Generación de Órdenes Totales mediante el Vector de Perron de Cadenas de Markov Irreducibles

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 aumento 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 [**CITA**], 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 Jacson 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 el ojo humano o incluso quizás una cámara normal 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 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.

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

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 pueden 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:

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

Sin embargo, los polinomios interpoladores 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. Para la aplicación del efecto de $slowmotion$, se utilizará la lineal y cúbica, mientras que la cuadrática será descartada de la asignación. A continuación, se dará una breve explicación de estos dos métodos de interpolación.

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

1.2.2. Lineal

El caso más sencillo lo corresponde 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 en cada uno de ellos, se interpola un segmento equiespaciado sobre x_i y x_{i+1} ($i = 0 \ldots n$). Pero he aquí la primer gran diferencia, y es que dichos polinomios son de grado 3 y por ende, se expresan de la forma:

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

Sin entrar en gran detalle, se desea obtener la tupla de coeficientes (a_i, b_i, c_i, d_i) y para ello se revela una serie de ecuaciones que comprenderán ciertas condiciones, buscando mejorar el desempeño con respecto a su antecesora. Enumeramos estos requisitos sin desprendernos de una perspectiva más conceptual y no tan técnica:

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 de que la función resultante sea continua.
2. Se debe mantener también, la continuidad en las derivadas primeras y segundas. En otras palabras, por cada par S_i y S_{i+1} se debe cumplir en los puntos x_i , la igualdad de evaluar en la derivada primera y segunda dichos polinomios (con $i = 0 \ldots n - 1$).
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* y proviene de una curiosidad.
   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 interpoladora*.

A pesar de su mejora en ciertos aspectos, como la suavidad del $spline$ [a], nos enfrentamos a una elevada cantidad de cálculos a realizar, lo cual nos dirige a un problema en el ámbito computacional.

a. Llamamos $spline$ al conjunto de funciones cúbicas, provenientes de interpolar cada segmento de la función real a aproximar.

2. DESARROLLO

Finalizada la enumeración de ciertos rubros que motiven la investigación sobre el tema, con la adición de una breve perspectiva del problema a trabajar, nos centraremos a desarrollar con mayor precisión, cada método número en cuestión. Recordemos que estas herramientas encaminarán a las posible soluciones al problema de construir, a partir de un video en tiempo real, una secuencia de imágenes generadas de forma artificial que explaye la sensación de que el video original ha sido alentizado (efecto de $slowmotion$).

2.1. Los métodos propuestos

2.1.1. Cuadro más cercano

Este método se basa en una idea de carácter simplista, pero comprende de una intuición que puede resultar de utilidad para ciertos tipos de videos, como por ejemplo, la filmación de un objeto inmóvil.

Se extrae cada cuadro/frame en el orden en que el video los reproduce y tomando de a pares, se le adiciona una cierta cantidad de frames de por medio. No hay un número exacto que represente esta cantidad, por lo que se lo interpreta como parámetro de experimentación. Sin embargo, nos resta como incógnita, el procedimiento por el cual se obtiene dichos cuadros. Esto último es lo que caracterizará al método en cuestión.

Nos abstraemos por un momento de la totalidad de frames que compone una grabación para analizar en detalle el conjunto de imágenes resultantes entre cada par de cuadros de la repoducción original. Asumimos que se decidió la cantidad de frames a realizar. Para un mejor entendimiento, llamaremos a estos cuadros como *frames artificiales*. Por otro lado, el par de frames provenientes del video original, se lo conocerán como *frames originales*.

Como lo indica el sub-título de la sección, mediante un algoritmo que muestra la posición del frame artificial entre los dos frames originales, determinamos a qué distancia se encuentra de ambos extremos. Recordemos que se ingresaron n cuadros entre medio de los originales. Una vez realizado el paso de búsqueda, se decide copiar la imágen del extremo más cercano al frame artificial que se está evaluando.

De esta manera, habrá al menos (despreciando decimales) $n/2$ frames cuya imágen será idéntica a la de alguno de los dos frames originales. En caso que se decida agregar una cantidad impar, se opta por fragmentar en dos partes de $n/2$ y $n/2+1$ cuadros. En la partición de mayor peso , es el usuario quien selecciona qué extremo escoger de los originales. En la siguiente figura, nos encontramos con un ejemplo consico de lo explicado:

(Ejemplo grafíco o dejar eso para el desarrollo)

(Breve explicación del ejemplo)

Volviendo al análisis del video en su mera totalidad, se repite el anterior procedimiento en cada par de frames en el orden concebido por la secuencia del video. De tal forma, se obtienen los frames artificiales, consiguiendo una nueva filmación con el efecto de $slowmotion$ que este método propone.

(Algún comentario final)

2.1.2. Interpolación lineal

Semejante al *cuadro más cercano*, en el sentido que comparten la ideología de trabajar cíclicamente con cada par de frames para eventualmente, obtener el video deseado con su respectivo efecto. No obstante, contaremos lo particular y característico de este método, que puede propocionar ciertas mejoras en el movimiento de objetos durante su filmación.

Por un lado, no posee la misma intuición que su antecesora, donde la imágen de cada frame artificial se basa en copiar alguno de los dos cuadros originales en evaluación. En cambio, se busca utilizar fundamentos matemáticos que ayuden a *predecir y reflejar* lo sucedido entre cada par de frames del video original.

Nuevamente, nos enfocamos en analizar el algoritmo propuesto para la creación de los n frames artificiales que se desean adherir entre cada conjunto de pares. En primer lugar, debemos pensar a cada frame como una matriz de $m\ x\ n$ píxeles (dependiendo la resolución en que se dispone el video), siendo m el largo del cuadro y n el ancho. En segundo lugar, el procedimiento destina a generar un polinomio de grado uno [a] entre ese par de frames originales. Aunque, ¿de qué forma inventaremos dichos polinomios? Para ello, introducimos algunos conceptos que resolverán esta incógnita.

a. También definida como función lineal; https://es.wikipedia.org/wiki/Lineal.

Contamos previamente acerca de la idealización de tales cuadros como matrices compuestas por píxeles. Sin embargo, en ningún momento se aclaró del valor numérico que posee cada posición. Consideramos un rango en el conjunto de números enteros del 0 al 255, inclusive. Esta última se la conoce como **escala de grises en 8 bits** [a], y su importancia revoca en que cada píxel de los frames artificiales adquirirá un valor numérico dentro ese rango.

Dicho esto, el método se centra en crear un polinomio interpolador por cada posición del frame cuyos puntos interpolados **(creo que no es puntos interpolados, pero no se me ocurre el verdadero)** resultan ser los valores de dicha ubicación en el par de frames originales que se está trabajando. Por ende, tendremos por cada par de cuadros reales, $m\ x\ n$ polinomios de grado uno. De lo anterior, nace una nueva duda : ¿Cómo esto resuelve nuestro problema de crear mútiples frames artificiales?

El hecho esta en que ahora conocemos los valores intermedios entre los dos cuadros originales y en consecuencia, podemos particionar ese dominio en p partes (p siendo la cantidad de frames que se adiciona en cada par) y finalmente, evaluar el polinomio en dicho borde de cada fragmento. Como observación, este paso lo tendremos que aplicar para cada posición de la imágen.

De misma forma, se realiza la tarea sobre cada par agrupado en el orden indicado con lo que eventualmente, el método en cuestión concluye con su labor, dejando el efecto de $slowmotion$ que la misma ofrece.

(Cosas para agregar : imagenes o ejemplo sencillo. No se si explicar polinomio interpolado aca o en el desarrollo. Etc. Rta de L: va antes, lo pongo en una sección anterior. Va a haber que reorganizar un poco esto porque nos estamos repitiendo.)

a. Numerical representation : https://en.wikipedia.org/wiki/Grayscale

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.

En esta sección primero hablaremos sobre una implementación del algoritmo **??** (página **??**), su justificación y correctitud. Luego pasaremos a resolver el problema de las estructuras de datos utilizadas para las matrices del problema (las cuales, por representar grafos completos, presentarán incovenientes al querer trabajar con *datasets* cada vez más grandes).

4. EXPERIMENTACIÓN

A continuación se detallan todos los experimentos realizados en este trabajo y sus resultados. Se detalla no sólo el experimento en si, sino que también se explican los resultados que se esperan comprobar y sus motivaciones.

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