CSC320— Visual Computing, Winter 2005

Assignment 2: Image Morphs

Due: 9:10am, Wed., Mar. 2 (at the start of the lecture)
This assignment is worth 10 percent for your grade in this course.

1. **Fourier Transform of Comb Functions [10pts]:** In the lecture notes on sampling we defined the comb function as

$$C(n; n_s) = \sum_{m=0}^{(N/n_s)-1} \delta_{n,mn_s}$$

Here we assume $C(n; n_s)$ is a signal of length N, and N is divisible by n_s. Prove the following proposition:

Proposition 1. The discrete Fourier transform of the comb function is another comb function:

$$\mathcal{F}(C[n; n_s]) = \frac{N}{n_s} C[k; N/n_s]. \tag{1}$$

2. **Convolutions of Fourier Transforms [10pts]:** Prove the following proposition, as stated in the lecture notes on sampling:

Proposition 2. Suppose $f[n]$ and $g[n]$ are two signals of length N (extended to be N-periodic). Then

$$\begin{aligned} \mathcal{F}(f[n]g[n]) &= \frac{1}{N}\mathcal{F}(f) * \mathcal{F}(g) \\ &\equiv \frac{1}{N} \sum_{j=-N/2}^{N/2-1} \hat{f}[j]\, \hat{g}[k-j]. \end{aligned} \tag{2}$$

where $\hat{f}$ and $\hat{g}$ denote the Fourier transforms of f and g, respectively.

3. **Warp for Image Morphing [10pts]:** Read the paper by Beier and Neely, Feature-based image metamorphosis, available from the course home page. Pay particular attention to section 3.

Suppose $I_0(\vec{x})$ and $I_1(\vec{x})$ are two source images to be used in image morphing. Let $\{(\vec{P}_{k,j},\ \vec{Q}_{k,j})\}_{j=1}^{J}$ denote J directed line segments from $\vec{P}_{k,j}$ to $\vec{Q}_{k,j}$ in each of the source images $I_k(\vec{x})$ for $k = 0, 1$. Here the directed segment $(\vec{P}_{0,j},\ \vec{Q}_{0,j})$ in image $I_0(\vec{x})$ corresponds to the j^{th} directed segment $(\vec{P}_{1,j},\ \vec{Q}_{1,j})$ in the second source image $I_1(\vec{x})$. An example of such data is provided in the Matlab code in `morphHandout.zip` available from the home page.

Given these source images, and the sets of corresponding line segments, we wish to form intermediate destination images $I_s(\vec{x})$ for $s \in [0, 1]$. To do this, we first define interpolated segments $(\vec{P}_{s,j},\ \vec{Q}_{s,j})$ to have the endpoints

$$\begin{aligned} \vec{P}_{s,j} &= (1-s)\vec{P}_{0,j} + s\vec{P}_{1,j}, \\ \vec{Q}_{s,j} &= (1-s)\vec{Q}_{0,j} + s\vec{Q}_{1,j}. \end{aligned}$$

(see the Matlab script `showLineInterp.m` available from the above zip file).

Given these interpolated segments we can describe any point $\vec{x}$ in the destination image $I_s(\vec{x})$ in terms of (u, v) coordinates relative to the j^{th} segment $(\vec{P}_{s,j},\ \vec{Q}_{s,j})$. Beier and Neely define these (u, v) coordinates as follows:

$$\begin{aligned}
L_{s,j} &= ||\vec{Q}_{s,j} - \vec{P}_{s,j}||_2, \\
\vec{t}_{s,j} &= (\vec{Q}_{s,j} - \vec{P}_{s,j})/L_{s,j}, \\
\vec{n}_{s,j} &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \vec{t}_{s,j}, \\
u_{s,j}(\vec{x}) &= \vec{t}_{s,j}^{\,T}(\vec{x} - \vec{P}_{s,j})/L_{s,j} \\
v_{s,j}(\vec{x}) &= \vec{n}_{s,j}^{\,T}(\vec{x} - \vec{P}_{s,j})
\end{aligned}$$

Here $(u_{s,j}, v_{s,j})$ provide coordinates for the point $\vec{x}$ relative to the j^{th} line. Indeed, it follows that

$$\vec{x} = \vec{P}_{s,j} + (\vec{Q}_{s,j} - \vec{P}_{s,j})u_{s,j} + \vec{n}_{s,j}v_{s,j}. \tag{3}$$

So, for example, $(u, v) = (0, 0)$ corresponds to $\vec{x} = \vec{P}_{s,j}$, and $(u, v) = (1, 0)$ corresponds to $\vec{x} = \vec{Q}_{s,j}$.

Since the directed line segment $(\vec{P}_{s,j},\ \vec{Q}_{s,j})$ in the destination image $I_s(\vec{x})$ is meant to correspond to the segment $(\vec{P}_{k,j},\ \vec{Q}_{k,j})$ in the k^{th} source image, it is natural to use the same (u, v) coordinates for corresponding points in the source image $I_k(\vec{x})$. That is, define the warp function for the j^{th} segment to be

$$\vec{W}_{k,s,j}(\vec{x}) = \vec{P}_{k,j} + (\vec{Q}_{k,j} - \vec{P}_{k,j})u_{s,j}(\vec{x}) + \vec{n}_{k,j}v_{s,j}(\vec{x}), \tag{4}$$

This is identical to (3) except that endpoints $\vec{P}_{s,j}$ and $\vec{Q}_{s,j}$ in the destination image $I_s(\vec{x})$ have been replaced by the endpoints $\vec{P}_{k,j}$ and $\vec{Q}_{k,j}$ in the k^{th} source image, and the derived normal $\vec{n}_{s,j}$ is replaced by $\vec{n}_{k,j}$. As a consequence of this definition, $(u, v) = (0, 0)$ and $(1, 0)$ produce $\vec{W}_{k,s,j}$ equal to $\vec{P}_{k,j}$ and $\vec{Q}_{k,j}$, respectively. Moreover, points $\vec{x}$ on the segment between $\vec{P}_{s,j}$ and $\vec{Q}_{s,j}$ map to points $\vec{W}_{k,s,j}(\vec{x})$ on the corresponding segment $(\vec{P}_{k,j}, \vec{Q}_{k,j})$.

Finally, we need to combine these individual segment warp functions $\vec{W}_{k,s,j}(\vec{x})$ together to form a single warp function $\vec{W}_{k,s}(\vec{x})$ which takes the location $\vec{x}$ in the destination image to the corresponding location $\vec{W}_{k,s}(\vec{x})$ in the k^{th} source image. Beier and Neely suggest doing this by allowing each of the segment-based warps $\vec{W}_{k,s,j}(\vec{x})$ to vote for the source location, with the amount of the vote denoted by $m_{s,j}(\vec{x}) \geq 0$. The combined warp is then defined by

$$\vec{W}_{k,s}(\vec{x}) = \frac{\sum_{j=1}^{J} m_{s,j}(\vec{x})\vec{W}_{k,s,j}(\vec{x})}{\sum_{j=1}^{J} m_{s,j}(\vec{x})} \tag{5}$$

The denominator here is just the total number of votes, and therefore the right hand side is just a weighted average of the warps $\vec{W}_{k,s,j}(\vec{x})$ for individual segments.

In order for this warp $\vec{W}_{k,s}(\vec{x})$ to roughly map each segment $(\vec{P}_{s,j}, \vec{Q}_{s,j})$ to the corresponding segment $(\vec{P}_{k,j}, \vec{Q}_{k,j})$ we need the votes $m_{s,j}(\vec{x})$ to be large when $\vec{x}$ is near $(\vec{P}_{s,j}, \vec{Q}_{s,j})$ and the other votes $m_{s,l}(\vec{x})$ relatively small for $l \neq j$. In this case, the right hand side in (5) reduces to roughly $\vec{W}_{k,s,j}(\vec{x})$, which properly maps the j^{th} segment.

Beier and Neely propose that the voting functions be defined by

$$\vec{m}_{s,j}(\vec{x}) = \left(\frac{L_{s,j}^{p}}{a + d_{s,l}(\vec{x})} \right)^{b}. \tag{6}$$

Here $a \in (0, 1]$, $b \in [1/2, 2]$, $p \in [0, 1]$ are selected constants (see Beier and Neely's paper). And $d_{s,l}(\vec{x})$ is the minimum distance between $\vec{x}$ and the segment $(\vec{P}_{s,j}, \vec{Q}_{s,j})$. That is,

$$d_{s,j}(\vec{x}) = \begin{cases} ||\vec{x} - \vec{Q}_{s,j}||_2 & \text{for } u_{s,j}(\vec{x}) > 1, \\ |v_{s,j}| & \text{for } 0 \leq u_{s,j}(\vec{x}) \leq 1, \\ ||\vec{x} - \vec{P}_{s,j}||_2 & \text{for } u_{s,j}(\vec{x}) < 0. \end{cases} \tag{7}$$

Your job is to write a Matlab function

```
function [W0, W1] = morphWarps(linePairs, szIm, s, a, b, p)
```

where `linePairs` is as in `showLineInterp.m`, `szIm` is the size of the source images $I_0(\vec{x})$ and $I_1(\vec{x})$, and $s \in [0,1]$ is the interpolation parameter for the destination image. The last three arguments specify the constants a, b, and p used in the definition of the votes, namely equation (6).

This function should return the warp functions as defined above, namely $\vec{W}_{k,s}$ for $k = 0, 1$, in the 3-dimensional arrays $W0$ and $W1$, respectively. That is, $W0(j,i,1)$ gives the first component of $\vec{W}_{0,s}$ evaluated at $\vec{x} = (i,j)$, and similarly for $W1(j,i,1)$. These warps need to be evaluated for all $\vec{x} = (i,j)$ (i.e. $1 \leq i \leq$ `szIm`(2) and $1 \leq j \leq$ `szIm`(1)).

In order to run quickly in Matlab (and for you to get more than half marks on this question) you **must not use a loop over image positions** $\vec{x}$. Instead, use one loop over the corresponding line segments, and use array operations over **all pixel locations** to build up the required warps, as defined in (5).

You will be required to submit your M-file `morphWarps.m` electronically.

4. **Perform the Warp by Looping Over Pixels [10pts]:** Write a Matlab function

```
function ims = morphImagesLoop(im0, im1, linePairs, s)
```

where `im0` and `im1` are gray-level images and `linePairs` are the corresponding line segments, as in `showLineInterp.m`. The last parameter $s \in [0,1]$ is the interpolation parameter for the destination image. This function should return the morphed image

$$I_s(\vec{x}) = (1-s)I_0(\vec{W}_{0,s}(\vec{x})) + sI_1(\vec{W}_{1,s}(\vec{x})). \tag{8}$$

This function should loop over pixels $\vec{x} = (i,j)$ and perform a bicubic spline interpolation using the Catmull-Rom interpolation kernel discussed in the interpolation notes. (Since the displacements are likely to vary with image position, we cannot simply use convolution.)

Hand in printed copies of the resulting morphed images for s equal to $1/4$, $1/2$ and $3/4$. You will also be required to submit your M-file `morphImagesLoop.m` electronically.

Note that in performing the warps we did not use any prefiltering to blur the source images. In your write up, explain precisely when this may lead to aliasing and poor results. (You do **not** need to fix this problem.)

5. **Perform the Warps Using Interp2 [10pts]:** Redo problem 4, only this time write a Matlab function

```
function ims = morphImages(im0, im1, linePairs, s)
```

which **does not loop over pixel positions**. Instead it uses the Matlab image interpolation function `interp2` to perform the interpolation.

Hand in printed copies of the resulting morphed images for s equal to $1/4$, $1/2$ and $3/4$. You will also be required to submit your M-file `morphImages.m` electronically.