

Laboratorio de Métodos Numéricos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico Número 2

Un juego de niños

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	gonzalo.ciruelos@gmail.com
Costa, Manuel José Joaquín	035/14	manuc94@hotmail.com
Gatti, Mathias Nicolás	477/14	mathigatti@gmail.com

asdf

key1 key2 key3 key4

Índice

1. Introducción teórica	3
2. Desarrollo	4
2.1. Convenciones	4
2.2. Métodos numéricos usados	4
2.2.1. Vecino más cercano	4
2.2.2. Interpolación lineal	5
2.2.3. Splines cúbicos	5
3. Resultados y discusión	6
4. Conclusiones	7
5. Apéndices	8

1. Introducción teórica

El objetivo del presente informe es resolver un problema práctico mediante el modelado matemático del mismo. Este problema consiste en generar videos en cámara lenta dados los videos originales, utilizando métodos de interpolación numérica.

Por lo tanto, vamos a tener que colocar más cuadros entre cada par de cuadros consecutivos del video original. Para esto, vamos a pensar un pixel del video a lo largo de todos los frames.

$$p_{ij}(f)$$

Donde el pixel (ij-ésimo) de un video depende del frame. Entonces, la idea va a ser interpolar esta función para obtener los potenciales valores intermedios.

Entonces, por ejemplo, si tenemos un video de 1x1 píxeles, y los valores

$$p_{11}(0), p_{11}(1), \dots, p_{11}(n)$$

Si nos piden colocar 2 nuevos frames entre cada frame viejo, vamos a estar interesado en los valores

$$p_{11}(0), p_{11}\left(\frac{1}{3}\right), p_{11}\left(\frac{2}{3}\right), p_{11}(1), p_{11}\left(\frac{4}{3}\right), p_{11}\left(\frac{5}{3}\right), p_{11}(2), \dots, \\ p_{11}(n-1), p_{11}\left(n-\frac{2}{3}\right), p_{11}\left(n-\frac{1}{3}\right), p_{11}(n)$$

Entonces la idea es fijar un algoritmo de interpolación y obtener todos esos valores, para luego poder rearmar el video.

Estos métodos, además de permitirnos realizar cámaras lentas, también permiten realizar interpolación para otros fines. Por ejemplo, al streamear un video, podríamos bien no recibir todos los cuadros del video, y que el reproductor los interpole acordemente para generar un video fluido. Otro ejemplo de uso es una cámara – de mala calidad – que no puede filmar a 24 cuadros por segundo, pero que sin embargo con estos algoritmos, a partir de videos de menor framerate, podemos generar videos fluidos.

2. Desarrollo

2.1. Convenciones

2.2. Métodos numéricos usados

Como dijimos en la introducción, nuestro objetivo será, dada una lista de valores que toma un pixel a lo largo de diferentes frames, obtener una función que los interpole y luego usarla para obtener valores intermedios.

Dados,

$$p_0, p_1, p_2, \dots, p_n$$

Queremos obtener

$$p_{ij}(f)$$

Tal que

$$p_{ij}(0) = p_0, p_{ij}(1) = p_1, p_{ij}(2) = p_2, \dots, p_{ij}(n) = p_n$$

Y el resto de los valores intermedios se obtienen usando p_{ij} . Entonces, si nos piden colocar k cuadros entre 2 cuadros existentes (supongamos cuadro 0 y cuadro 1 por simplicidad), debemos computar los siguientes valores:

$$p_{ij}\left(\frac{1}{k+1}\right), p_{ij}\left(\frac{2}{k+1}\right), p_{ij}\left(\frac{3}{k+1}\right), \dots, p_{ij}\left(\frac{k}{k+1}\right)$$

Y análogamente para el resto de los cuadros.

Ahora pasemos a ver los métodos implementados y analizados en este trabajo.

2.2.1. Vecino más cercano

El primer método, llamado vecino más cercano o *nearest neighbour* en inglés, se basa en interpolar un punto por el valor del punto más cercano conocido. Formalmente, si $(x_1, y_1), \dots, (x_n, y_n)$ son nuestros puntos de dato, la función que interpola sería:

$$f(x) = f(\arg \min_i |x - x_i|)$$

$$f(x_i) = y_i$$

En el caso particular de este problema, es más simple aún. Si tenemos que colocar k cuadros entre 2 cuadros consecutivos c_1 y c_2 , simplemente podemos poner $\frac{k}{2}$ cuadros con el valor de c_1 y luego $\frac{k}{2}$ cuadros con el valor de c_2 .

En el caso de que k sea impar, simplemente tenemos un valor en el medio al que podemos darle cualquiera de los 2 valores, es una decisión de diseño. En el caso de este trabajo, ese valor es c_2

2.2.2. Interpolación lineal

La interpolación lineal (de a trozos) es la interpolación polinomial de a trozos más simple. Consiste en unir una serie de puntos

$$(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_n, f(x_n))$$

por líneas rectas. Una desventaja de este método, como analizaremos en la experimentación, es que no necesariamente la función resultante va a ser diferenciable en los x_i , algo generalmente deseable, sobre todo en este caso en que la no diferenciabilidad se vería reflejada en cambios bruscos en el video.

En general, dados los puntos $(x_i, f(x_i)), (x_{i+1}, f(x_{i+1}))$, si queremos interpolarlos con una recta obtenemos

$$l_i(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i)$$

Entonces, el algoritmo se basa en esa fórmula, evaluándola en los puntos que se nombraron anteriormente.

2.2.3. Splines cúbicos

Los splines cúbicos son una interpolación polinómica (cúbica) de a trozos, que se basa en pedirle condiciones fuertes a los polinomios cúbicos que interpolan cada trozo de la función. En el caso del spline natural, si le llamamos S_i al i ésimo spline cubico,

1. $S_i(x_i) = S_i(x_{i+1})$, es decir, que la función resultante sea continua en todo punto.
2. $S'_i(x_i) = S'_i(x_{i+1})$, es decir, que la función resultante sea derivable en todo punto.
3. $S''_i(x_i) = S''_i(x_{i+1})$, es decir, que la función resultante sea segundo derivable en todo punto.
4. $S''_0(x_0) = S''_{n-1}(x_n) = 0$, porque esta es la condición del spline natural.

Este método obviamente permite obtener como resultado una función mas apropiada, a priori, para nuestro problema, dado que la derivabilidad nos garantiza que las transiciones van a ser más suaves que con la interpolación lineal.

En el caso de este trabajo, se requirió que la cantidad de cuadros que se tienen en cuenta para hacer el spline sea fija y a elección del usuario. Analizaremos las implicaciones de esto en la experimentación.

Para una explicación más profunda del método, se puede consultar [BF11]. Además, la implementación de splines de este trabajo también esta basada en [BF11], posee algunas correcciones y está modificada para permitir realizar la interpolación de a bloques de tamaño fijo.

3. Resultados y discusión

4. Conclusiones

5. Apéndices

Referencias

[BF11] R. Burden y D. Faires. *Numerical Analysis*. Brooks/Cole, 2011.