Digital Image Modifications

I. Kolingerová
Contents:
1. Geometrical transformations
2. Emboss
3. Warping
4. Morphing
5. Imitation of painters’ techniques
6. Sharpening
7. Blur
8. Image transitions
References

- ....
Image for us: a matrix of a) intensity pixels, b) colour components triples (R,G,B)

1. Geometrical transformations

- A source image is transformed to a destination image:

\[ S(u_p, v_q) \rightarrow D(x_j, y_k) \]

- \[ 0 \leq p \leq P-1, \ 0 \leq q \leq Q-1, \]
- \[ 0 \leq j \leq J-1, \ 0 \leq k \leq K-1 \]
Geometrical transformations:

Most often translation, rotation, scale

- Translation:
  - Forward mapping – computed for each pixel $[u_p,v_q]$ of source image
    \[ x_j = u_p + t_x \quad y_k = v_q + t_y, \]
    where $(t_x, t_y)$ is a translation vector
    $x_j, y_k$ are not necessarily integers ($\leq t_x, t_y$), due to rounding, holes may appear
  - Backward mapping – computed for each pixel $[x_j,y_k]$ of destination image:
    \[ u_p = x_j - t_x \quad v_q = y_k - t_y, \]
    where $(t_x, t_y)$ is a translation vector
    $u_p, v_q$ are not necessarily integers ($\leq t_x, t_y$), then interpolation
Geometrical transformations:

- Similar equations are possible for rotation and scale but more often, matrix equations are used.

Translation:
\[
\begin{bmatrix}
 x_j \\
 y_k
\end{bmatrix} = \begin{bmatrix}
 u_p \\
 v_q
\end{bmatrix} + \begin{bmatrix}
 t_x \\
 t_y
\end{bmatrix}
\]

Scaling:
\[
\begin{bmatrix}
 x_j \\
 y_k
\end{bmatrix} = \begin{bmatrix}
 s_x & 0 \\
 0 & s_y
\end{bmatrix} \begin{bmatrix}
 u_p \\
 v_q
\end{bmatrix}
\]

Rotation:
\[
\begin{bmatrix}
 x_j \\
 y_k
\end{bmatrix} = \begin{bmatrix}
 \cos \theta & -\sin \theta \\
 \sin \theta & \cos \theta
\end{bmatrix} \begin{bmatrix}
 u_p \\
 v_q
\end{bmatrix}
\]

- Translation, then scale, then rotation:
\[
\begin{bmatrix}
 x_j \\
 y_k
\end{bmatrix} = \begin{bmatrix}
 s_x \cos \theta & -s_y \sin \theta \\
 s_x \sin \theta & s_y \cos \theta
\end{bmatrix} \begin{bmatrix}
 u_p \\
 v_q
\end{bmatrix} + \begin{bmatrix}
 s_x t_x \cos \theta & -s_y t_y \sin \theta \\
 s_x t_x \sin \theta & s_y t_y \cos \theta
\end{bmatrix}
\]
Geometrical transformations:

- It can be rewritten as
  \[
  \begin{bmatrix}
  x_j \\ y_k
  \end{bmatrix} =
  \begin{bmatrix}
  c_0 & c_1 \\ d_0 & d_1
  \end{bmatrix}
  \begin{bmatrix}
  u_p \\ v_q
  \end{bmatrix}
  + \begin{bmatrix}
  c_2 \\ d_2
  \end{bmatrix}
  \]

- \((c_0 \ldots d_2\text{ see previous equations})\)

- The same relations expressed using a square matrix:
  \[
  \begin{bmatrix}
  x_j \\ y_k \\ 1
  \end{bmatrix} =
  \begin{bmatrix}
  c_0 & c_1 & c_2 \\ d_0 & d_1 & d_2 \\ 0 & 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  u_p \\ v_q \\ 1
  \end{bmatrix}
  \]
Geometrical transformations:

- The inverse computation is:

\[
\begin{bmatrix}
u_p \\
v_q \\
1
\end{bmatrix} =
\begin{bmatrix}
a_0 & a_1 & a_2 \\
b_0 & b_1 & b_2 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_j \\
y_k \\
1
\end{bmatrix}
\]

- Translation:
  
  \[a_0 = 1, \ a_1 = 0, \ a_2 = -t_x,\]
  
  \[b_0 = 0, \ b_1 = 1, \ b_2 = -t_y\]

- Scale:
  
  \[a_0 = 1/s_x, \ a_1 = 0, \ a_2 = 0,\]
  
  \[b_0 = 0, \ b_1 = 1/s_y, \ b_2 = 0\]

- Rotation:
  
  \[a_0 = \cos \theta, \ a_1 = \sin \theta, \ a_2 = 0,\]
  
  \[b_0 = -\sin \theta, \ b_1 = \cos \theta, \ b_2 = 0\]
Geometrical transformations:

- Another example:
  Shear: \(c_0=d_1=1, \ c_1=0.1, \ d_0=0, \ c_2=d_2=0\)
2. Emboss

1. An auxiliary image – an inverse of the source
2. Translation/shift of the auxiliary image (user-given, usually 1 pixel in both x and y)
3. Sum of the source and shifted auxiliary image
4. All pixel intensities are divided by 2 (the average of the source and auxiliary images)

- More precisely:
  \[ D[i,j] = \frac{1}{2} \times (S[i,j] + (1-S[i+1,j+1])) \]
  where \( S[i,j] \) – a source image pixel,
  \( D[i,j] \) – a destination image pixel
An example - emboss

Can be done in colour components, or in intensity (weighted sum of components)
3. Warping

- A “free deformation” of the image
- 2 basic algorithms: grid-based or line-segments-based

An example of grid-based warping
Grid-based warping

- Defined by a grid put on the image
- Polylines, Bézier cubics or spline curves are used as boundaries of cells
- The user moves the nodes
- 4gons then change their shape correspondingly
- Suitable for global changes in the image
Grid-based warping

- 2 iterations
- 1\textsuperscript{st} iteration: points moved in rows
  - Computation of intersections of vertical curves/polylines with image rows => new intervals on the rows
  - Resampling of intervals (most often a linear interpolation), output is an inter-image I
- 2\textsuperscript{nd} iteration: points of inter-image moved in columns (using horizontal curves/polylines)
Line segment warping

- Also called field warping, feature-based warping
- For local changes in the image (e.g., to close eyes)
- Backward mapping is used – a corresponding pixel of the source image S is looked for the pixel of the destination image D
- Local changes are defined by line segments (each line segment in S has a corresponding line segment in D)
Line segment warping

- Compute $u, v$, then a source pixel $X'$ can be found for a destination pixel $X$
- $v$ – absolute distance,
- $u$ - normalized

$$u = \frac{(X - P) \cdot (Q - P)}{||Q - P||^2}$$

$$v = \frac{(X - P) \cdot (Q - P)_{\perp}}{||Q - P||}$$

$$X' = P' + u(Q' - P') + \frac{v.(Q' - P')_{\perp}}{||Q' - P'||}$$
Line segment warping

- **More line segments used:**
  - To find $u,v$ for all line segments and one destination pixel $X$,
  - Each line segment provides one $X_i'$
  - Resulting $X'$ is a weighted sum of all $X_i'$

$$D_i = X_i' - X$$

$$X' = X + \frac{\sum_{i=1}^{n} \vec{D}_i w_i}{\sum_{i=1}^{n} w_i}$$

$$w_i = \left( \frac{|Q'_i - P'_i|_p}{(a + \text{dist})} \right)^b$$

- Line segment should not cross, otherwise strange effects
Line segment warping - parameters

- $a, b, p$ given by the user
- Influence of particular segment on the transformation

- Suitable values: e.g., $a=0.001$, $b=2$, $p=0.5$
- $a$ – line segment influence, $a \to 0$; a bigger value – a smoother warp, but less exact user control
- $b$ – fading influence of the line segment with the distance ($b=0$ – all line segments the same influence, $b$ big – only the nearest line segments influence)
- $p$ from $<0,1>$, a bigger value – bigger influence of longer line segments
- $dist$ – distance of the point $X$ from l.s. $P_iQ_i$; $dist = \text{abs}(v)$ for $u$ from $<0,1>$, $dist=|PX|$ for $u<0$, $dist=|QX|$ for $u>1$

$$w_i = \left(\frac{|Q'_i - P'_i|^p}{(a + dist)}\right)^b$$
Line segment warping

- The algorithm for the whole image:

- For all pixels $X$ of destination image
  - $d_{\text{sum}} = (0,0), w_{\text{sum}} = 0$ // accumulated vector of sums, accumulated sum of weights
  - For all pairs of line segment $P_iQ_i$ and $P_i \)'Q_i \)'$
    - Compute $P_i$ and $Q_i$ from $u_i$ and $v_i$
    - Compute $X_i \)'$ from $P_i \)', Q_i \)', u_i$ and $v_i$
    - $D_i = X_i \)' - X$
    - Compute $\text{dist}$ between $X$ and $P_iQ_i$
    - Compute $w_i$
    - Compute $d_{\text{sum}} = d_{\text{sum}} + w_iD_i$
    - $w_{\text{sum}} = w_{\text{sum}} + w_i$
    - Compute $X \)' = X + d_{\text{sum}}/w_{\text{sum}}$
    - $D(X) = \text{Source}(X \)'$

- More images: line segment vertices gradually change, usually by linear interpolation
4. Morphing

- Transfer of one image into another (also possible for geometric objects)
- Warp of source image $S$ to $S_i$, destination $D$ to $D_i$ and interpolation between $S_i$ and $D_i$
- Either interpolation of corresponding pixels of both images, or with the computation of correspondence of pixels from both images
5. Imitation of painters’ techniques

- Impressionism – 2nd half of 19th cent., capturing of countryside in a time point (spreading pure tones in spots, halftones obtained as late as on the retina of the observer).

- Pointilism – colours placed with the stress on the contrast of complementary.

- Given: A simple shape (circle, ellipse, square), thickness of the brush, a direction to draw (a vector) and a template (a dig. image).

- Subdivide the template into squares a x a, average a square and find the closest colour in the palette, make a dot or a draw in the given direction; the draws overlap without colour blending.

- All randomly modified (the direction, draw length, colour).
Claude Monet: The Cliffs at Etretat, 1885
Claude Monet:  
*Woman with a Parasol, Facing Right*  
(also known as  
*Study of a Figure Outdoors (Facing Right))*  
1886
Claude Monet: Cathedral in the morning sun, 1894
Georges Seurat: 
Sitting model, profile, 
1886
Georges Seurat:
La Maria, Honfleur,
1886
Georges Seurat: 
Le Chahut, 
1889-90
Computer-produced impressionism
Computer-produced pointilism
6. Image sharpening

- Based on edge detection and enhancement
- An edge given by the value and direction of the gradient

Let \( s(i,j) \) be the value of the gradient of the image \( f \) in a point \( i,j \), then the image \( g \) obtained by sharpening \( f \) in the point \( i,j \) is

\[
g(i,j) = f(i,j) + c \cdot s(i,j)
\]

where \( c \) is the sharpening coefficient.
Gradient in the digital image - ambiguous

- The simplest: Roberts operator- intensity change in the direction of the main and side diagonals

\[ |\nabla f(i, j)| = |f(i, j) - f(i+1, j+1)| + |f(i, j+1) - f(i+1, j)| \]
Laplace operator- differences of intensity in the directions perpendicular to the coordinate axes

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & -8 & 1 \\
1 & 1 & 1
\end{bmatrix}
\]

\( h \) is used in the so-called discrete convolution:

\[
I'[x, y] = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I[x + i, y + j] h[i + k, j + k]
\]
Edge points extraction using Laplace operator

4x4  8x8

8x8
Roberts, 
~10% points

Laplace 4x4, 
~10% points

Laplace 8x8, 
~10% points

Original
Extraction \(~10\%\) edge points + triangulation + interpolation

- Roberts
- Laplace 4x4
- Laplace 8x8
Examples – edge enhancement (here too much - see artefacts)
Other operators: **Sobel operator** – directional, 8 variant, etc.

Operators enhancing edges enhance all high frequencies, thus they are sensitive to noise, sensitivity decreases with a growing neighbourhood.

Sobel operators detecting vertical and horizontal lines
7. Blur

- Replace the pixel by the average of the neighbourhood
8. Image transitions

- Visual effects transforming a scene A to a scene B
- Not too much stress – it should transform the attention from A to B, not to disturb
- The simplest solution: dissolve, alfa-blending
- Fading out – dissolve A to black, from black to B
Wipe

- Vertical, horizontal or other directions

- The boundary can be complex (a), fuzzy (b)
Iris effect

Cliche: heart-shaped iris effect 😊
Biased wipe

- Besides images transition also, e.g., a scale

- Suitable, e.g., to emphasize something in A, B (scaling down and up)