \mathbb{R}[x]$ de grado menor o igual que n tal que los interpola[[**wiki˙lagrange˙polynomial**]]. A este polinomio se lo conoce como *Polinomio Interpolador de Lagrange*, y su fórmula esta dada de la siguiente manera:

a. De hecho el primer video animado de la historia, Fantasmagorie de 1908, tiene solo la mitad de sus cuadros realmente dibujados.
b. Como se puede ver en este excelente ejemplo: https://www.youtube.com/watch?v=3ZHtL7CirJA
c. Numerical representation: https://en.wikipedia.org/wiki/Grayscale

$$P(x) = \sum_{k=0}^{n} \left(y_k \prod_{i \neq k}^{n} \frac{x - x_i}{x_k - x_i} \right)$$

Sin embargo, los polinomios interpolantes tienen la desventaja de que cuando el n es grande *oscilan* demasiado. Es por esto que en general se utiliza la técnica de *Interpolación segmentada*, que consiste en desarrollar la función interpolante f de a partes tomando subconjuntos de puntos consecutivos, generando el poliniomio interpolante de cada subconjunto y luego "conectando" cada uno de estos en el orden adecuado.

Existen diversos tipos de Interpolación segmentada, entre ellos la lineal, cuadrática y cúbica, nombrados según el grado de cada polinomio interpolante. Para la aplicación del efecto de *slowmotion*, se utilizará la lineal y cúbica. A continuación se dará una breve explicación de estos dos métodos de interpolación.

1.2.2. Lineal

El caso más sencillo es la **Interpolación segmentada lineal**: construimos $S_0, \ldots, S_{n-1}$ polinomios tal que S_i interpola x_i y x_{i+1} y definimos

$$f(x) = \begin{cases} S_0(x) & \text{si } x \in [x_0, x_1] \\ \vdots & \\ S_{n-1}(x) & \text{si } x \in [x_{n-1}, x_n] \end{cases}$$

como la función interpolante final. Notar que, por ser cada S_i polinomio interpolante de dos puntos, cada S_i es de grado a lo sumo 1, con lo cual es simplemente una recta.

La interpolación segmetada lineal, no obstante, posee el problema de no resultar "suave" geométricamente, es decir, no es derivable en los puntos considerados. Esto, sumado a problemas diferentes que trae la utilización de polinomios cuadráticos, nos motiva a usar polinomios cúbicos, dando lugar a la técnica conocida como *splines*.

1.2.3. Splines (Interpolación cúbica)

Al igual que en la Interpolación lineal, se procede a construir $S_0, \ldots, S_{n-1}$ polinomios donde cada uno de ellos interpola un segmento equiespaciado sobre x_i y x_{i+1} ($i = 0, \ldots, n$). La diferencia es que dichos polinomios son de grado 3. Por conveniencia, vamos a considerar que son de la forma

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Se desea obtener la tupla de coeficientes (a_i, b_i, c_i, d_i) para $i = 0, \ldots, n$. Para ello se utiliza una serie de condiciones que, de cumplirse, mejorarían el problema descrito de la interpolación lineal. Enumeramos estos requisitos, primero, desde una perspectiva más conceptual y no tan formal:

1. Los polinomios S_i deben pasar por los conjuntos de puntos (x_i, y_i) que se establecieron como datos iniciales. Es decir, asegurar que la función resultante sea continua e interpole a los puntos originales.
2. Se debe mantener también la continuidad en las derivadas primeras y segundas. En otras palabras, por cada par S_i, S_{i+1} con $i = 0, \ldots, n-2$ se debe cumplir la igualdad de la derivada primera y segunda de dichos polinomios en el punto en que se conectan (x_{i+1}).
3. Para la última restricción se consideran dos opciones:
   a) La derivada segunda en los bordes de la función debe ser nula cuando se evalua en el x_0 y x_n, respectivamente. Se la define como *Spline natural*.
   b) La derivada primera en los bordes coinciden con el de la función original que se está aproximando en el punto x_0 y $x + n$, respectivamete. Esto implica que se necesita de cierta información extra sobre la función a interpolar. Por esta razón, se la conoce como *Spline sujeto a la función interpolada*.

Formalmente, todo esto se expresa como[a]

a. El siguiente desarrollo formal fue obtenido y adaptado del **Apunte de Métodos Numéricos** de **Guido Tagliavini Ponce** accesible en https://github.com/CubaWiki/MetNum-ApunteFinal-gtagliavini/raw/master/notas.pdf

1. $S_i(x_i) = y_i$ para todo $i = 0, \ldots, n-1$, y $S_{n-1}(x_n) = y_n$
2. $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$ para todo $i = 0, \ldots, n-2$
3. $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ para todo $i = 0, \ldots, n-2$
4. $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$ para todo $i = 0, \ldots, n-2$
5. Se cumple que
   a) $S''(x_0) = S''(x_n) = 0$ (spline natural)
   b) $S'(x_0) = f'(x_0)$ y $S'(x_n) = f'(x_n)$ (spline sujeto)

Y definimos entonces al *spline* como el conjunto de todas las funciones cúbicas S_i:

$$f(x) = \begin{cases} S_0(x) & \text{si } x \in [x_0, x_1] \\ \vdots & \\ S_{n-1}(x) & \text{si } x \in [x_{n-1}, x_n] \end{cases}$$

A pesar de su mejora en ciertos aspectos, como la suavidad del *spline*, la cantidad de cálculos a realizar es mayor, lo cual nos lleva a un problema en el ámbito computacional. Estudiemos entonces cómo encontrar los coeficientes.

Recordemos que estamos considerando polinomios de la forma

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Observemos que $S_i(x_i) = a_i$. Usando la condición 1, tenemos

$$a_i = y_i \text{ para todo } i = 0, \cdots, n-1$$
$$a_{n-1} + b_{n-1}(x_n - x_{n-1}) + c_{n-1}(x_n - x_{n-1})^2 + d_{n-1}(x_n - x_{n-1})^3 = y_n$$

Combinando las condiciones 1 y 2 obtenemos $S_i(x_{i+1}) = y_{i+1} = a_{i+1}$. Entonces

$$a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 = a_{i+1} \text{ para todo } i = 0, \cdots, n-2$$

Observar que $S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$. Como $S_{i+1}'(x_{i+1}) = b_{i+1}$ entonces la condición 3 equivale a

$$b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2 = b_{i+1} \text{ para todo } i = 0, \cdots, n-2$$

Observar que $S_i''(x) = 2c_i + 6d_i(x - x_i)$. Como $S_{i+1}''(x_{i+1}) = 2c_{i+1}$ entonces la condición 4 equivale a

$$2c_i + 6d_i(x_{i+1} - x_i) = 2c_{i+1} \text{ para todo } i = 0, \cdots, n-2$$

Finalmente, si el spline es natural (la versión que utilizaremos en este trabajo, por desconocer la derivada de la función que genera los puntos), la condición 5 equivale a

$$2c_0 = 0$$
$$2c_{n-1} + 6d_{n-1}(x_n - x_{n-1}) = 0$$

Definamos $a_n = y_n$, $c_n = 0$ y $h_i = x_{i+1} - x_i$. Entonces las ecuaciones son

1. $a_i = y_i$ para todo $i = 0, \ldots, n$
2. $a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = a_{i+1}$ para todo $i = 0, \ldots, n-1$
3. $b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1}$ para todo $i = 0, \ldots, n-2$
4. $2c_i + 6d_i h_i = 2c_{i+1}$ para todo $i = 0, \ldots, n-1$
5. $c_0 = 0$

De la ecuación 4 despejamos

$$d_i = \frac{c_{i+1} - c_i}{3h_i}$$

Sustituyendo esto último y la ecuación 1 en la ecuación 2 y despejando

$$\begin{aligned} b_i &= \frac{1}{h_i}\left(y_{i+1} - y_i - c_i h_i^2 - d_i h_i^3)\right) \\ &= \frac{1}{h_i}\left(y_{i+1} - y_i - c_i h_i^2 - \frac{(c_{i+1} - c_i)h_i^3}{3h_i}\right) \\ &= \frac{y_{i+1} - y_i}{h_i} - c_i h_i - \frac{1}{3}(c_{i+1} - c_i)h_i \\ &= \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}\left(2c_i + c_{i+1}\right) \end{aligned}$$

Sustituyendo en la ecuación 3 y despejando

$$\begin{aligned} 0 &= b_i - b_{i+1} + 2c_i h_i + 3d_i h_i^2 \\ &= \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}\left(2c_i + c_{i+1}\right) - \frac{y_{i+2} - y_{i+1}}{h_{i+1}} + \frac{h_{i+1}}{3}\left(2c_{i+1} + c_{i+2}\right) \\ &+ 2c_i h_i + 3\frac{c_{i+1} - c_i}{3h_i}h_i^2 \\ &= \left[\frac{y_{i+1} - y_i}{h_i} - \frac{y_{i+2} - y_{i+1}}{h_{i+1}}\right] - \frac{2}{3}h_i c_i - \frac{1}{3}h_i c_{i+1} + \frac{2}{3}h_{i+1}c_{i+1} + \frac{1}{3}h_{i+1}c_{i+2} \\ &+ 2h_i c_i + h_i c_{i+1} - h_i c_i \\ &= \left[\frac{y_{i+1} - y_i}{h_i} - \frac{y_{i+2} - y_{i+1}}{h_{i+1}}\right] + \frac{1}{3}h_i c_i + \frac{2}{3}(h_i + h_{i+1})c_{i+1} + \frac{1}{3}h_{i+1}c_{i+2} \end{aligned}$$

Equivalentemente

$$h_i c_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = 3\left[\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i}\right]$$

Esta ecuación, que vale para $i = 0, \cdots, n-2$, contiene toda la información de las demás (pues la obtuvimos a través de sustituciones sucesivas). Juntando estas $n-1$ ecuaciones con $c_0 = 0$ y $c_n = 0$ tenemos un sistema de $n+1$ ecuaciones y $n+1$ incógnitas

$$\begin{pmatrix} 1 & 0 & 0 & 0 & & & \\ h_0 & 2(h_0+h_1) & h_1 & 0 & & & \\ 0 & h_1 & 2(h_1+h_2) & h_2 & & & \\ & & & \ddots & & & \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & & 0 & 0 & 1 \end{pmatrix}$$

Esta matriz es estrictamente diagonal dominante (pues $h_i = x_{i+1} - x_i > 0$), por lo cual es inversible[**properties˙of˙diagonally˙d**]. Luego, la solución es única: existe un único spline natural para $x_0, \cdots, x_n$, la cual se puede encontrar utilizando métodos de resolución de sistemas de ecuaciones como eliminación gaussiana[**TP1**].

1.3. Error

Para establecer la correctitud de nuestros métodos, hará falta utilizar la noción de *Error Cuadrático Medio* (ECM) y *Peak to Signal Noise Ratio* (PSNR). Básicamente, dado un video de $m \times n$ (alto $\times$ ancho) con frames originales F

y uno con la misma resolución con frames generados artificialmente $\bar{F}$, definimos

$$\texttt{ECM}(F, \bar{F}) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} |F_{k_{ij}} - \bar{F}_{k_{ij}}|^2$$

$$\texttt{PSNR}(F, \bar{F}) = 10 \log_{10} \left(\frac{255^2}{\texttt{ECM}(F, \bar{F})} \right)$$

Como forma de intuición, notemos que el ECM, como su nombre lo indica, es un promedio de la diferencia de todos los píxeles de dos frames al cuadrado. Asimismo, el PSNR es el una medida de cuánto se acerca el error al máximo posible (255^2) medido en escala logarítmica.

2. DESARROLLO

Finalizada la enumeración de los problemas que motivan la investigación sobre el tema y la adición de una breve explicación del problema a trabajar y de la justificación teórica detrás del método de interpolación, nos dedicaremos a desarrollar con mayor precisión cada método número en cuestión. Recordemos que estos métodos intentan atacar el problema de construir, a partir de un video filmado en tiempo real, nuevos frames que den la sensación de que el video original ha sido ralentizado (efecto de *slowmotion*), o, aún mejor, filmado con una cámara de alta frecuencia.

2.1. Aclaraciones útiles para la comprensión del desarrollo

Antes de comenzar con la explicación de los métodos queremos definir ciertas frases o palabras claves que ocuparán terreno en el resto de la sección:

- *Frame original*: Cuadro que pertenece al video original, es decir, aquel que no contiene los frames agregados que conciben al efecto de cámara lenta.
- *Frame artificial*: Cuadro que se realizará a partir del procedimiento que cada método vaya planeando. Siempre tiene a cierta distancia, tanto a derecha como izquierda, frames originales.
- *Cantidad de frames a adjuntar*: Entero que indica cuántos cuadros artificiales le adicionaremos entre cada par de frames originales. Lo denotaremos con las letras fr.
- *Matriz frame*: Aunque no se hablará de este nombre en particular, es fundamental aclarar que en cualquier comentario acerca de los valores o píxeles de un frame, se lo está modelando como una matriz de $m \times n$ (resolución del video) que lleva los valores del 0 al 255 inclusive (escala de grises).
- *Píxel*: Transladamos esta palabra asociada a la imagen digital al campo del álgebra lineal. Por lo tanto, se lo considerará como un componente de la matriz frame cuyos valores estan restringido en el rango $[0; 255]$.

Para la explicación de todos los métodos consideramos dado (como parámetros del problema) el video sobre el cual trabajar (con su resolución $m \times n$) y la cantidad fr de frames a generar entre cada par de frames del original.

Además, dado $t \in \mathbb{R}$ definiremos lo siguiente para ayudarnos en las definiciones más formales:

$$
\begin{aligned}
& original(t) \text{ sii } t \text{ coincide con el instante de tiempo de un frame original} \\
original(t) \Rightarrow\ & frame(t) = \text{el frame original del instante } t \\
\neg original(t) \Rightarrow\ & anterior(t) = \max_k(original(k) \wedge k < t) \\
\neg original(t) \Rightarrow\ & siguiente(t) = \min_k(original(k) \wedge k > t) \\
\neg original(t) \Rightarrow\ & resto(t) = t - anterior(t) \\
0 \leq i < m \wedge 0 \leq j < n \Rightarrow\ & f[i, j] = \text{el valor de brillo de píxel en la posición } (i, j) \text{ para un frame } f
\end{aligned}
$$

2.2. Los métodos propuestos

2.2.1. Cuadro más cercano

Este método se basa en una idea simplista, muy sencilla de implementar, y que nos permitirá tener una base para establecer comparaciones con métodos más "inteligentes". A pesar de esto, puede resultar precisa para ciertos tipos de videos, como por ejemplo, la filmación de un objeto inmóvil.

Intuición

Como lo indica el subtítulo de la sección, la idea del método es que cada frame artificial es una copia de su frame original más cercano en términos temporales. Para cada frame a generar el método calcula cuál es dicho "vecino más cercano" y luego copia la imágen de dicho frame al frame artificial a generar.

De esta manera, habrá al menos $fr/2$ copias de cada frame original. En caso que se decida agregar una cantidad impar, se opta por fragmentar en dos partes de $fr/2$ y $fr/2 + 1$ cuadros. En la figura 1 vemos un ejemplo conciso de lo explicado. La cantidad de cuadros a agregar es de $fr = 22$, por lo que se lo divide en dos particiones de 11 frames artificiales cuya imagen corresponderá al frame original más cercano (el marrón o blanco).

Volviendo al análisis del video en su totalidad, se repite el anterior procedimiento para cada par de frames en el orden establecido por la secuencia del video. De tal forma se obtienen los frames artificiales, consiguiendo una nueva filmación con el efecto de *slowmotion* que queríamos.

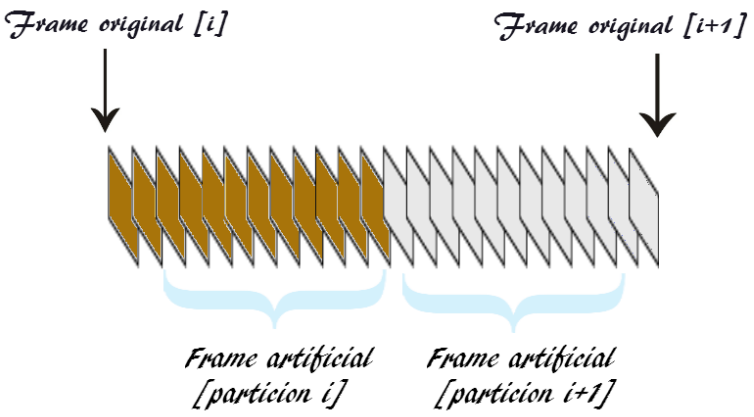

Figura 1. Ejemplo de vecino más cercano

Perspectiva matemática

¿Qué relación tiene este método con el concepto de interpolar? A pesar de ser muy sencillo, este método está usando cierta forma de interpolación. Si consideramos cada píxel (i, j) del video a lo largo del tiempo, obtenemos para cada uno una lista de pares $f_{i,j}(t) = y$ donde t es el instante de un frame original y y el valor de brillo del píxel (i, j) de dicho frame. El método propuesto interpola dichos puntos usando

$$f_{i,j}(t) = \begin{cases} frame(t)_{i,j} & \text{si } original(t) \\ frame(anterior(t))_{i,j} & \text{si } resto(t) \leq fr/2 \\ frame(siguiente(t))_{i,j} & \text{si } resto(t) > fr/2 \end{cases}$$

Es fácil ver que la función interpola efectivamente los puntos $f_{i,j}(t) = y$, aunque lo hace de modo "pobre": para cada par de frames originales tiene una discontinuidad entre ellos a distancia $fr/2$, que será percibida como un "salto" en las imágenes del video. El único caso en que esto no ocurre es cuando los frames considerados son idénticos, por lo que arriesgamos que es el caso en que el método mejor se comportará.

2.2.2. Interpolación lineal

Compartiendo con *vecino más cercano* la idea de trabajar píxel a píxel con cada par de frames para eventualmente obtener el video deseado con su respectivo efecto, este método puede propocionar ciertas mejoras a la hora de trabajar con videos con movimiento[a].

Intuición

El método se centra en tomar crear un interpolador lineal segmentado por cada píxel del video, cuyos puntos interpolados son los valores de brillo del píxel en los frames originales. Por ende, tendremos por cada par de frames originales $m \times n$ polinomios de grado uno, que usaremos para conocer los valores intermedios entre los dos cuadros originales y gererar así los píxeles de los frames artificiales.

En la figura 2 se puede observar el efecto producido por una interpolación de este estilo.

Perspectiva matemática

Nuevamente consideramos para cada píxel una lista de pares $f(t) = y$. Aplicando lo visto en la Introducción teórica, para cada píxel construimos $S_0, \ldots, S_{n-1}$ polinomios de grado a lo sumo 1 (con n cantidad total de frames del video) tal que S_p interpola $t_p = y_p$ y $t_{p+1} = y_{p+1}$ para cada t_p instante de un frame original. Luego, definimos la siguiente función:

a. Que, entendemos, es el caso de la mayoría de los videos, exceptuando quizás los subidos a YouTube para compartir música.

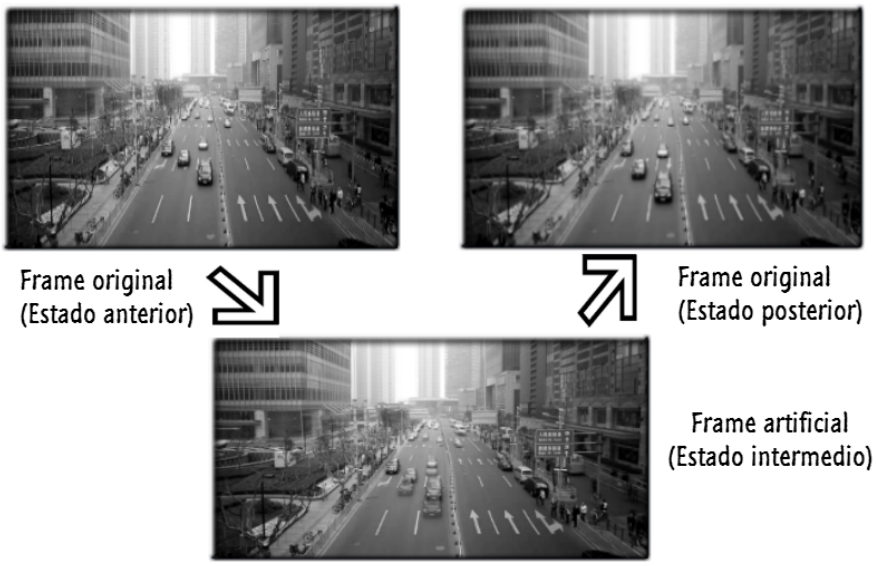

Figura 2. Muestra de interpolación lineal con un solo frame intermedio

$$f(t) = \begin{cases} S_0(t) & \text{si } t \in [t_0, t_1] \\ \vdots & \\ S_{n-1}(t) & \text{si } t \in [t_{n-1}, t_n] \end{cases}$$

Como ya dijimos, habrá una de estas $f_{i,j}$ por cada píxel (i, j), pero nos restringimos a definirla para un único para evitar la notación $_{i,j}$ en todas las ecuaciones.

Cada S_p se define como la siguiente recta:

$$S_p(t) = y_p + (y_{p+1} - y_p) * \frac{t - t_0}{t_1 - t_0}$$

Donde $y_p = frame_p[i, j]$, $y_{p+1} = frame_{p+1}[i, j]$, $t_0 = anterior(t)$ y $t_1 = siguiente(t)$. Es importante aclarar que siempre se debe evaluar esta función dentro del rango (t_0, t_1), es decir, los puntos intermedios.

Para una mayor comprensión, representaremos gráficamente el método. Supongamos que tenemos el algoritmo que instancia una función S_p, tomando la información de un par de píxeles de igual posición (por ejemplo $(1, 2)$) en $frame_0$ y $frame_1$, y la evalúa en fr puntos para generar píxeles artificiales. Asumiendo $fr = 2$, observamos en la figura 3 el resultado.

Si lo usamos considerando ahora múltiples frames ($frame_0 \dots frame_6$) generamos 6 splines $S_0, \dots, S_5$ y 12 frames artificiales como se puede ver en la figura 4.

2.2.3. Construcción mediante interpolación por splines (cúbica)

Continuando con la idea de conseguir los frames artificiales o intermedios mediante la creación y evaluación de polinomios para cada píxel, pasamos a utilizar polinomios de grado 3. Por lo dicho en la introducción teórica creeríamos que esto puede brindar resultados más satisfactorios en la construcción del video final, aunque habrá que confirmar esto durante la experimentación.

Intuición

Siendo éste el tercer y último método a desarrollar, podemos abstraernos de ciertos detalles que fueron previamente explicados.

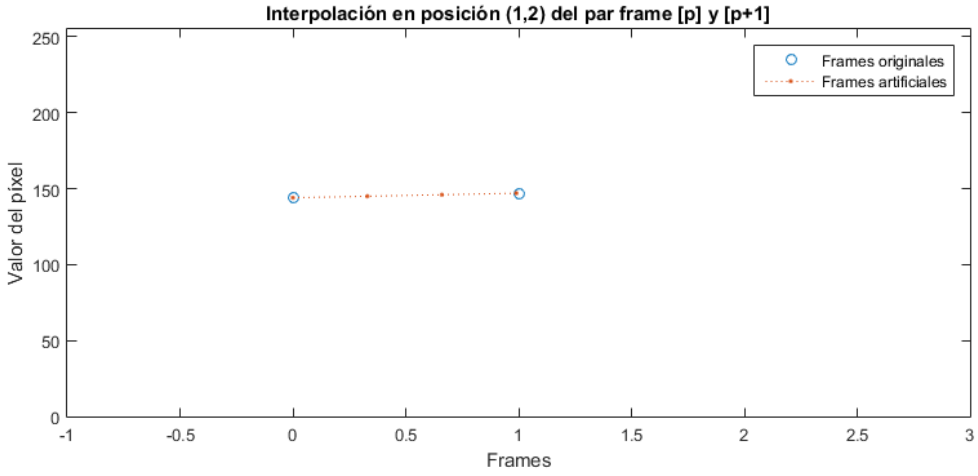

Figura 3. Resultado de interpolar usando 2 frames

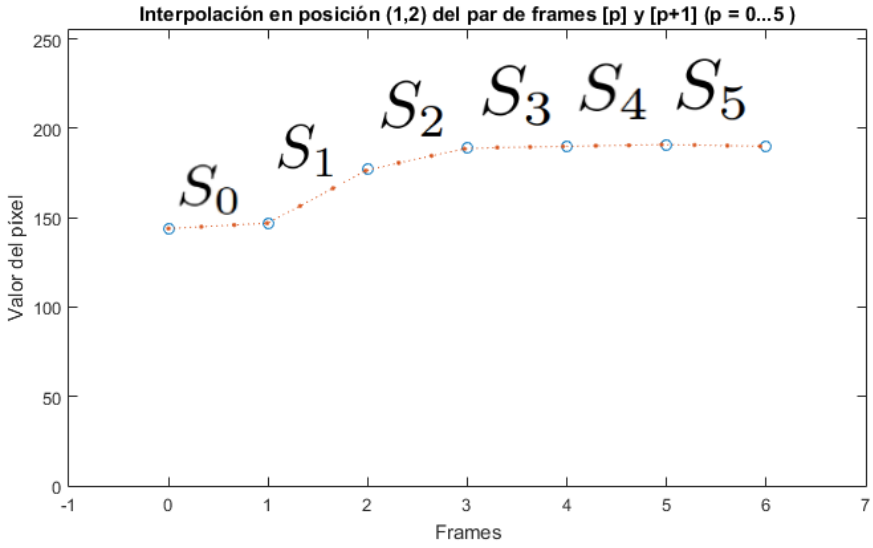

Figura 4. Resultado de interpolar usando 7 frames

Ya hemos dicho que pertenece a la familia de los interpoladores por segmentos. En principio, se podría pensar que el método no cambia su metodología con respecto al anterior. Nos limitaremos aquí a expresar en qué se diferencian.

A diferencia del método de Interpolación lineal, este método varía sus resultados dependiendo de todos los frames considerados para hacer el spline, es decir, no genera cada S_k únicamente en función del valor del píxel en los frames k y $k+1$, sino que todos los frames influyen en los coeficientes de la función S_k, y su evaluación en el punto t puede variar dependiendo de qué frames fueron considerados. Esto se debe, intuitivamente, a que el método utiliza la información de todos los frames para generar cada función $S_{i,j}$, combinándola en un único sistema de ecuaciones[a].

Una opción sencilla en su implementación es usar todos los frames del video para generar el spline. El problema es que el sistema matricial a resolver puede ser muy grande, potencialmente excediendo las capacidades de memoria de las computadoras comunes y trayendo inconvenientes no deseados al uso del método. Queda entonces la aplicación de una alternativa: crear múltiples splines de a bloques. Es decir, tomar cierta cantidad de frames y definirla como el *tamaño de bloque*, para luego generar un spline por cada bloque.

a. ver sección 1.2.3

Perspectiva matemática

Considerando como un nuevo tamaño el tamaño de bloque (que eventualmente podría ser igual a la cantidad de frames del video, dando lugar a un único spline por píxel) generamos un spline por píxel por bloque. Los puntos a interpolar serán el tiempo t y el valor del brillo y de dicho píxel en todos los frames del bloque.

Usando la equivalencia vista en la Introducción teórica (del problema de encontrar los coeficientes de un spline con el de resolver un sistema de ecuaciones lineal) podemos generar la matriz asociada al sistema para cada spline a generar y resolver el sistema usando eliminación gaussiana normalmente. Eso nos da como resultado los coeficientes (a_i, b_i, c_i, d_i) de cada polinomio cúbico S_k de los que componen el spline. Luego basta hacer

$$S(t) = \begin{cases} S_0(t) & \text{si } t \in [t_0, t_1] \\ \vdots & \\ S_{n-1}(t) & \text{si } t \in [t_{n-1}, t_n] \end{cases}$$

para generar cada spline S.

3. IMPLEMENTACIÓN

A continuación nos explayaremos sobre los detalles de la implementación del método propuesto. En las secciones previas hemos dado una introducción al problema, su modelo y justificación de por qué el mismo sirve y a su vez hemos expuesto un método que nos permite hayar la solución.

Datos al aire

$f(x) = a + b(x - x0)$

$y0 + (y1 - y0) * ((x - x0)/(x1 - x0)) \implies a = y0 \ y \ b = (y1 - y0)/(x1 - x0)$

Algoritmo 1: Pseudocódigo del algoritmo de Interpolación lineal

Entrada: Puntos del dominio conocidos x ; Puntos de imágen conocidos y, grado función gr
Salida: Vector coeficiente a, Vector coeficiente b

per $i = 1 \dots gr - 1$ **fai**
  $a_i \leftarrow y_i$;
  $b_i \leftarrow \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$;
fine
devolver a,b

$f(x) = d + c(x - x0) + b(x - x0)^2 + a(x - x0)^3$

Algoritmo 2: Pseudocódigo del algoritmo de Interpolación por Splines

Entrada: Puntos del dominio conocidos x ; Puntos de imágen conocidos y, grado función gr
Salida: Vector coeficiente a, Vector coeficiente b, Vector coeficinete c, Vector coeficiente d

$MatrizSplines(0, 0) \leftarrow 1$;
$MatrizSplines(1, 0) \leftarrow 0$;
$vectorIndep_0 \leftarrow 0$;
per $i = 2 \dots gr - 1$ **fai**
  $MatrizSplines(i, i - 1) \leftarrow x_i - x_{i-1}$;
  $MatrizSplines(i, i) \leftarrow 2\ (x_{i+1} - x_{i-1})$;
  $MatrizSplines(i, i + 1) \leftarrow (x_{i+1} - x_i)$;
  $vectorIndep_i \leftarrow 3\ \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}}$;
fine
$MatrizSplines(gr, gr) \leftarrow 1$;
$MatrizSplines(gr, gr - 1) \leftarrow 0$;
$vectorIndep_{gr} \leftarrow 0$;
$b \leftarrow ResolverSistemaEcuaciones(MatrizSplines, vectorIndep)$;
per $i = 1 \dots gr$ **fai**
  $a_i \leftarrow \frac{1}{3\ \frac{b_{i+1} - b_i}{x_{i+1} - x_i}}$;
  $b_i \leftarrow \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{1}{3\ (2\ b_i + b_{i+1}) * (x_{i+1} - x_i)}$;
fine
$y \leftarrow d$
devolver a,b,c,d

4. EXPERIMENTACIÓN

A continuación se detallan todos los experimentos realizados en este trabajo y sus resultados.

A diferencia de trabajos previos, la experimentación de este trabajo es mucho más explorativa que basada en hipótesis a confirmar. Claramente, al comenzar la experimentación se tuvieron algunas hipótesis (detalladas a continuación), pero se comenzó esta tarea más con un objetivo de observar que ocurre con los métodos propuestos según algunas variables que fueron identificadas.

A su vez, el dominio de este trabajo es más subjetivo, si se quiere, que los trabajos previos. Es decir, al interpolar cuadros (o *frames*) de un video, nos interesa como es la percepción de la audiencia del mismo. ¿Se nota la interpolación?¿Da una sensación anormal?¿No se nota, quizás?. Básicamente: el resultado obtenido, ¿es bueno?. Claramente esto es muy subjetivo y dependiente, entre otras cosas, de la persona que emite el juicio. Así pues, se tuvo que buscar alguna forma de comparar los distintos métodos de la forma más objetiva posible. Pero tampoco se quizo dejar de lado este análisis "subjetivo", que al fin y al cabo, es el más importante.

A lo largo de esta sección comenzaremos primero enunciando las hipótesis que se pudieron enunciar antes de comenzar, la explicación de las variables con las que se experimentó (derivadas a partir de las hipótesis), para luego pasar a la experimentación misma (con un mínimo análisis de correctitud), dónde se analizan los aspectos cuántitativos y cualitativos (y como se a explicado, algunos de estos últimos de manera subjetiva). Finalmente, realizamos una análisis breve de los *artifacts* frutos de los videos interpolados y terminamos esta sección indicando ciertos trabajos a futuros que podrían ser de interés.

4.1. Hipótesis

EN esta parte del trabajo se explican algunas de las hipótesis que pudieron ser formuladas al pensar en el problema. Como ya se ha dicho, la naturaleza de este trabajo ha sido más exploratoria, pero aún así se comenzó el mismo con algunas conjeturas y nociones de lo que debía ocurrir.

1. **Hipótesis** saraza
   Motivación saraza

4.2. Variables Identificadas para la Experimentación

A partir de las hipótesis enunciadas, se determinaron ciertas características y parámetros que posiblemente afecten a la calidad de las interpolaciones aplicadas a los videos. Estas son:

Frames Interpolados
Método de Interpolación
Tamaño de Bloque (splines)
Resolución del Video
Duración del Video
Tipo Movimiento grabado por la Cámara
Tipo de Movimiento de la Cámara

Estas son algunas de las variables que se podrían en tener en cuenta (en particular las que se tuvieron en cuenta en este trabajo), pero no son las únicas. Por dar un ejemplo, una variable con la que no se trabajó fue con los videos que cambián de cámara, donde se pasá de un tipo de imágen a otra de formá rotunda en frames contiguos (podría haber algún tipo de transición también, que sería otro caso a estudiar). Fue meramente por cuestiones de tiempo que se decidió limitar el enfoque de este trabajo a las variables/parámetros enunciados.

4.3. Metodología de Experimentación

DURANTE la experimentación, y dada la naturaleza de la misma, se siguió una misma metodología para recabar la información necesaria para ser luego analizada.

La forma en la que se presentarán los resultados más adelante nada tiene que ver con el orden en el que se realizó la experimentación.

De la sección anterior se puede observar que existen dos variables que nos indican (o limitan) los tipos de videos con los que trabajaremos. Estas dos variables no son otra que las diferentes combinación del tipo de movimiento

filmado y de la cámara que realiza la grabación. Así pues, se terminó teniendo cuatro combinaciónes posibles: *cámara fija-imágen fija*, *cámara fija-imágen móvil*, *cámara móvil-imágen fija* y *cámara móvio-imágen móvil*[a].

Para cada una de estas categorías de videos, se efectuó la interpolación con los 3 métodos expuestos y todas sus posibles combinaciones de parámetros (cantidad de frames a interpolar entre frame y frame, y en el caso de spline el tamaño de bloque.). Para luego poder hacer un análisis objetivo de la calidad de los frames interpolados resultantes, lo que se hizo fue remover de los videos originales una cantidad calculada[b] de frames antes de interpolar el mismo. De esta manera se puede comparar los frames interpolados resultantes con los frames originales (los que fueron removidos previos al procesamiento), que no son otra cosa que el resultado que uno quisiera obtener de la interpolación.

Luego se realizan los análisis de los métodos independientemente de los demás métodos, es decir que analizamos los resultados obtenidos de cada método respecto de sus parámetros de entrada.

Continuamos realizando un análisis de método versus método, para los mismos parámetros (en el caso de splines, al tener 2 parámetros de entrada[c] se tuvieron que utilizar resultados obtenidos del análisis del método realizado previamente).

Por último, antes de terminar enunciando las conclusiones obtenidas para el tipo de video estudiado, se generaron unos videos que realizan la comparativa frame a frame de la diferencia entre el frame del video original y su contraparte interpolada para luego ser estudiados (visualizado en forma de *heatmap*). La idea de esto es la de encontrar en que zonas de los frames, respecto de lo que ocurre en el video, genera problemas para los métodos estudiados.

Las métricas utilizadas para los análisis de los métodos no son otras que las sugeridas por la cátedra: el *Error Cuadrático Medio* [**mse**] y el *Peak Signal to Noise Ratio* [**psnr**]. El primero es una medida de error entre un valor (en nuestro caso, un frame) estimado y lo que es estimado (el frame original). Mientras que el *PSNR* es un ratio que toma en cuenta el *ECM* y el valor máximo siendo estimado[d]. Justamente el PSNR es utilizado para cuantificar la calidad de la reconstrucción de una señal (o en nuestro caso, de un frame), un alto valor del mismo del mismo indica mayor calidad de la reconstrucción (aunque se debe tener cierto cuidado en los casos donde existe compresión de datos involucrada, caso que no aplica a este trabajo).

Vale la pena aclarar que en las conclusiones generales es donde también se realiza un análisis más subjetivo, basado en la percepción de los autores de este trabajo. Claramente esta no es una muestra lo suficientemente alta de las posibles audiencias de los videos, pero alcanza dado el objetivo didáctico del trabajo.

Para concluir con esta sección, agregamos a su vez que se realizaron otros experimentos más puntuales para analizar los aspectos de correctitud de los métodos (mínimamente) y del tiempo de cómputo, como así también una comparativa de como los métodos se ven afectados según la variable más importante de todas: el video.

4.4. Validación de la Implementación

4.4.1. Blanco-Negro

4.4.2. Cámara Fija - Imágen Fija

4.5. Tiempos de ejecución

4.6. Cámara Fija - Imágen Fija

4.7. Cámara Fija - Imágen Móvil

4.8. Cámara Móvil - Imágen Fija

4.9. Cámara Móvil - Imágen Móvil

4.10. Análisis por Método en función del Video

4.11. Artifacts

Experimentos a Futuro

a. Existen matices en estas combinaciones. Por ejemplo, *la suavidad* de los movimientos, que podrían variar en intensidad yendo de suaves a bruscos. Nuevamente, por cuestiones de tiempo se decidió no incursionar en estos aspectos.

b. En base a la cantidad de frames a interpolar.

c. Cantidad de bloques a interpolar y tamaño de bloque.

d. Oriundo del análisis de señales, donde indica la relación entre la potencia máxima de una señal y la potencia del ruido que la afecta.

5. CONCLUSIONES

A lo largo de este trabajo pudimos vislumbrar las complejidades propias del problema de ordenar una colección en principio desordenada de elementos. Tomamos el ejemplo de las páginas web por un lado y el de las competiciones deportivas por otro, enfocándonos particularmente en el fútbol. Usamos el algoritmo de PageRank y su adaptación GeM para resolver ambos problemas respectivamente, y estudiamos su comportamiento al variar diferentes parámetros de entrada, particularmente el factor de teletransportación α.

APÉNDICE A
ENUNCIADO DEL TRABAJO PRÁCTICO

Métodos Numéricos

Segundo Cuatrimestre 2015

Trabajo Práctico 3

Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Un juego de niños

Introducción

¿Quién nunca ha visto un video gracioso de bebés? El éxito de esas producciones audiovisuales ha sido tal que el sitio youborn.com es uno de los más visitados diariamente. Los dueños de este gran sitio, encargado de la importantísima tarea de llevar videos graciosos con bebés a todo el mundo, nos ha pedido que mejoremos su sistema de reproducción de videos.

Su objetivo es tener videos en cámara lenta (ya que todos deseamos tener lujo de detalle en las expresiones de los chiquilines en esos videos) pero teniendo en cuenta que las conexiones a internet no necesariamente son capaces de transportar la gran cantidad de datos que implica un video en *slow motion*. La gran idea es minimizar la dependencia de la velocidad de conexión y sólo enviar el video original. Una vez que el usuario recibe esos datos, todo el trabajo de la cámara lenta puede hacerse de modo offline del lado del cliente, optimizando los tiempos de transferencia. Para tal fin utilizaremos técnicas de interpolación, buscando generar, entre cada par de cuadros del video original, otros ficticios que nos ayuden a generar un efecto de slow motion.

Definición del problema y metodología

Para resolver el problema planteado en la sección anterior, se considera el siguiente contexto. Un video está compuesto por cuadros (denominados también *frames* en inglés) donde cada uno de ellos es una imagen. Al reproducirse rápidamente una después de la otra percibimos el efecto de movimiento a partir de tener un "buen frame rate", es decir una alta cantidad de cuadros por segundo o fps (frames per second). Por lo general las tomas de cámara lenta se generan con cámaras que permiten tomar altísimos números de cuadros por segundo, unos 100 o más en comparación con entre 24 y 30 que se utilizan normalmente.

En el caso del trabajo práctico crearemos una cámara lenta sobre un video grabado normalmente. Para ello colocaremos más cuadros entre cada par de cuadros consecutivos del video original de forma que representen la información que debería haber en la transición y reproduciremos el resultado a la misma velocidad que el original. Las imágenes correspondientes a cada cuadro están conformadas por píxeles. En particular, en este trabajo utilizaremos imágenes en escala de grises para disminuir los costos en tiempo necesarios para procesar los datos y simplificar la implementación; sin embargo, la misma idea puede ser utilizada para videos en color.

El objetivo del trabajo es generar, para cada posición (i, j), los valores de los cuadros agregados en función de los cuadros conocidos. Lo que haremos será interpolar en el tiempo y para ello, se propone considerar al menos los siguientes tres métodos de interpolación:

1. *Vecino más cercano:* Consiste en rellenar el nuevo cuadro replicando los valores de los píxeles del cuadro original que se encuentra más cerca.
2. *Interpolación lineal:* Consiste en rellenar los píxeles utilizando interpolaciones lineales entre píxeles de cuadros originales consecutivos.
3. *Interpolación por Splines:* Simliar al anterior, pero considerando interpolar utilizando splines y tomando una cantidad de cuadros mayor. Una alternativa a considerar es tomar la información de bloques de un tamaño fijo (por ejemplo, 4 cuadros, 8 cuadros, etc.), con el tamaño de bloque a ser determinado experimentalmente.

Cada método tiene sus propias características, ventajas y desventajas particulares. Para realizar un análisis cuantitativo, llamamos F al frame del video real (ideal) que deberíamos obtener con nuestro algoritmo, y sea $\bar{F}$ al frame del video efectivamente construido. Consideramos entonces dos medidas, directamente relacionadas entre ellas, como el *Error Cuadrático Medio* (ECM) y *Peak to Signal Noise Ratio* (PSNR), denotados por $\texttt{ECM}(F, \bar{F})$ y

$\texttt{PSNR}(F, \bar{F})$, respectivamente, y definidos como:

$$\texttt{ECM}(F, \bar{F}) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} |F_{k_{ij}} - \bar{F}_{k_{ij}}|^2 \tag{1}$$

y

$$\texttt{PSNR}(F, \bar{F}) = 10 \log_{10} \left(\frac{255^2}{\texttt{ECM}(F, \bar{F})} \right). \tag{2}$$

Donde m es la cantidad de filas de píxeles en cada imagen y n es la cantidad de columnas. Esta métrica puede extenderse para todo el video.

En conjunto con los valores obtenidos para estas métricas, es importante además realizar un análisis del tiempo de ejecución de cada método y los denominados *artifacts* que produce cada uno de ellos. Se denominan *artifacts* a aquellos errores visuales resultantes de la aplicación de un método o técnica. La búsqueda de este tipo de errores complementa el estudio cuantitativo mencionado anteriormente incorporando un análisis cualitativo (y eventualmente subjetivo) sobre las imágenes generadas.

Enunciado

Se pide implementar un programa en `C` o `C++` que implemente como mínimo los tres métodos mencionados anteriormente y que dado un video y una cantidad de cuadros a agregar aplique estas técnicas para generar un video de cámara lenta. A su vez, es necesario explicar en detalle cómo se utilizan y aplican los métodos descriptos en 1, 2 y 3 (y todos aquellos otros métodos que decidan considerar opcionalmente) en el contexto propuesto. Los grupos deben a su vez plantear, describir y realizar de forma adecuada los experimentos que consideren pertinentes para la evaluación de los métodos, justificando debidamente las decisiones tomadas y analizando en detalle los resultados obtenidos así como también plantear qué pruebas realizaron para convencerse de que los métodos funcionan correctamente.

Programa y formato de entrada

Se deberán entregar los archivos fuentes que contengan la resolución del trabajo práctico. El ejecutable tomará cuatro parámetros por línea de comando que serán el archivo de entrada, el archivo de salida, el método a ejecutar (0 para vecinos más cercanos, 1 para lineal, 2 para splines y otros números si consideran más métodos) y la cantidad de cuadros a agregar entre cada par del video original.

Tanto el archivo de entrada como el de salida tendrán la siguiente estructura:

- En la primera línea está la cantidad de cuadros que tiene el video (`c`).
- En la segunda línea está el tamaño del cuadro donde el primer número es la cantidad de filas y el segundo es la cantidad de columnas (`height width`).
- En la tercera línea está el framerate del video (`f`).
- A partir de allí siguen las imágenes del video una después de la otra en forma de matriz. Las primeras `height` líneas son las filas de la primera imagen donde cada una tiene `width` números correspondientes a los valores de cada píxel en esa fila. Luego siguen las filas de la siguiente imagen y así sucesivamente.

Además se presentan herramientas en Matlab para transformar videos (la herramienta fue probada con la extensión .avi pero es posible que funcione para otras) en archivos de entrada para el enunciado y archivos de salida en videos para poder observar el resultado visualmente. También se recomienda leer el archivo de README sobre la utilización.

Sobre la entrega

- FORMATO ELECTRÓNICO: Martes 10 de Noviembre de 2015, **hasta las 23:59**, enviando el trabajo (`informe` + `código`) a `metnum.lab@gmail.com`. El `asunto` del email debe comenzar con el texto `[TP3]` seguido de la lista de apellidos de los integrantes del grupo. Ejemplo: `[TP3] Artuso, Belloli, Landini`
- FORMATO FÍSICO: Miércoles 11 de Noviembre de 2015, en la clase práctica.

APÉNDICE B
CÓDIGO FUENTE RELEVANTE