Image and video processing
(EBU723U)

Image transformations

Dr. Yi-Zhe Song
yizhe.song@qmul.ac.uk
Today’s agenda

• Algebraic transformations
• Geometric transformations
Today’s agenda

- Algebraic transformations
- Geometric transformations
Example: background subtraction
Algebraic operations

- Binary operators involving more than one image
- Usually the images are of the same dimension
- Examples
  - addition of two or more images
  - subtraction of one image from another
  - multiplication and division
Algebraic operations

• Non invertible
  – Original information is lost (e.g., 2+2 = 3+1 = 1+3 = …)

• Clipping
  – Combination of pixel values may exceed range of output variable
  – then clipping occurs, e.g., if unsigned byte, 230 +80 = 310 > 255

• Normalization
  – In general, may need to map \( f \in [a, b] \) to \( g \in [c, d] \)
    \[
g = c + \frac{f - a}{b - a} (d - c)
    \]

Where typically \( c = 0 \) and \( d = 255 \) for eight-bit images
Example

3 + 5 = 8

Grid diagram with numbers 3, 5, and 8 connected by arrows.
Example: noise removal

\[ \text{+} \]

\[ \times \frac{1}{2} = \]

EBU723U
Noise Removal

Single image

Average of 10

Average of 50

Average of 100
Sum for noise reduction

- If statistically independent noise is added to each of several identical images
- Or multiple noisy images are summed
  - The sum image has a higher SNR
- SNR can be computed by taking ratio of expected values of Signal and Noise
Edge detection
Edge detection using subtraction

• If an image is displaced (translated) relative to another image, then the difference between them approximates the first derivative

\[
\frac{\Delta y}{\Delta x} = \frac{f(x + \Delta x) - f(x)}{\Delta x}
\]

where \(\Delta x\) is the displacement

• This is called a *forward difference* approximation because \(f(x)\) is compared to a value on the right. The *central difference* is often a better approximation:

\[
\frac{\Delta y}{\Delta x} = \frac{f(x + \Delta x/2) - f(x - \Delta x/2)}{\Delta x}
\]
Finite-difference filters

- Derivative operators can be implemented as linear filters.
- The backward x-difference (applied to image-rows) filter is \( t = [-1, 1] \)
- The central x-difference is between values at \( x+1/2 \) and \( x-1/2 \)
- Let the image-row (locally) be \( f = [a, b, c] \)
- Values between \( a,b \) and between \( b,c \) are computed by the averaging filters
  - \( u = [1/2, 1/2, 0] \) and \( v = [0, 1/2, 1/2] \) respectively.
- Then the central difference is \( v \cdot f - u \cdot f = (v-u) \cdot f \)
- So we can define the central x-difference filter as \( w = v-u = [-1/2, 0, 1/2] \)
- The y-difference filters can be similarly constructed.
Edge detection using subtraction

Original

Gradient Image
Edge-profile and derivative

\[ f(t) \]

\[ f'(t) \]
Generalization of scalar derivative: Gradient

Given an image: $f(x, y)$, then

$$\nabla f(x, y) = \left[ \frac{\partial}{\partial x} f(x, y), \frac{\partial}{\partial y} f(x, y) \right]$$

where

\nabla is called the gradient operator

Note that the gradient ‘image’ contains a vector for each pixel!
Gradient properties

- Points in direction of max upward slope
- Is a vector *perpendicular* to the *iso-contour* of the intensity
- Gradient magnitude
  - Useful scalar function of gradient
  - is equal to value of slope

\[ |\nabla f(x,y)| = \sqrt{\left(\frac{\partial}{\partial x} f(x,y)\right)^2 + \left(\frac{\partial}{\partial y} f(x,y)\right)^2} \]
Gradient magnitude

- Square root
  - computationally expensive $\rightarrow$ sometimes simpler approximation

$$|\nabla f(x,y)| \approx \max \left[ \left| f(x,y) - f(x+1,y) \right|, \left| f(x,y) - f(x,y+1) \right| \right]$$
Directional derivative

- More generally, given the horizontal and vertical derivatives, the derivative in any other direction $\theta$ can be obtained:

$$\frac{\partial}{\partial \theta} f(x, y) = \cos(\theta) \frac{\partial}{\partial x} f(x, y) + \sin(\theta) \frac{\partial}{\partial y} f(x, y)$$

- This is just the dot-product of the gradient with the unit-vector in direction $\theta$.
- E.g. the derivative at angle zero is just $\frac{\partial}{\partial x} f(x, y)$
- This property is called *steerability* of the gradient
Today’s agenda

• Algebraic transformations
• Geometric transformations
Spatial transformations
Example: rotation

- Simply a matrix multiplication
Example: scaling
Geometric operations

• Change spatial relationship among pixels in an image
  – Continuous: rubber-sheet geometry
  – Discontinuous: change pixel neighborhoods

• Two considerations
  – Spatial transformation
  – Grey-level interpolation
Coordinate transformations

\[
I_{\text{old}}(x, y) \ast (x, y)_{\text{old}} \rightarrow (x', y')_{\text{new}} = \left( a(x_{\text{old}}, y_{\text{old}}), b(x_{\text{old}}, y_{\text{old}}) \right)_{\text{new}}
\]

\[
I_{\text{new}}(x', y')
\]
Scaling

- Scaling operation:

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = 
\begin{bmatrix}
ax \\
by
\end{bmatrix}
\]

- Or, in matrix form:

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = 
\begin{bmatrix}
a & 0 \\
0 & b
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]

scaling matrix
2D rotation

\[(x', y')\]

\[(x, y)\]
2D rotation

- Or, in matrix form:

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

- Even though \( \sin(\theta) \) and \( \cos(\theta) \) are nonlinear functions of \( q \),
  - \( x' \) is a \textit{linear combination} of \( x \) and \( y \)
  - \( y' \) is a \textit{linear combination} of \( x \) and \( y \)

- To derive this, represent \([x,y]\) in the form \([r \cos(\Phi), r \sin(\Phi)]\)
- Represent \([x',y']\) in the form \([r \cos(\Phi+\Theta), r \sin(\Phi+\Theta)]\)
- Use the trigonometric identities:

\[
\cos(\Phi+\Theta) = \cos(\Theta) \cos(\Phi) - \sin(\Theta) \sin(\Phi)
\]
\[
\sin(\Phi+\Theta) = \sin(\Theta) \cos(\Phi) + \cos(\Theta) \sin(\Phi)
\]
Matrix representation - summary

- Represent 2D transformation by a matrix

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

- Multiply matrix by column vector
  ⇔ apply transformation to point

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
a & b \\
c & d
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix}
\]

\[
x' = ax + by
\]

\[
y' = cx + dy
\]
Matrix representation - summary

• Transformations combined (‘composed’) by multiplication
• Matrix multiplication is associative

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

Matrices are a convenient and efficient way to represent a sequence of transformations

• Matrices are invertible if the determinant is nonzero
• The determinant of \( \begin{bmatrix} a & b \\ c & d \end{bmatrix} \) is the scalar \( ad - bc \)
2x2 matrices

• What types of transformations can be represented with a 2x2 matrix?

2D identity

$x' = x$

$y' = y$

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

2D scale around (0,0)

$x' = s_x \times x$

$y' = s_y \times y$

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  s_x & 0 \\
  0 & s_y
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2x2 matrices

• What types of transformations can be represented with a 2x2 matrix?

2D rotate around (0,0)

\[
\begin{align*}
x' &= \cos \Theta \cdot x - \sin \Theta \cdot y \\
y' &= \sin \Theta \cdot x + \cos \Theta \cdot y
\end{align*}
\]

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

2D shear

\[
\begin{align*}
x' &= x + sh_x \cdot y \\
y' &= sh_y \cdot x + y
\end{align*}
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} =
\begin{bmatrix}
1 & sh_x \\
sh_y & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]
2x2 matrices

• What types of transformations can be represented with a 2x2 matrix?

2D mirror about Y axis

\[
x' = -x \\
y' = y
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

2D mirror over (0,0)

\[
x' = -x \\
y' = -y
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]
2x2 matrices

• What types of transformations can be represented with a 2x2 matrix?

2D translation

\[ x' = x + t_x \]
\[ y' = y + t_y \]

Only linear 2D transformations can be represented with a 2x2 matrix

NO!
Homogeneous coordinates

- How can we represent translation as a 3x3 matrix?

\[ x' = x + t_x \]
\[ y' = y + t_y \]
Homogeneous coordinates

- Homogeneous coordinates
  - represent coordinates in 2 dimensions with a 3-vector
  - seem unintuitive, but they make graphics operations much easier
Homogeneous coordinates

- How can we represent translation as a 3x3 matrix?
  - Using the rightmost column

\[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
x' = x + t_x \\
y' = y + t_y
\]
Homogeneous coordinates

- In a plane, the **homogeneous coordinates** of a point whose Cartesian coordinates are \((x, y)\) are any three numbers \((a_1, a_2, a_3)\) for which

\[
\frac{a_1}{a_3} = x
\]

\[
\frac{a_2}{a_3} = y
\]

Let \(a_3 = 1\)
Coordinate translation

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}_{\text{new}} = \begin{bmatrix}
  a(x, y) \\
  b(x, y) \\
  1
\end{bmatrix}_{\text{new}} = \begin{bmatrix}
  1 & 0 & x_0 \\
  0 & 1 & y_0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}_{\text{old}}
\]

\((x_0, y_0)\) \(I_{\text{new}}\) Coordinates

\(I_{\text{old}}\) Coordinates
Coordinate scaling

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix}_{\text{new}} = \begin{bmatrix}
    a(x, y) \\
    b(x, y)
\end{bmatrix}_{\text{new}} = \begin{bmatrix}
    \frac{1}{c} & 0 & 0 \\
    0 & \frac{1}{d} & 0 \\
    0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}_{\text{old}}
\]
Coordinate rotation (around the origin)

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}_{\text{new}} =
\begin{bmatrix}
  a(x,y) \\
  b(x,y) \\
  1
\end{bmatrix}_{\text{new}} =
\begin{bmatrix}
  \cos(\theta) & -\sin(\theta) & 0 \\
  \sin(\theta) & \cos(\theta) & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}_{\text{old}}
\]
Compound transformations

- Compound transformations
  - order of operations is important
  - perform individual operations and combine into single function
  - Example: rotation around arbitrary point \((x_0, y_0)\)
    - Translate from \((x_0, y_0)\) to \((0, 0)\)
    - Rotate
    - Translate back to \((x_0, y_0)\)
Rotation around arbitrary point

\[
\begin{bmatrix}
  x' \\
y' \\
\end{bmatrix}_{\text{new}} = \begin{bmatrix}
a(x, y) \\
b(x, y) \\
1 \\
\end{bmatrix}_{\text{new}} = \begin{bmatrix}
1 & 0 & x_0 \\
0 & 1 & y_0 \\
0 & 0 & 1 \\
\end{bmatrix}\begin{bmatrix}
x \\
y \\
\end{bmatrix}_{\text{transl}}
\]

\[
\begin{bmatrix}
x \\
y \\
1 \\
\end{bmatrix}_{\text{transl}} = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1 \\
\end{bmatrix}\begin{bmatrix}
x \\
y \\
\end{bmatrix}_{\text{trans & rot}}
\]
3D basic transformation in space

- **Rotation around X-axis**
  $$\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & -\sin \theta & 0 \\
0 & \sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

- **Rotation around Y-axis**
  $$\begin{bmatrix}
\cos \theta & 0 & \sin \theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

- **Rotation around Z-axis**
  $$\begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$
Skew and shear

- Take an image and skew it to the side

\[ Skew_\theta = \begin{bmatrix} 1 & \frac{1}{\tan \theta} \\ 0 & 1 \end{bmatrix} \]

- Effect of the transformation
  - Squares become parallelograms
    - \( y \) coordinates skew to the right
    - \( x \) coordinates stay the same
  - \( 90^0 \) between axes becomes \( \theta \)
  - Each scan line of the original image shifts relative to the one below it
Skew and shear
Skew and shear

- Skew and shear
  - Everything along the line $y=1$ stays on the same line $y=1$, but is translated to the right
  - Distance between points on this line is preserved
Example
Inverse warping

- In general, if we map each pixel $P = [x, y]$ to a new position $P' = W(x, y)$, we would end up with *holes* in the new image!
- This can be avoided by using the *inverse* of the warp $W$.
  - Loop through all destination pixels $[x', y']$
  - Find corresponding source position $[x, y] = W^{-1}(x', y')$
  - Find colour $f_{x'y'} = f_{xy}$ by interpolation in source image.

- Recall bilinear interpolation:

  $$f_{xy} = [1 - y, y] \begin{bmatrix} f_{00} & f_{10} \\ f_{01} & f_{11} \end{bmatrix} \begin{bmatrix} 1 - x \\ x \end{bmatrix}$$

- Interpolation may blur the result slightly.
What did we learn today?

• Algebraic transformations
• Geometric transformations