Computer Vision Cheat Sheet

March 22, 2022

1 Image Filtering

1.1 Convolution

1. Proof of associativity $f * (g * h) = (f * g) * h$:

   https://math.stackexchange.com/questions/2170534/proof-of-associativity-of-convolution

2. Prove the associativity property for 1D discrete convolution

$$\begin{aligned}
f * (h_1 * h_2) &= \sum_{j=-\infty}^{\infty} f[j] \sum_{k=-\infty}^{\infty} h_1[k] h_2[i-j-k] \\
&= \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} f[j] h_1[i-j-k] h_2[k] \\
&= \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[j] h_1[i-j-k] h_2[k] \\
&= \sum_{k=-\infty}^{\infty} (f * h_1)[i-k] h_2[k] \\
&= (f * h_1) * h_2
\end{aligned}$$

4. Properties

   - Translation invariant - Shares weights in kernel

1.2 Smoothing

1. Gaussian smoothing:

   - A form of denoising to improve edge detection response

     Larger σ (more blurry) surpresses noise and results in smoother derivatives.

   - Used with Harris detector to change the scale in order to detect objects of different scales in images e.g. humans to buildings

     Different σ values finds edges at different scales. Since the Harris detector is not invariant to scale, a corner could be detected as an edge if it's too big/wide. Hence, we need to use the Gaussian filter to scale the edges at different levels (thus search on both spatial and scale domain) and find the response at the most suitable scale

   - Image augmentation for motion detection purposes

     Images can be blurred to better train models to detect objects in motion as objects taken in still motion would not be representative of objects in motion

1.3 Separable filter

1. Show that the 2D Gaussian filter is a separate filter.

$$\begin{aligned}
f[x,y] * h[x,y] &= \sum_i \sum_j f[x-i, y-j].h[i,j] \\
&= \sum_i \sum_j f[x-i, y-j].\frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \\
&= \sum_i \left(\sum_f f[x-i, y-j].\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{j^2}{2\sigma^2}} \right. \\
&= \sum_i f * h_y[x-i].\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{i^2}{2\sigma^2}} \\
&= (f * h_y) * h_x
\end{aligned}$$

1.4 Median filtering

1.

2 Edge detection

1. Convolution process:

   - Flip kernel (For 2D, both horizontally and vertically)
   - Multiply signal with flipped kernel
   - Sum over the support of kernel
   - Divide by the size of the kernel

2. Edge detection filters:

   - Sobel:

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$h_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Prewitt:

$$h_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$h_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

3. Computational cost
   - Non-separable - $N \times N$ image with $K \times K$ filter:

     Multiplications: K^2N^2
     Additions: $(K^2 - 1)N^2$
     O-notation: $O(K^2N^2)$

   - Separable filters - $N \times N$ image with $K \times 1$ and $1 \times K$ filter::

     Multiplications: $2MN^2$
     Additions: $2(K - 1)N^2$
     O-notation: $O(KN^2)$

4. Gradient magnitude and orientation:

   Magnitude: $g = \sqrt{g_x^2 + g_y^2}$
   Orientation: $\theta = \text{arctan2}(g_y, g_x)$

2.1 Canny Edge detection

1. Criteria for good edge detector:
   - Good detection: Low probability of failing to mark real edge points and low probability to falsely mark non-edge points
   - Good localisation: Points marked as edge points by operator should be as close as possible to center of true edge
   - Single response: Only one response to a single edge
2. Process:
   - Perform Gaussian filtering to suppress noise
   - Calculate the gradient magnitude and direction (Sobel or Prewitt)
   - Apply *Non-Maximum Suppression* (NSM) to get a single response for each edge
   - Perform *hysteresis thresholding* to find potential edges

2.2 NMS

1. Algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions
2. Process;
   - Check whether pixel q is local maximum along gradient direction
   - Move to pixel r and compare the gradient magnitudes between q and r
   - Move to pixel p and compare the gradient magnitudes between q and p
   - If pixel q is the local maximum, it is an edge point and suppress other pixels
   - Suppression:

$$M(x, y) = \begin{cases} M(x, y) & \text{if local maximum} \\ 0 & \text{otherwise} \end{cases}$$

3. Optimisations:
   - If pixels p and r are not located in pixel lattice, use nearest neighbour or linear interpolation
   - Round gradient direction into 8 possible angles

2.3 Thresholding

1. Converts magnitude map into a binary image (either edge or not)
2. Hysteresis thresholding - Define two thresholds t_{low} and t_{high}:
   - If pixel gradient magnitude is $\geq t_{high}$ - It is accepted as edge pixel
   - If pixel gradient magnitude is $< t_{low}$ - It is rejected
   - If pxel between - Check neighbouring pixels. Accepted if connected to edge pixel.
   - Repeated until all pixels are accepted or rejected

3 Feature/Interest point detection and description

3.1 Hough Transform

1. Feature extraction algorithm - Obtain a parametric representaiton of a line given **edge points**
2. General idea:
   - Detect some shapes or objects
   - There are two spaces, the image space and the parameter space (Hough space)
     - If the shape can be described by some parameters using an analytical equation, we use this equation to perform voting to the parameter space.

    - If it is not a simple shape without an analytical equation, as long as we have a model to describe it, we can still vote.

3. Line detection process:
   - Initialise the bins $H(\rho, \theta)$ to all zeros
   - For each edge points (x,y):
     - For θ from 0 to π
       * Calculate $\rho = x\cos(\theta) + y\sin(\theta)$
       * Accumulate $H(\rho, \theta) = H(\rho, \theta) + 1$
   - Find (ρ, θ) where $H(\rho, \theta)$ is a local maximum and larger than a threshold

     Thresholding to ensure a few random points would not lead to a line being detected
   - Detected lines are defined by $\rho = x\cos(\theta) + y\sin(\theta)$
4. Circle detection process (radius known):
   - The circle is defined as: $(a-x)^2+(b-y)^2 = r^2$
   - For each edge point (x, y), only need to vote for possible values of (a, b)
5. Circle detection process (radius unknown):
   - 3D parameter space $H(a, b, r)$
   - Set range for radius r
   - For each $r \in [r_{min}, r_{max}]$
     - For each edge point (x, y): Vote for possible values of (a, b) and accumulate $H(a, b, r)$
6. Properties:
   - Robust to noise:
     - Edge map often generated after image smoothing
     - Broken edge points can still vote and contribute to line detection
   - Robust to object occlusion
     - Remaining edge points can still vote and contribute to line detection

3.2 Harris Corner Detector

1. Process:
   - Compute x and y derivatives of an image

$$I_x = G_x * I; \quad I_y = G_y * I$$

     where G is a filter e.g. Sobel filter
   - At each pixel, compute the matrix K

$$K = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

   - Calculate detector response

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$
$$R = \det(K) - k(\text{tr}(K))^2$$

   - Detect interest points which are local maxima and whose response R are above a threshold
2. Properties:
   - Invariant to rotation
   - **Not** invariant to scale.

3.3 Scale-invariant Harris Corner Detector

1. Process:
   - For each σ
     - Perform Gaussian smoothing with σ
     - Calculate x and y derivatives of the smoothed image $I_x(\sigma)$ and $I_y(\sigma)$
     - At each pixel, compute the matrix M

$$M = \sum_{x,y} w(x,y)\sigma^2 \begin{bmatrix} I_x^2(\sigma) & I_x(\sigma)I_y(\sigma) \\ I_x(\sigma)I_y(\sigma) & I_y^2(\sigma) \end{bmatrix}$$

3.4 DoG

1. Gaussian filter:

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

2. Laplacian operator:

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

3. Heat diffusion equation and proof:

$$\begin{aligned}
G(x, y, \sigma) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\
\frac{\partial G}{\partial \sigma} &= \frac{1}{2\pi\sigma^3} \left(\frac{x^2+y^2}{\sigma^2} - 1\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \\
\frac{\partial G}{\partial x} &= -\frac{1}{2\pi\sigma^4} x e^{-\frac{x^2+y^2}{2\sigma^2}} \\
\frac{\partial^2 G}{\partial x^2} &= \frac{1}{2\pi\sigma^4} \left(\frac{x^2}{\sigma^2} - 1\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \\
\frac{\partial^2 G}{\partial y^2} &= \frac{1}{2\pi\sigma^4} \left(\frac{y^2}{\sigma^2} - 1\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \\
\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} &= \frac{1}{2\pi\sigma^4} \left(\frac{x^2+y^2}{\sigma^2} - 2\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \\
\frac{\partial G}{\partial \sigma} &= \sigma \left(\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}\right) = \sigma \nabla^2 G
\end{aligned}$$

4. Relationship between $DoG(x, y, \sigma)$ and scale normalised Laplacian of Gaussian $\sigma^2 \nabla^2 G$

$$DoG(x, y, \sigma) = G(k\sigma) - G(\sigma)$$
$$\frac{\partial G}{\partial \sigma} \approx \frac{G(k\sigma) - G(\sigma)}{k\sigma - \sigma}$$
$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$
$$DoG(x, y, \sigma) = G(k\sigma) - G(\sigma)$$
$$DoG(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

$DoG(x, y, \sigma)$ is proportional to $\sigma^2 \nabla^2 G$ with constant $k - 1$

3.5 Feature description

1. Simple descriptors:
   - Pixel intensity:
     - Simple to use
     - Sensitive to absolute intensity value - Changes under different illuminations
     - Not very discriminative - Single pixel does not represent any local content
   - Patch intensities
     - Performs well if images are of similar intensities and roughly aligned
     - Sensities to absolute intensity value
     - Not rotation invariant
   - Gradient orientation
     - Robust to orientation
     - Not rotation invariant
   - Intensity histogram of patch
     - Robust to rotation and scaling
     - Sensitive to intensity changes

3.6 SIFT

1. Combines histogram and gradient orientation
2. Process:
   - Detection of scale-space extrema
   - Keypoint localisation
   - Orientation assignment
   - Keypoint descriptor

3.7 RANSAC

1. Process:
   - For each iteration:
     - Select random sample points
     - Fit a model
     - Based on the model, check how many points are inliers
     - Terminate after a certain number of iterations or enough inliers have been found

3.8 SURF

1. Only calculates the gradients along horizontal and vertical directions by using the Haar wavelets instead of usign histograms of gradient orientations for describing local features
2. This speeds up evaluation of gradient orientation

3.9 BRIEF

1. In SURF, compare a local region to another and calculate the difference, which is a floating point number
2. In BRIEF, we compare a point p to another point q and get a binary value as output.
3. Assumptions:
   - Ignore rotation and scale invariances - assumes images are taken from a moving camera that only involves translation

4 Image classification, detection and segmentation

4.1 HOG

- Extend feature description to a whole image or image region
- Divides a large region into a dense grid of cells, describes each cell, then concatenate these local descriptions to form a global description
- In comparision to SIFT:
  - Both use gradient orientation histograms for feature description
  - Difference is that HOG describes features for a large image region, instead of just around a point.
- Process:
  - Divide the image into equally spaced cells
  - Describe the content there using orientation histograms
  - Move to next block
  - For each block, the descriptor vector v is normalised

- HOG descriptor formed by concatenating the normalised local descriptors for all blocks

4.2 Image classification

- Pipeline:
  - Feature extraction - Transform input images into low-dimensional vectors to be compared or matched
  - Classifier - Trained with a dataset consisting of images and labels
- Pre-processing operations:
  - Detecting where feature is in bigger image
  - Normalising size of image
  - Normalise the location, placing the mass center at the center of image
  - Slant correction

4.3 Neural networks

1. Exploding gradient problem:

   The gradient becomes very large, preventing the algorithm from converging. The solution is gradient clipping either by norm or value.

2. Vanishing gradient problem:

   The sigmoid function suffers from the vanishing gradient problem. When $f(x)$ saturates at either 0 or 1, its derivative $f'(x)$ becomes nearly 0.

3. Precision and recall:

$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{Precision} = \frac{TP}{TP + FP}$$

4. ReLU problem:
   - For negative x, the gradient is 0 which means nothing changes during gradient descent. Solution is Leaky ReLU.

5 Motion estimation

5.1 Optic flow

1. Over determined: More equations than unknowns which is estimated using least square method
$$x = \text{argmin}_x ||Ax - b||^2$$

2. Least square solution given by:
$$x = (A^T A)^{-1} A^T b$$
$$x = \begin{bmatrix} u \\ v \end{bmatrix}$$
$$A^T A = \sum_p \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
$$A^T b = -\sum_p \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix}$$

3. Optic flow assumptions:
   - Brightness constancy: A pixel has constant brightness across time.
   - Small motion: Between frames, motion is small.
   - Spatial coherence: Pixels move like their neighbours i.e. flow is constant within a small neighbourhood.

4. Constraints:
   - Brightness constancy assumption:
$$I(x + u, y + v, t + 1) = I(x, y, t)$$
     where I: Intensity, (x, y, t): Spatial and temporal coordinates, (u, v): Displacement
   - Small motion assumption:
$$I(x+u, y+v, t+1) \approx I(x, y, t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}$$
   - Optical flow constraint equation (Combining both):
$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

5.2 Lucas-Kanade method

1. Spatial coherence assumption: Flow is constant within a small neighbourhood
2. Aperture problem:
   - Looking through an aperture (hole), the motion of a line is ambiguous, because the motion component parallel to the line cannot be inferred based on the visual input
   - Motion of a corner is clearer to define, even through the aperture
3. Process:
   - Compute the image gradient I_x, I_y (finite difference between neighbouring pixels) and I_t (finite difference between neighbouring time frames)
   - For each pixel:

- Calculate the following matrix for pixels in its neighbourhood

$$A^tA = \sum_p \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

- Calculate the optic flow for this pixel

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^tA)^{-1}A^Tb$$

5.3 Multi-scale framework

1. Suppose an estimate of the flow field $u^{(4)}, v^{(4)}$ for image I and J is obtained at scale 4.
2. When moving to scale 3, the flow field becomes $2u^{(4)}, 2v^{(4)}$. This will provide the initial values of the calculation at scale 3.
3. Only need to calculate the incremental flow for the following two images:
   - Image $I(x, y)$
   - Image $J_{wraped} = J(x + 2u^{(4)}), y + 2v^{(4)}$
4. Accumulate the flow estimation

5.4 Multi-scale Lucas-Kanade method

1. To estimate large motion, multi-scale/multi-resolution framework is used

   Although the motion is large in the original resolution, it will look small in a downsampled resolution.
2. Process:
   - For scale l from coarse to fine
     - Initial guess at scale l by upsampling the flow estimate previous scale $l + 1$

       $$u^{(l)} = 2u^{(l+1)}, v^{(l)} = 2v^{(l+1)}$$

     - Compute the wraped image

       $$J^l_{wraped} = J^l(x + u^{(l)}), y + v^{(l)}$$

     - For image I^l and J^l_{wraped} at this scale, compute image gradients I_x, I_y and I_t
       * I_x, I_y are calculated from image I^l
       * $I_t = J^l_{wraped}(x, y) - I^l(x, y)$
     - Estimate the incremental flow

       $$\begin{bmatrix} u^\delta \\ v^\delta \end{bmatrix} = (A^TA)^{-1}A^Tb$$

     - Update the flow at this scale

       $$u^{(l)} = 2u^{(l+1)} + u^\delta, v^{(l)} = 2v^{(l+1)} + v^\delta$$

5.5 Horn-Schunck method

1. It is an optimisation-based method, which defines a global energy functional (i.e. a cost function) for the flow.
2. The Horn-Schunck method optimises the energy functional with regard to u and v
3. It aims to enforce the optic flow constraint, while at the same time achieves a smooth flow field
4. Process:
   - Compute the image gradients I_x, I_y and I_t
   - Intialise the flow field $u = 0, v = 0$
   - For each iteration k:
     - Calculate the average flow field

       $$\overline{u}^{(k)}, \overline{v}^{(k)}$$

     - Update the flow field

       $$u^{(k+1)} = \overline{u}^{(k)} - \frac{I_x(I_x\overline{u}^{(k)} + I_y\overline{v}^{(k)} + I_t)}{I_x^2 + I_y^2 + \alpha}$$
       $$v^{(k+1)} = \overline{v}^{(k)} - \frac{I_y(I_x\overline{u}^{(k)} + I_x\overline{v}^{(k)} + I_t)}{I_x^2 + I_y^2 + \alpha}$$

     - Terminate if the change of value is smaller than a threshold or the maximum number of iterations is reached.

5.6 Flow evaluation

- Image classification: manually annotate label class
- Object detection: manually annotate label class and bounding box
- Segmentation: manually annotate label class for each pixel
- Optic flow methods:
  - Generate the ground truth flow field between two time frames - Using external data or generating synthetic data
  - Compare the difference between estimation and ground truth.
    * Average endpoint error (EE)

6 Motion tracking

1. Spatial and temporal image gradients:

$$I_x = \frac{I(x+1, y, t) - I(x-1, y, t)}{2}$$
$$I_y = \frac{I(x, y+1, t) - I(x, y-1, t)}{2}$$
$$I_t = I(x, y, t+1) - I(x, y, t)$$

Lucas-Kanade tracker

1. Lucas-Kanade aims to estimate the motion from the template image I to the image J in the next time frame
2. With brightness constancy constraint, optimisation problem

   $\min_{u,v} E(u,v) = \sum_x \sum_y [I(x,y) - J(x+u, y+v)]^2$