



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

## Trabajo Práctico III

---

Métodos Numéricos  
Segundo Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Iván Arcuschin	678/13	iarcuschin@gmail.com
Martín Jedwabny	885/13	martiniedva@gmail.com
José Massigoge	954/12	jmmassigoge@gmail.com
Iván Pondal	078/14	ivan.pondal@gmail.com



Facultad de Ciencias Exactas y Naturales  
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

<http://www.fcen.uba.ar>

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Modelo</b>	<b>4</b>
2.1. Video . . . . .	4
2.2. Vecinos . . . . .	4
2.3. Interpolación Lineal . . . . .	4
2.4. Interpolación por Splines . . . . .	5
2.5. Interpolación por Splines con tamaño de bloque variante . . . . .	5
<b>3. Implementación</b>	<b>6</b>
3.1. Interpolación por vecinos . . . . .	6
3.2. Interpolación lineal . . . . .	6
3.3. Interpolación por Splines . . . . .	7
<b>4. Experimentación</b>	<b>9</b>
4.1. Funcionamiento de los métodos implementados . . . . .	9
4.2. Determinación del tamaño de bloque del método Spline . . . . .	9
4.2.1. Comparación complejidad temporal . . . . .	9
4.2.2. Comparacion ECM y PSNR . . . . .	10
4.2.3. Conclusiones . . . . .	10
4.3. Medición del ECM y PSNR de los métodos. . . . .	10
4.4. Medición de los tiempos de ejecución de los métodos . . . . .	10
4.5. Análisis cualitativos de los métodos, fenómeno de artifacts. . . . .	10
4.5.1. Artifacts: Movimientos Bruscos . . . . .	10
4.5.2. Artifacts: Cambios de Camara . . . . .	10
4.5.3. Artifacts: Movimientos Armonicos . . . . .	10
<b>5. Conclusión</b>	<b>11</b>
<b>6. Referencias</b>	<b>12</b>
<b>7. Enunciado</b>	<b>13</b>

# **1. Introducción**

## 2. Modelo

### 2.1. Video

Definiremos un modelo para los videos con el cual sea fácil de trabajar a la hora de realizar el *slow motion*. Dado un video, definiremos:

- $w$  el ancho en píxeles de cada frame.
- $h$  el alto en píxeles de cada frame.
- $f_i$  el  $i$ -ésimo frame, con  $0 < i < k$ , donde  $k$  es la cantidad de frames totales.
- $p(x, y, f_i)$ , con  $0 < x < w$ ,  $0 < y < h$ , el píxel en la posición  $(x, y)$  del frame  $f_i$ .

Luego, si tomamos  $p(x, y, f_i)$  y  $p(x, y, f_{i+1})$  queremos agregar una cierta cantidad de píxeles entre ambos, de forma que haya una transición del primero al segundo y se produzca el *slow motion*.

Para elegir que valores agregar entre los píxeles, utilizaremos diferentes métodos de interpolación.

- **Vecinos:** Consiste en rellenar los nuevos frames replicando los valores de los píxeles del frame original que se encuentra más cerca.
- **Interpolación Lineal:** Consiste en rellenar los píxeles utilizando interpolaciones lineales entre píxeles de frames originales consecutivos.
- **Interpolación por Splines:** Consiste en rellenar los píxeles utilizando Splines entre píxeles de frames originales consecutivos. En este método, utilizaremos la información provista por todos los frames del video, y generaremos  $k - 1$  funciones, cada una de a lo sumo grado cúbico.
- **Interpolación por Splines con tamaño de bloque variante:** Similiar al anterior, pero con la posibilidad de variar la cantidad de frames tomados en cuenta al generar las funciones.

En las siguientes secciones explicaremos con mayor detalle cada uno de los métodos.

### 2.2. Vecinos

En este método, elijiremos para cada nuevo píxel el valor del frame original que se encuentre más cercano. Si definimos  $c$  como la cantidad de frames a agregar entre cada par original, y  $g_0, \dots, g_{c-1}$  los nuevos frames. Tenemos que:

$$\begin{aligned} p(x, y, f_i) &= p(x, y, f_i) \\ p(x, y, g_0) &= p(x, y, f_i) \\ &\vdots \\ p(x, y, g_{c/2-1}) &= p(x, y, f_i) \\ p(x, y, g_{c/2}) &= p(x, y, f_{i+1}) \\ &\vdots \\ p(x, y, g_{c-1}) &= p(x, y, f_{i+1}) \\ p(x, y, f_{i+1}) &= p(x, y, f_{i+1}) \end{aligned}$$

### 2.3. Interpolación Lineal

En este método, buscaremos interpolar los píxeles de frames contiguos con una función lineal. Para ello, construiremos un Polinomio Interpolante de grado 1 utilizando *diferencias divididas*, ya que ofrece una construcción más sencilla que al seguir el método de Lagrange.

Luego, si llamamos  $f$  a la función (desconocida excepto en los puntos  $x_j$ ), definimos:

- Diferencia dividida de orden cero en  $x_j$ :

$$f[x_j] = f(x_j)$$

- Diferencia dividida de orden uno en  $x_j, x_{j+1}$ :

$$f[x_j, x_{j+1}] = \frac{f[x_{j+1}] - f[x_j]}{x_{j+1} - x_j} = \frac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j}$$

- Polinomio Interpolante de grado 1 para  $x_j, x_{j+1}$ :

$$P_1(x) = f[x_j] + f[x_j, x_{j+1}](x - x_j) = f(x_j) + \frac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j} * (x - x_j)$$

## 2.4. Interpolación por Splines

## 2.5. Interpolación por Splines con tamaño de bloque variante

### 3. Implementación

#### 3.1. Interpolación por vecinos

Este método consiste en reemplazar los cuadros intermedios a ser rellenados por el cuadro original mas cercano en el tiempo. Es decir, dados los cuadros del video sin camara lenta, generamos otro video en camara lenta copiando los cuadros originales de la siguiente manera:

Sean Frame1 y Frame2 dos cuadros consecutivos del video original:

Frame1	Frame2
--------	--------

Si queremos ahora 6 cuadros entre cada 2 del archivo original lo transformamos a:

Frame1	Frame1	Frame1	Frame1	Frame2	Frame2	Frame2	Frame2
--------	--------	--------	--------	--------	--------	--------	--------

El pseudocódigo sería el siguiente:

```
Sean W,H,I el ancho, alto y la cantidad de frames del video original
Sea video[W][H][I] el triple vector de numeros enteros que representa el
    video original
Sea K la cantidad de frames que queremos agregar entre cuadro y cuadro
Crear un triple vector de enteros new_video[W][H][I+(I-1)*K]
Para w = 0 hasta W-1 hacer
    Para h = 0 hasta H-1 hacer
        Para i = 0 hasta I-2 hacer
            Para j = 0 hasta K/2 hacer
                new_video[w][h].push_back(video[w][h][i])
            Fin para
            Para j = (K/2)+1 hasta K hacer
                new_video[w][h].push_back(video[w][h][i+1])
            Fin para
        Fin para
        new_video[w][h].push_back(video[w][h][I-1])
    Fin para
Fin para
Devolver new_video
```

#### 3.2. Interpolación lineal

En este caso, usamos el polinomio interpolador de Lagrange entre cada par de puntos/pixeles consecutivos para aproximar los valores intermedios que irían en el video de camara lenta. Esto genera una función lineal para los pixeles consecutivos en la misma posición.

Por ejemplo, sean dos pixeles con valores 1 y 4:

1	4
---	---

Si queremos un video en camara lenta con 5 cuadros intermedios por cada 2 del original, estos se replicarán de la siguiente forma:

1	1.5	2	2.5	3	3.5	4
---	-----	---	-----	---	-----	---

El procedimiento es el siguiente:

```
Sean W,H,I el ancho, alto y la cantidad de frames del video original
Sea video[W][H][I] el triple vector de numeros enteros que representa el
    video original
Sea K la cantidad de frames que queremos agregar entre cuadro y cuadro
Crear un triple vector de enteros new_video[W][H][I+(I-1)*K]
Para w = 0 hasta W-1 hacer
    Para h = 0 hasta H-1 hacer
        Para i = 0 hasta I-2 hacer
```

```
coef_cero = video[w][h][i]
coef_uno = (video[w][h][i+1] - video[w][h][i]) / (K+1);
Para k = 0 hasta K hacer
    pixel = coef_cero + coef_uno*k;
    Si (pixel < 0) pixel = 0
    Si (pixel > 255) pixel = 255
    new_video.push_back(pixel)
Fin para
new_video.push_back(video[w][h][I-1])
Fin para
Fin para
Devolver new_video
```

### 3.3. Interpolación por Splines

En este método aplicamos la técnica de Splines. Esta consiste en generar un sistema de ecuaciones para encontrar una función por partes que interpole cada par de puntos con polinomios de forma que la curva resultante sea continua y dos veces derivable. Como el sistema es tridiagonal, podemos aprovechar para guardar los valores de la matriz de forma más eficiente. El pseudocódigo es el siguiente:

```
Sean W,H,I el ancho, alto y la cantidad de frames del video original
Sea video[W][H][I] el triple vector de numeros enteros que representa el
    video original
Sea K la cantidad de frames que queremos agregar entre cuadro y cuadro
Crear un triple vector de enteros new_video[W][H][I+(I-1)*K]
Para w = 0 hasta W-1 hacer
    Para h = 0 hasta H-1 hacer
        Crear doble vector de enteros sistema[I][2]
        sistema[0] = {1,0}
        Para j = 1 hasta I-2 hacer
            sistema[j] = {1, 4}
        Fin para
        sistema[I-1] = {0,1}
        // Factorización LU
        Para j = 1 hasta I-2 hacer
            coef = sistema[i + 1][0]/sistema[i][1];
            sistema[i + 1][0] = coef
            sistema[i + 1][1] -= coef
        Fin para
        Crear vectores x[I], a[I], b[I], c[I], d[I]
        Para i = 1 hasta I-2 hacer
            x[i] = 3*(a[i + 1] - 2*a[i] + a[i - 1]));
            x[i] -= x[i - 1]*sistema[i][0];
        Fin para
        // Resuelvo triangular superior (Uc = x)
        Para i = I - 2 hasta 1 hacer
            c[i] = x[i];
            c[i] -= c[i + 1];
            c[i] /= sistema[i][1];
        Fin para
        // Calculo mis coeficientes "b" y "d"
        Para i = 0 hasta I-2 hacer
            b[i] = a[i + 1] - a[i] - (2*c[i] + c[i + 1])/3;
            d[i] = (c[i + 1] - c[i])/3;
```

```
Fin para
//Calculo los pixeles resultantes con el Spline
Para i = 0 hasta I-1 hacer
  Para j = 0 hasta K hacer
    dif = j/(K+1)
    val = a[i] + b[i]*(dif) + c[i]*(dif)*(dif) + d[i]*(dif)*(dif)*(
      dif)
    new_video[w][h].push_back(val)
  Fin para
Fin para
new_video.push_back(video[w][h][I-1])
Fin para
Devolver new_video
```



## 4. Experimentación

En esta sección, se detallan los diferentes experimentos que realizamos para medir el funcionamiento, la eficiencia y calidad de resultados, tanto de forma cuantitativa como cualitativa, de los métodos implementados.

Para lograr tal fin realizamos los siguientes tipos de experimentos:

- Funcionamiento de los métodos implementados.
- Determinación del tamaño de bloque del método interpolación por splines.
- Medición del ECM y PSNR de los métodos.
- Medición de los tiempos de ejecución de los métodos.
- Análisis cualitativos de los métodos, fenómeno de artifacts.

Los videos utilizados para los diversos experimentos, fueron los siguientes:

- **Video 1 (CAMBIAR):** cantidad de cuadros originales: , fps: , duracion:
- **Video 2 (CAMBIAR):** cantidad de cuadros originales: , fps: , duracion:
- **Video 3 (CAMBIAR):** cantidad de cuadros originales: , fps: , duracion:

Es importante mencionar que cada video representa una clase de video distinto, en donde el Video 1 contiene movimientos bruscos, el Video 2 cambios de camara, y el Video 3 movimientos armonicos.

El motivo de estas elecciones se debe a la busqueda de diversos *artifacts* a partir de las características de cada clase.

### 4.1. Funcionamiento de los métodos implementados

En este experimento nuestro objetivo fue asegurarnos el correcto funcionamiento de nuestra implementación de la interpolación fragmentaria lineal, interpolación por splines, e interpolación por splines con tamaño de bloque fijo, tomando bloques de 2 cuadros, 4 cuadros y 8 cuadros.

Con ese fin, generamos diversas instancias de distintos tamaño y comparamos los resultados obtenidos con los resultados de OCTAVE utilizando la funcion *interp1*<sup>1</sup> con los parámetros acorde.

Las instancias utilizadas tienen las siguientes características:

- Cantidad de puntos a interpolar:
- Valores de los puntos a interpolar:

En el siguiente grafico mostramos la diferencia para cada punto calculado con nuestras implementaciones contra los resultados de OCTAVE.

### 4.2. Determinación del tamaño de bloque del método Spline

En este experimento buscamos determinar cual es el mejor tamaño de bloque para la interpolación por splines con tamaño de bloque fijo, teniendo en cuenta la performance y la calidad de los resultados obtenidos de cada tamaño propuesto.

Planteamos los siguientes tamaños de bloques: 2 cuadros, 4 cuadros y 8 cuadros. PORQUE?

Las comparaciones que vamos a realizar seran en terminos de complejidad temporal y ECM, PSNR (ver Seccion4.3 para las definiciones de ambas metricas).

#### 4.2.1. Comparación complejidad temporal

Como instancias de prueba, tomamos cada uno de los videos elegidos y fuimos aumentando la cantidad de cuadros agregados.

En los siguientes graficos mostramos los resultados obtenidos para cada tamaño de bloque:

---

<sup>1</sup>[https://www.gnu.org/software/octave/doc/interpreter/One\\_002ddimensional-Interpolation.html](https://www.gnu.org/software/octave/doc/interpreter/One_002ddimensional-Interpolation.html)

#### 4.2.2. Comparacion ECM y PSNR

#### 4.2.3. Conclusiones

Habiendo hecho las comparaciones entre los diversos tamaños.

### 4.3. Medición del ECM y PSNR de los métodos.

Sea  $F$  un frame del vídeo real (ideal) , y  $\bar{F}$  el mismo frame del vídeo efectivamente contruidos por alguno de los métodos. Sea  $m$  la cantidad de filas de píxeles en cada imagen y  $n$  la cantidad de columnas. Definimos el Error Cuadrático Medio, ECM, como el real dado por:

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

A su vez definimos *Peak to Signal Noise Ratio*, PSNR, como el real dado por:

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

Ambas medidas nos sirven para realizar un análisis cuantitativo de la calidad de los resultados obtenidos con los distintos métodos.

En este experimento utilizamos los videos propuestos al inicio de la experimentacion, variando la cantidad de cuadros que agregamos

Los resultados obtenidos son los siguientes: (GRÁFICO COMPARANDO LOS MÉTODOS)

### 4.4. Medición de los tiempos de ejecución de los métodos

A partir de la implementación descrita en la Sección xxx, podemos inferir una complejidad temporal para cada método: Nuevamente, sea  $m$  la cantidad de filas de píxeles en cada imagen y  $n$  la cantidad de columnas, sea  $c$  es la cantidad de cuadros originales y sea  $f$  la cantidad de cuadros a agregar entre los originales.

- Interpolacion lineal: dado que realizamos 4 ciclos, en donde el primero se ejecuta  $n$  veces, el segundo  $m$  veces, el tercero  $c$  veces, y el cuarto  $f$  veces, la complejidad temporal del mismo es  $\Theta(nmcf)$ .
- Vecino mas cercano: situación idéntica al método lineal, realizamos 4 ciclos, en donde el primero se ejecuta  $n$  veces, el segundo  $m$  veces, el tercero  $c$  veces, y el cuarto  $f$  veces, la complejidad temporal del mismo es  $\Theta(nmcf)$ .
- Interpolacion por Splines:
- Interpolacion por Splines con tamaño de bloque fijo:

Las instancias que utilizamos, en este caso, fueron

Los resultados obtenidos:

### 4.5. Análisis cualitativos de los métodos, fenómeno de artifacts.

Los *artifacts* son errores visuales resultantes de la aplicación de los métodos. Estos errores visuales se caracterizan por romper la coherencia entre imágenes al generar distorsiones evidentes.

#### 4.5.1. Artifacts: Movimientos Bruscos

#### 4.5.2. Artifacts: Cambios de Camara

#### 4.5.3. Artifacts: Movimientos Armonicos

## 5. Conclusión

## 6. Referencias

## 7. Enunciado