CV Cheat Sheet

Images

- color depth = bit depth = bits per pixel = bbp
- dithering: putting 2 colors close to give illusion of a 3rd color.
- grayscale img: 8 bpp, black = 0 & white = 255.
- color: 24 bpp, 8 bits for each channel of RGB, & possible alpha channel. $\alpha = 0 \implies$ fully transp., $\alpha = 1 \implies$ fully opaque.

BMP Format

file header (size, offset, ...)
info header (DIB) (width, height, ...)
optional color palette
image data

File Header

14 bytes

- magic identifier: 2 bytes
- file size : 4 bytes
- 2 reserved places: 2 bytes each
- offset to image data: 4 bytes

Info Header

40 bytes

- header size in bytes: 4 bytes
- width and height : 4 bytes each
- number of color planes: 2 bytes
- number of bits per pixel: 2 bytes
- compression (0 to 3): $0 = none$
- image size in bytes

Note that the order is $B \to G \to R$.

Color Palette

- If present, then a pixel is stored in ≤ 1 bytes.
- Each color entry is in $RGBA$ format with 4 bytes.
- If not present, $offset = 14 + 40 = 54$, else $offset = 54 + 4 * nColors$.

Arithmetic Operations

Addition

$I(x, y) = I_1(x, y) + I_2(x, y)$
OR
$I(x, y) = I_1(x, y) + C$

Overflow

$I(x, y) > max\ (255)$?

1. Wrapping: $I'(x, y) = I(x, y) - (max + 1)$
2. Saturation: $I'(x, y) = max$

Subtraction

usage: detect changes between 2 images.
$I(x, y) = I_1(x, y) - I_2(x, y)$
OR
$I(x, y) = I_1(x, y) - C$

Underflow

$I(x, y) < 0$?

1. Wrapping: $I'(x, y) = I(x, y) + (max + 1)$
2. Saturation: $I'(x, y) = 0$
3. Absolute: $I'(x, y) = |I(x, y)|$

Multiplication

usage: enhance contrast
$I(x, y) = I_1(x, y) * I_2(x, y)$
OR
$I(x, y) = I_1(x, y) * C$

Division

$I(x, y) = I_1(x, y) \div I_2(x, y)$
OR
$I(x, y) = I_1(x, y) \div C$

Blending

$I(x, y) = I_1(x, y) * C + I_2(x, y) * (1 - C)$

Logical Operations

NOT, AND/NAND, OR/NOR, XOR/XNOR

Conversion

1. Bitwise: convert values to binary base and apply operators bitwise.
2. Thresholding: convert each pixel value to 1 bit: $I_{new} = \begin{cases} 0, & I > 127, \\ 1, & I \leq 127. \end{cases}$

Applications

- **AND/NAND**: intersection bet. 2 images.
- **OR/NOR**: union bet. 2 images.
- **NOT**: negative of input image.

Logical NOT

Ways to apply NOT:

- normal boolean NOT.
- grayscale: $I'(x, y) = 255 - I(x, y)$.
- float pixel format: $I'(x, y) = -I(x, y)$, followed by normalization.

Bitshift Operations

usage: fast multiplication and division

- **Shift left** i bits = 2^i (multiplication)
- **Shift right** i bits = $\div 2^i$ (division)

Empty Places

1. fill with zeroes.
2. fill with ones.
3. fill with bits from other side (rotate).

Geometric Operations

Translation

inhomogeneous

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

homogeneous

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Rotation

- inverse rotation prefered (from dest. to source) to avoid gaps.
- y upwards $\implies$ counterclockwise is $+ve$, y downwards $\implies$ clockwise is $+ve$.

Rotate about origin:
inhomogeneous

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

homogeneous

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Rotate about arbitrary pt:

1. Translate to origin.
2. Rotate about origin.
3. Translate back.

Scaling

- inverse scaling prefered (from dest. to source) to avoid gaps.

inhomogeneous

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

homogeneous

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Subsampling (shrink)

1. **Replacement:** replace a group of pixel by a chosen one (upper left).
2. **Interpolation:** mean value of group.

Upsampling (magnify)

1. **Replication:** fill group of pixels with same value of original pixel.
2. **Interpolation:** get missing values at boundaries, then interpolate by distance.

Reflection

About $x - axis$:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

About $y - axis$:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

About a general axis:

1. Translate to origin.
2. Rotate abt origin to fit on x- or y-axis.
3. Reflect about this axis.
4. Rotate back.
5. Translate back.

Affine Transformation

- 6 degrees of freedom.
- needs 3 pairs of points to estimate.

inhomogeneous

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

homogeneous

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How to get matrix A from 3 pairs of points?

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix}$$

$$\begin{bmatrix} a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \end{bmatrix}$$

$$\begin{bmatrix} a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Homography Transformation

- 8 degrees of freedom.
- needs 4 pairs of points to estimate.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

How to get matrix H from 4 pairs of points?
Let $h_{33} = 1$, $M =$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 \end{bmatrix}$$

$$M \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_1 \\ y'_2 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

Digital Filters

Convolution

For an $n \times m$ kernel K:

$$I(x,y) = \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} I_1(x+k, y+l)K(k,l)$$

Noise

1. Salt & pepper noise: the color of a noisy pixel has no relation to surrounding pixels.
2. Gaussian noise: each pixel is changed from its original value by a small amount.

Smoothing Filters

Linear Filters

$LFilter(I_1 + I_2) = LFilter(I_1) + LFilter(I_2)$

1. Uniform (mean) filter
2. Triangular filter
3. Gaussian filter

Non-linear Filters

1. Median filter
2. Kuwahara filter

Uniform (Mean) Filter

- replace each pixel with the mean of its neighbourhood.
- all coeffs. in kernel have same weights.
- smoothing effect increases with kernel size.
- filter is always normalized (divide by sum of weights).

1	1	1
1	1	1
1	1	1

$* \frac{1}{9}$ — 3×3 kernel

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$* \frac{1}{25}$ — 5×5 rectang. kernel

0	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	1	1	0

$* \frac{1}{21}$ — 5×5 circ. kernel $(R = 2.5)$

Triangular Filter

- similar to mean filter, but weights are diff.
- filter is always normalized (divide by sum of weights).

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

$* \frac{1}{81}$ — 5×5 pyramid. kernel

0	0	1	0	0
0	2	2	2	0
1	2	5	2	1
0	2	2	2	0
0	0	1	0	0

$* \frac{1}{25}$ — 5×5 cone kernel $(R = 2.5)$

Gaussian Filter (Blur)

Gaussian in 1D:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Gaussian in 2D:

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

1	4	7	4	1
4	16	26	16	4
7	26	40	26	7
4	16	26	16	4
1	4	7	4	1

$* \frac{1}{272}$ — 5×5 Gauss. $(\sigma = 1)$

Mean Filter

- reduces noise, but preserves details.
- replace each pixel with the median of its neighbourhood.
- sort values and pick the middle one (or average of two middles if even).
- $median(I_1 + I_2) \neq median(I_1) + median(I_2)$.

Kuwahara Filter

- edge-preserving filter, doesn't disturb sharpness and position of edges.

$$\text{Variance: } \sigma^2 = \frac{\sum_{i=1}^{N} (I(x_i) - mean)^2}{N}$$

1. Calculate mean and variance of each 3×3 region (upper left, upper right, lower left, & lower right).
2. Output value of center pixel = mean value of region of smallest variance.