Morphing

Joaquín Pérez Araya
Departamento de Ciencias de la Computación
Universidad de Chile
Santiago, Chile
joaquin.perez.a@ug.uchile.cl

***Resumen*—En este documento se aborda el tema del *morphing*, una técnica de mezclado de imágenes la cual consiste en crear un proceso de metamorfosis entre una imagen y otra, también se muestra una implementación realizada a la par con este informe con resultados utilizando diferentes imágenes.**

Introducción

El *morphing* un tipo de efecto visual el cual se produce al cambiar una imagen a otra con un efecto de metamorfosis, actualmente se utiliza principalmente para el entretenimiento. En este documento se implementará el algoritmo descrito por Beier-Neely [1], que consiste en utilizar líneas de correspondencia entre la imagen de partida y la imagen de destino para describir la forma en que la mutación se va a llevar a cabo. Primero se describirá el diseño en la implementación, qué partes del algoritmo se implementaron y de qué forma, luego se probará éste código con diferentes dos muestras con distintos tipos de líneas para observar qué lineas son las mejores para este algoritmo. Para finalizar con las observaciones y conclusiones de dichos experimentos.

Diseño e Implementación

El código implementado se define en 3 funciones las cuales están implementadas en `morphing.py`: *warp*, *morph* y *create morphing video*: La transformación de una imagen según un par de conjuntos de líneas (*warp*), la creación de múltiples imágenes que son las que forman parte del morphing (*morph*) y finalmente la creación, a partir de las imágenes calculadas en el proceso anterior el vídeo que muestra el cambio de las imágenes a lo largo del tiempo *create morphing video*. También se dispone de un módulo adicional llamado `util.py` el cual contiene funciones auxiliares.

Warpping

Para el warpping se utiliza el algoritmo propuesto en el artículo [1], que está especificado en el Algoritmo 1, el cual crea una imagen vacía con las mismas dimensiones de la imagen fuente, y itera sobre ésta, calculando qué colores debería tener según las líneas de correspondencia y la imagen fuente.

Para los parámetros del peso se eligió: $A = 1, B = 1, p = 0{,}5$ En el módulo de utilidades, están implementadas las funciones para el cálculo de u, v, X' y $weight$ según como se indica en el artículo.

La implementación usa el algoritmo en su versión *inverse mapping*, es decir durante la ejecución se recorre la imagen de destino calculando qué pixeles de la imagen original deberían estar allí utilizando interpolación bilineal de los pixeles más cercanos de la imagen original, este método ofrece más simplicidad dado que se conoce los pixeles de destino de antemano por lo que el único inconveniente es el caso de que se requieren píxeles fuera de la imagen original por lo que se interpola según los pixeles ya calculados anteriormente, en cuyo caso sería:

- Si se está en el primer pixel de la imagen, el de la esquina superior izquierda, éste se calcula como el pixel del mismo punto de la imagen de origen.
- Si se está en la fila superior, se calcula usando el pixel anterior calculado, el de la izquierda de éste.
- Si se está en la columna de la izquierda, se calcula usando el pixel de la fila superior.
- Si se está en la columna de la derecha, se calcula utilizando los 3 pixeles que están cercanos a éste: Superior izquierdo, superior e izquierdo.
- De otra forma se calcula utilizando 4 pixeles cercanos: Superior izquierdo, superior, superior derecho e izquierdo.

Algoritmo 1: Warpping

Data: $image$ imagen fuente, $lines_{src}$ líneas fuente, $lines_{dst}$ líneas de destino.
Result: Imagen que corresponde a la imagen fuente modificada según las líneas dadas.

$destinationImage = zeros(shape(image))$
for *Pixel* X *in* $image$ **do**
  $DSUM = (0, 0)$
  $weightsum = 0$
  for *Line* P_iQ_i *in* $lines_{src}$ *and* $P'_iQ'_i$ *in* $lines_{dst}$ **do**
    $u, v = calculate\_u\_v(X, P_i, Q_i)$
    $X' = calculate\_X'(u, v, P'_i, Q'_i)$
    $D_i = X'_i - X$
    $weight = calculate\_weight(X, u, v, P_i, Q_i)$
    $DSUM += D_i * weight$
    $weightsum += weight$
  $\bar{X} = X + DSUM/weightsum$
  $destinationImage(X) = image(\bar{X})$
return $destinationImage$

Morphing y Video

Para la creación del conjunto de imágenes que forman el Morphing completo se realiza lo siguiente: se itera por el número de imágenes n que uno quiere incluir en el vídeo, se calcula el valor de entrelazado t que va desde 0 hasta 1 de $\frac{1}{n}$ en $\frac{1}{n}$ pasos, según el grado de avance del Morphing.
Dado un t de éste proceso, para crear una imagen, se realiza un cross-fade entre dos imágenes: el *warp* de la imagen fuente con sus líneas y la interpolación de las líneas fuente hacia las líneas de destino según t, es decir $t \cdot lines_{src} + (1-t) \cdot lines_{dst}$, y el *warp* entre la imagen de destino con sus líneas y la interpolación de las líneas de destino hacia las fuentes según t, que es $(1-t) \cdot lines_{src} + t \cdot lines_{dst}$. Al final se añade a la colección de imágenes la ponderación de t por la primera imagen más $1-t$ de la segunda.

Luego de finalizado la creación de la colección de imágenes del morphing, se utiliza la librería de OPENCV para formar un vídeo con éstas, en esta implementación, los vídeos generados están a 10 cuadros por segundo y en formato `.avi`.

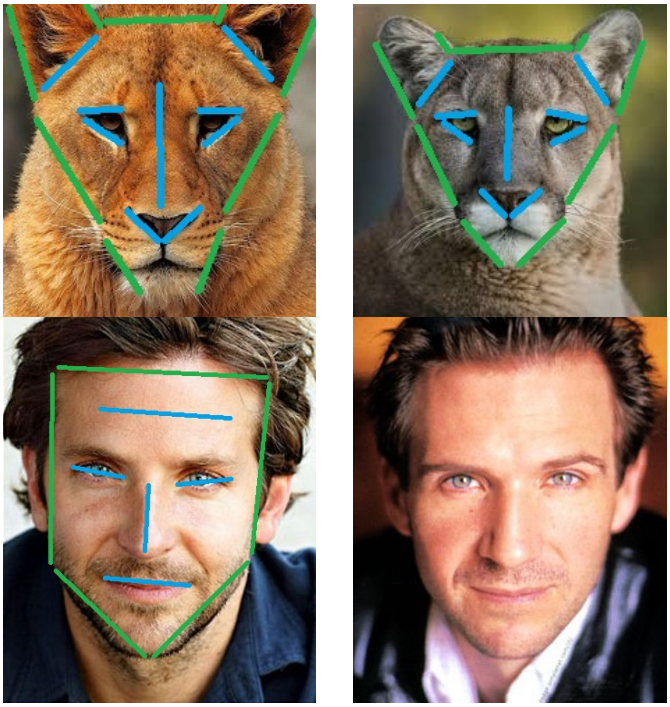

Figura 1. Imágenes usadas para la experimentación, están marcadas las líneas utilizadas para el Morphing, las verdes corresponden a contornos de la figura mientras que las azules corresponden a características intrínsecas de la imagen. Las imágenes de la izquierda corresponden a las de fuente mientras que las de la derecha corresponden a las de destino.

EXPERIMENTACIÓN

Se realizaron pruebas con dos pares de imágenes: Una con un par con dos felinos distinos, un puma y una leona, y la otra con un par de un mismo actor. Para prueba se crearon 3 conjuntos de líneas, uno con líneas de contorno (véase las líneas verdes en la figura 1), uno con líneas de expresión (las líneas azules en la misma figura) y por último uno con ambas líneas en conjunto.

Para cada prueba, se ejecutó la implementación requiriendo 50 imágenes en total, de las cuales, por motivos de simplicidad, sólo se presentarán la $1^{ra}, 10^{ma}, 20^{va}, 30^{va}, 40^{va}$ y 50^{va} imagen de dichas ejecuciones, sin embargo dentro del directorio de la implementación estarán las pruebas con sus respectivas imágenes y vídeos. De aquí en adelante, se mencionará como imágenes verdes, las imágenes que fueron creadas a partir del conjunto de líneas verdes, e imágenes azules de forma análoga.

Imágenes de Caras

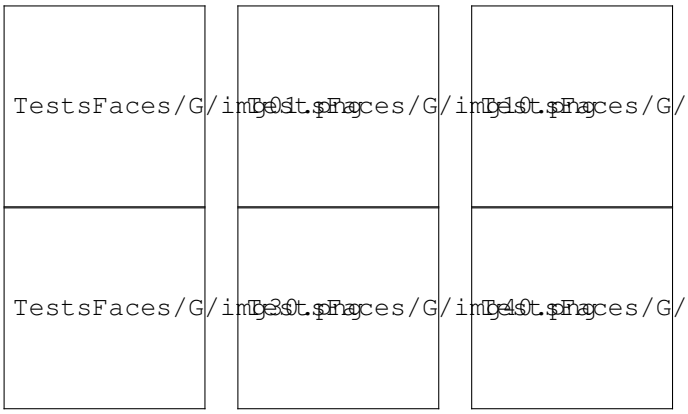

Figura 2. Imágenes creadas con la implementación y con ambos pares de líneas. El progreso va de izquierda a derecha de arriba hacia abajo.

Figura 3. Imágenes creadas con la implementación y con ambos pares de líneas. El progreso va de izquierda a derecha de arriba hacia abajo.

Figura 4. Imágenes creadas con la implementación y con ambos pares de líneas. El progreso va de izquierda a derecha de arriba hacia abajo.

Imágenes de Felinos

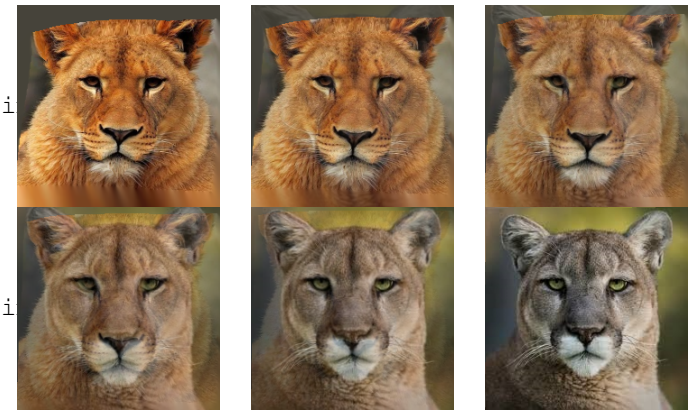

Figura 5. Imágenes creadas con la implementación usando sólo las líneas verdes. El progreso va de izquierda a derecha de arriba hacia abajo.

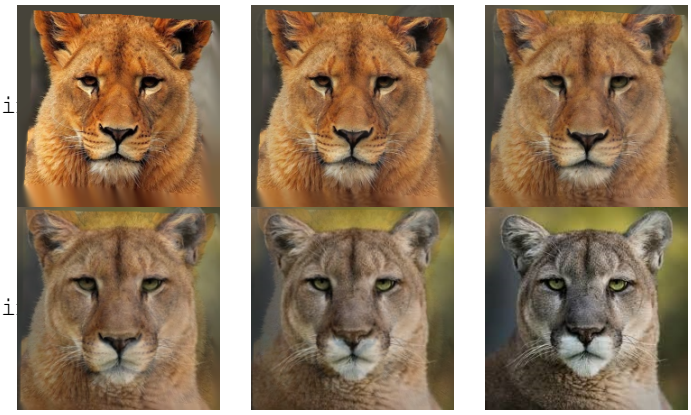

Figura 6. Imágenes creadas con la implementación usando sólo las líneas azules. El progreso va de izquierda a derecha de arriba hacia abajo.

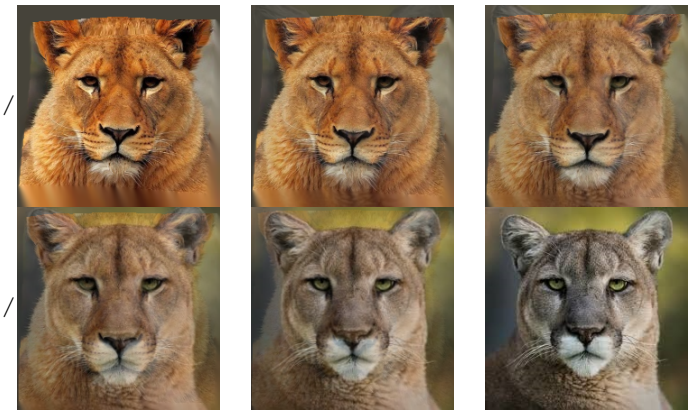

Figura 7. Imágenes creadas con la implementación usando todas las líneas. El progreso va de izquierda a derecha de arriba hacia abajo.

Con el conjunto de imágenes obtenidas con los diferentes experimentos se observa que en las imágenes creadas usando las líneas verdes (de contorno), la imagen inicial se dobla para hacer calzar las orejas de ambos felinos, creando así una aparción de las orejas más armoniosa que en el caso de las imágenes creadas utilizando sólo las líneas azules, sin embargo, el conjunto de imágenes verdes presenta discordancias con las facciones de la cara de los felinos, éstas son más aparentes en la cuarta imagen (la primera de la segunda fila), donde se evidencian éstas en ojos y en nariz. Estos dos detalles no están presentes en las imágenes que usan todas las líneas.

-A. *Errores*

Mientras el desarrollo de los experimentos empezaban se obtuvieron resultados dignos de notar:

Figura 8. Morphing creado con las imagenes y líneas de la derecha a ésta.

Dentro de la experimentación queda en evidencia que la elección de líneas no es trivial, considerando la figura, ahí que la elección de líneas muy cercanas entre unas y otras afectan gravemente determinadas zonas, en el caso de la figura anterior la zona de los ojos.

También se puede evidenciar en el siguiente ejemplo, que hay que utilizar una cantidad mínima de líneas para indicar una misma línea. Dado que el alto número de líneas para describir el borde de ambas caras causa altas anomalías.

Figura 9. Morphing creado con las imagenes y líneas de la derecha a ésta.

Estos errores permitieron realizar mejor

CONCLUSIÓN

El uso de líneas de contorno (verdes) y líneas de características internas (azules) permiten obtener un *morphing* con pocos errores de discordancias, sin embargo, resulta aún más efectivo combinar ambos tipos de líneas. Desafortunadamente esta técnica se vuelve mucho más lenta con cada línea adicional que se agrega, dado que se deben realizar una iteración de *warping* más para obtener la contribución de ésta línea a los puntos de la imagen de destino, por lo que agregar tanto como líneas de contorno como líneas de características internas tendrán un costo importante en el tiempo de ejecución. De esta forma se tiene que priorizar qué líneas se deben incluir si se está buscando el balance entre errores y tiempo de ejecución.

REFERENCIAS

[1] T. Bier, S. Neely. Feature-Based Image Metamorphosis, Computer Graphics 1992-07