Image Interpolation

• Introduction
  – What is image interpolation?
  – Why do we need it?
• Linear Interpolation Techniques
  – Pixel replication
  – Bilinear
  – Bicubic*
• Applications
  – Digital zooming
  – Error concealment
  – Geometric transformations

Introduction

• What is image interpolation?
  – An image f(x,y) tells us the intensity values at the integral lattice locations, i.e., when x and y are both integers
  – Image interpolation refers to the “guess” of intensity values at arbitrary locations, i.e., x and y can be any real numbers
  – Note that it is just a guess (Note that all sensors have finite sampling distance)
Introduction (Con’t)

• Why do we need image interpolation?
  – We want BIG images
    • When we see a video clip on a PC, we like to see it in the full screen mode
  – We want GOOD images
    • If some block of an image gets damaged during the transmission, we want to repair it
  – We want COOL images
    • Manipulate images digitally can render fancy artistic effects as we often see in movies

Image Interpolation

• Introduction
  – What is image interpolation?
  – Why do we need it?
• Linear Interpolation Techniques
  – Pixel replication
  – Bilinear and Bicubic*
• Applications
  – Digital zooming
  – Error concealment
  – Geometric transformations
Replication (Zero-order)

\[ f(n) \]
\[ \downarrow \]
\[ f(x) \]

Bilinear Interpolation (first-order)

\[ f(n) \]
\[ \downarrow \]
\[ f(x) \]
**Bilinear Interpolation Formula**

Basic idea: the closer to a pixel, the higher weight is assigned

\[
f(n+a) = (1-a)f(n) + af(n+1), \quad 0 < a < 1
\]

Note: when \( a = 0.5 \), we simply have the average of two

---

**Numerical Examples**

\( f(n) = [0, 120, 180, 120, 0] \)

Interpolate at 1/2-pixel

\( f(x) = [0, 60, 120, 150, 180, 150, 120, 60, 0], \quad x = n/2 \)

Interpolate at 1/3-pixel

\( f(x) = [0, 20, 40, 60, 80, 100, 120, 130, 140, 150, 160, 170, 180, ...], \quad x = n/6 \)
Bicubic Interpolation (third-order)*

From 1D to 2D

- Just like separable 2D transform (filtering) that can be implemented by two sequential 1D transforms (filters) along row and column direction respectively, 2D interpolation can be decomposed into two sequential 1D interpolations.
- The ordering does not matter (row-column = column-row)
- Such separable implementation is not optimal but enjoys low computational complexity
Graphical Interpretation of Interpolation at Half-pel

f(m,n)       g(m,n)

Numerical Examples

zero-order

\[
\begin{array}{cccc}
  a & a & b & b \\
  a & a & b & b \\
  c & c & d & d \\
  c & c & d & d \\
\end{array}
\]

first-order

\[
\begin{array}{cccc}
  a & (a+b)/2 & (a+c)/2 & (a+b+c+d)/4 \\
  (a+b+c+d)/4 & (b+d)/2 & (c+d)/2 & d \\
\end{array}
\]
Numerical Examples (Con’t)

Q: what is the interpolated value at Y?
Ans.: \((1-a)(1-b)X(m,n)+(1-a)bX(m+1,n)
+a(1-b)X(m,n+1)+abX(m+1,n+1)\)

Image Interpolation

- **Introduction**
  - What is image interpolation?
  - Why do we need it?
- **Linear Interpolation Techniques**
  - Pixel replication
  - Bilinear and Bicubic*
- **Applications**
  - Digital zooming
  - Error concealment
  - Geometric transformations
Pixel Replication

low-resolution image (100x100) → high-resolution image (400x400)

Bilinear Interpolation

low-resolution image (100x100) → high-resolution image (400x400)
Bicubic Interpolation

low-resolution image (100 × 100) → high-resolution image (400 × 400)

Error Concealment

damaged  interpolated
Image Inpainting

Widely used in computer graphics to generate special effects
MATLAB function: maketform, imtransform
Basic Principle

• \((x, y) \rightarrow (x', y')\) is a geometric transformation
• We are given pixel values at \((x, y)\) and want to interpolate the unknown values at \((x', y')\)
• Usually \((x', y')\) are not integers and therefore we can use linear interpolation to guess their values

Rotation

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  \cos \theta & \sin \theta \\
  -\sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
Rotation Example

\[ \theta = 3^\circ \]

Scale

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
a & 0 \\
0 & 1/a
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix}
\]
Affine Transform

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  d_x \\
  d_y
\end{bmatrix}
\]

square \rightarrow \text{parallelogram}

Affine Transform Example

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  .5 & 1 \\
  .5 & -2
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  0 \\
  1
\end{bmatrix}
\]
Shear

\[
\begin{bmatrix}
x'
\end{bmatrix} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}
\]

Shear Example

\[
\begin{bmatrix}
x'
\end{bmatrix} = \begin{bmatrix} 1 & 0 \\ .5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]
Projective Transform

\[ x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} \]
\[ y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1} \]

EE465: Introduction to Digital Image Processing

Projective Transform Example

\[
\begin{bmatrix}
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
-4 & 2 & -8 & -3 & -3 & -5 & 6 & 3
\end{bmatrix}
\]