Image Filtering (1) identity filter (2) low-pass or smoothing filters, removes high frequency signal (noise/sharpness) a) moving average filter b) Gaussian filter (3) High-pass or sharpening filters = identity + (1-moving average) (4) denoising, median filter. Processing an image with a filter is done by convoluting the image with the kernel. To convolute 2 matrices, flip the kernel along x and y axis, and scan over each pixel with the kernel. Element wise multiply and sum all values given new value for the centre.

Edge detection Refers to lines where image brightness changes sharply and has discontinuities. Color/depth/surface normal discontinuity. Provides important low-level features, where discontinuities occur, for both animal version and CV for analysing and understanding images. Derivatives characterize the discontinuities of a function. forward $f'[x] = f[x+1] - f[x]$ backward $f'[x] = f[x] - f[x-1]$ central $f'[x] = \frac{f[x+1]-f[x-1]}{2}$ $\begin{bmatrix}1\\1\\1\end{bmatrix} * [1\ 0\ -1]$ **Prewitt/Sobel filter**, moving average + finite difference. Image gradient $g_x = \frac{\partial f}{\partial x}$ Smoothing: suppress the noise, derivatives sensitive to noise. Gaussian kernel $h[x] = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$. $h[x,y] = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$. σ large, large-scale edges and smooths out fine details; small, detect fine features. **Canny edge detection.** <u>1.</u> perform GF to suppress noise. <u>2.</u> calculate gradient magnitude and direction. Used in NMS, it thins the detected edges to keep only the local max in g.m. along the direction of g.o. If it is not max along its neighbours, it is suppressed and set to 0. <u>3.</u> apply non-maximum suppression (**NMS**) to get a single response for each edge. (M(x,y)=M(x,y)if local maximum, 0 otherwise. Perform image interpolation or use 8 angles to round gradient direction.)<u>4.</u> perform **hysteresis thresholding** to find potential edges. Simple thresholding: binary(x,y)=1 if I(x,y)>=t, 0 otherwise. hysteresis, <low, rejected, between, weak edge pixel, accept when connect to edge pixel>=high, accepted edge pixel. **Learning based edge detection**: AL algo assumes paired data (image x, manual edge map y). find $\theta\ s.t.\ y = f(x|\theta)$. integrates from multiple scales with coarse-scale edges to form a final output, while Canny Edge detection uses single scale σ.

Hough transform <u>idea</u>: each edge point <u>votes</u> for possible parameters in the parameter space. <u>algo</u>. Initialize the bins $H(\rho,\theta)$to all zeros. For each edge point (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 (a few random points would not lead to a detected line). Detected lines: $\rho = x\cos\theta + y\sin\theta$. For circle, $a = x - r\cos\theta, b = y - r\sin\theta$, for each $r \in [r_{min}, r_{max}]$. advs. detect multiple lines. robust to noise, object occlusion. cons: CC is high, carefully design parameters (param for edge detector, threshold for accumulator, range of circle radius). **Hough forests**, predicts a displacement vector from the patch centre, given the image feature of the patch.

Interest Point Detection. (used in classification, matching, retrieval) **Harris detector** finds interest points with strong responses at corners and blobs where local structure rich. Calculate changes of intensities within a small window when it is shifted in the image, which can be qualified by local image derivatives. it is rotation-invariant (same change of intensities when shift the window along rotated direction), not scale. $E(u,v) = \sum_{(x,y)\in W} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \begin{bmatrix}u\\v\end{bmatrix} \sum w(x,y) \begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix} \begin{bmatrix}u\\v\end{bmatrix}$, $M = \sum_{x,y} w(x,y) \begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix} = P\Lambda P^T$. flat $\begin{bmatrix}0&0\\0&0\end{bmatrix}$ edge $\begin{bmatrix}10&0\\0&0.1\end{bmatrix}$ corner $\begin{bmatrix}10&0\\0&10\end{bmatrix}$ Cornerness $R = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2 = \det(M) - k\,trace(M)^2$. <u>Algo</u>. compute I_x=G_x*I,I_y. at each pixel, G can be Sobel. Compute M, get detector response R, detect interest points which are local maxima and r above threshold. Change along both directions: corner, blob, texture. **Scale adapted Harris detector**. apply HD at multiple scales with G with different σ and sampling with different spatial resolutions. σ larger, magnitude of derivative smaller. $M = \sum_{x,y} w(x,y)\sigma^2 \begin{bmatrix}I_x^2(\sigma) & I_x(\sigma)I_y(\sigma)\\ I_x(\sigma)I_y(\sigma) & I_y^2(\sigma)\end{bmatrix}$. <u>Algo:</u> calculate r from small-scale to large-scale, find local maxima both across space and scale. **LoG** (detector) G smoothing, followed by Laplacian. Laplacian is the sum of second $\begin{bmatrix}0&0&0\\1&-2&1\\0&0&0\end{bmatrix} + \begin{bmatrix}0&1&0\\0&-2&0\\0&1&0\end{bmatrix} = \begin{bmatrix}0&1&0\\1&4&1\\0&1&0\end{bmatrix}$ derivatives. $\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$. Response $LoG_{norm}(x,y,\sigma) = \sigma^2(I_{xx}(x,y,\sigma) + I_{yy}(x,y,\sigma))$ **DoG** response$DoG_{norm}(x,y,\sigma) = I * G(k\sigma) - I * G(\sigma)$. Convenience in calculating response across different scales.

feature description. Extract local features near an interest point. Consider robustness to rotation, scaling and intensity changes. Matching keypoints, estimate spatial correspondence. **Simple descriptors**: **pixel intensity** (sensitive to absolute intensity value, not discriminative, no local content), **patch intensities** (have local pattern, 表现好 similar intensities and roughly aligned [1,2]. sensitive, not-rotation-invariant), **gradient orientation**(无法旋转), **histogram** （可旋转扩大,sensitive to intensity changes）. **SIFT descriptor**(可旋转扩大+robust to intensity changes). <u>idea</u>: transforms an image into a large set of interest points, each of which is described by a feature vector that is invariant to scaling and rotation and change of illuminations. <u>steps:</u> **1.** <u>detection of scale-space extreme</u>. (DoG to identify potential interest points, use both extreme, 255 negative minima and 0 positive maxima) **2.** <u>keypoint localization.</u> (a <u>quadratic function</u> is fitted to the DoG response of neighbouring pixels, the location and scale of extremum for this quadratic function can be estimated to refine estimate $D(x+\Delta x) = D(x) + \frac{\partial D^T}{\partial x}\Delta x + \frac{1}{2}\Delta x^T \frac{\partial^2 D}{\partial x^2}\Delta x$, $\Delta x = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}$, move from x by Δx to arrive at refined estimate. location: sub-pixel accuracy, scale: some value between $\sigma, \sqrt{2}\sigma, 2\sigma$) **3.** <u>**orientation assignment**</u>. (Attempt to assign consistent orientation to each keypoint. we need to know the orientation of the feature, then we can perform a sampling in a rotated coordinate system when we calculate features. we can create the orientation histogram with 36 bins of 10°. each pixel in neighbor votes for an orientation bin, weighted by the gradient magnitude. the keypoint will be assigned an orientation according to maximum of the histogram. now we have $(x,y,\sigma), and\ \theta$. we can draw samples using window size prop to σ rotated by θ. <u>two uses,</u> draw points relative to this orientation, and when calculate SIFT features, go of each point is relative to the g.o. sampling is performed in a rotated coordinate system. g.o of each point needs to be adjusted by θ. **4.** <u>keypoint descriptor.</u> (Form a histogram) 16 subregions, each calculate an orientation histogram with 8 bins, each bin counts the sum of gradient magnitude for this orientation. (Length of arrow in histogram denotes the sum of gm.) Dimension of the descriptor: 16x8=128. To make it scale and rotation invariant, scale is known when detect the keypoint. dominant orientation. (<u>Sum.</u> <u>consideration of dominant orientation + sampling window proportional to scale + use of g.o for feature description + use of histograms</u>).

keypoint matching by identifying the nearest neighbours, for each keypoint in A, identify its nearest neighbours in the set of keypoints in B with Euclidean distance. $\begin{bmatrix}u\\v\end{bmatrix} = \begin{bmatrix}m1 & m2\\ m3 & m4\end{bmatrix} \cdot \begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}t_x\\t_y\end{bmatrix}$. least-square solution to Am=b is $m = (A^TA)^{-1}A^Tb$, minimize $\overset{A: 2N\times 6\ \ b: 2N\times 1}{||Am - b||^2}$. Once the affine parameters are solved, we know the spatial transformation between two images and matching is done. Outliers, Random Sample Consensus **RANSAC** task: Find the best fitting line to the points. Randomly sample some points, fit a model. Check out how many inliers found within a threshold. Repeat until reaches max iteration or enough inliers are found. By matching keypoints and solving the linear system, we can estimate the transformation between the images and stitch them to create a panorama. cons: calculating is slow in real-time performance. **speed-up**: use faster hardware e.g., on field programmable gate array (FPGA), much faster than CPU, optimize software implementation, develop a new algorithm.

SURF (speeded-up robust features), uses histograms of gradient orientation for describing local features, only calculates the gradient along the horizontal and vertical directions with Haar wavelets (evaluate image gradient with weight +1 or -1, which is fast. dx, -1 | 1, dy $\frac{-1}{1}$. For each subregion, sum up the Haar wavelet response over the sample points. the descriptor for the subregion is defined by 4 elements.($\sum d_x, \sum d_y, \sum|d_x|, \sum|d_y|$. homogeneous region, all values are low; zebra pattern, only $\sum|d_x|$is high; gradually increasing intensities, $\sum d_x\ and\ \sum|d_x|$are high. dimension=16x4, 5 times faster than SIFT. (Each dimension is a floating-point number or 4 bytes). **speed-up**: <u>1. shorten the descriptor</u>: quantization (连续值转化为 discrete number, e.g. 0 to 255, 8 bits). binarization (0 or 1, 2 bits). <u>2. distances other than Euclidean</u>. **BRIEF** (Binary Robust Independent Elementary Features). $\tau(p,q) = 1\ if\ I(p) < I(q), 0\ otherwise$. Randomly sample n_d pairs of points for binary tests, each test gives 1 bit (0 or 1). BRIEF descriptor: n_d-dimensional bitstring. If $n_d = 256$, 256 bits or 32 bytes, shorter than SIFT (512 bytes) or SURF (256 bytes). much faster: only compare two numbers without calculating the g.o. in SIFT or intensity difference in SURF. When comparing two descriptors, also faster because no Euclidean distance, but use Hamming distance (XOR). <u>cons:</u> ignore rotation and scale invariance, only involve translation. <u>pros:</u> 40 倍 over SURF, 200 倍 over SIFT. <u>Other acceleration:</u> google TPU for ML, CUDA supports 16-bit floating point format, FPGA for high frequency trading, Neural network pruning, weight quantisation.

之前的 extract local features near an interest point. vs. **HOG** (describe an image) by gradient orientation histogram. <u>idea</u>: divides a large region into a dense grid of cells, describes each cell, then concatenates these local descriptions to form a global description. <u>steps:</u> divide the image into equally spaces cells, each cell 8x8 pixels, 4 cells form a block. calculate g.o. histogram for this block. Move to the next block. Use normalized to reduce <u>overlap</u>. $v_{norm} = \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}}$. HOG descriptor is formed by concatenating the normalized local descriptors for all blocks. not rotation-invariant.

Image classification feature extraction: hand-crafted (e.g., pixel intensities, HOG), learnt features (e.g., CNN, auto learnt by algorithm). **Classifier**: **KNN** (compute K nearest neighbours and test data point is assigned the class given by majority voting, Euclidean distance, <u>pros:</u> no training step, simple but effective, multi-class classification, <u>cons:</u> storage and search are expensive); support vector machine (HOG+**SVM**) (linear SVM, a line that separates two different classes. wx+b=0, c=+1 if wx+b >0, -1 otherwise. (Maximum margin hyperplane) the innermost points, or <u>support vectors</u>, to fulfil equations wx+b=+1 or -1. $w \cdot x_1 + b = -1, w \cdot (x_1 + mn) + b = 1, then\ mw \cdot n = 2. since\ n = \frac{w}{||w||}$, margin $m = \frac{2}{||w||}$, optimisation $min||w||^2 + C\sum_{i=1}^N \max(0, 1 - y_i(w \cdot x_i + b))$, which is loss + hinge loss. Gradient $\nabla_w L = 2w + C\sum_{i=1}^N \nabla_w h, when\ h = \max hinge\ loss, \nabla_w h = -y_ix_i, if\ y_i(w \cdot x_i + b) < 1, 0\ otherwise$. Optimize loss f. using <u>gradient descent</u>. SVM classifier $w \cdot x + b = 0$, x is derived by HOG, w and b are trained using gradient decent. SVM only separates 2 classes. **one vs. rest strategy**, 1 and others, 2 and others, etc. K classifiers, who produces the highest response is the result. **one vs. one**, K(K-1) classifiers, 1 and 2, 1 and 3, etc. Count the vote for each class and perform majority voting. Neural network, (multi-layer perceptron **MLP**, is formed by putting several layers of neurons into connection, where the output of a neuron can be the input to another. **neuron** is a computational unit that takes an input, applies an activation function, and generates an output. **sigmoid (activation) (or logistic) function** $f(z) = \frac{1}{1+e^{-z}}$. loss function $J(W,b) = \frac{1}{M}\sum_{m=1}^M \frac{1}{2}||a_m - y_m||^2$. (x, y) are paired data and ground truth labels, for network a, find W and b. Use <u>forward propagation</u> to calculate the output of network a, given input x. Use <u>backpropagation</u> to calculate the gradient and then perform <u>stochastic</u> <u>gradient descent</u> for optimisation to find W and b that minimize the loss function so that a matching ground truth y as much as possible. For image classification, where y is a categorical variable, we define the loss function as the distance between the predicted probability and the true probability, and cross entropy. $H(p,q) = -\sum_i p_i \log(q_i)$. For binary classification, use sigmoid. For multi-class, use **softmax function** $f(z_i) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$. true probability p = [0, …, 1,…,0] **one-hot encoding**. $a = [f(z_1), \ldots,]$ cross entropy loss $J(W,b;x,y) = H(p,q) = -\sum_{i=1}^K y_i \log(f z_i))$. softmax generalizes the sigmoid function to multiple classes. error rate = 100% - classification accuracy. **affine transformation**: translation, scaling, squeezing, shearing. data augmentation improves classification performance. **MLP cons**: uses too many parameters, may not scale up to bigger images, e.g., ImageNet images. Too many params because 2D image is considered as a flattened vector, without considering its 2D nature. **CNN** <u>assume</u> input are images and encode certain properties (local connectivity, weight sharing) into the architecture, which make the computation more efficient and substantially reduce the number of parameters. Each neuron only sees a small local region in the layer before it, which is <u>receptive field</u>. n neurons, $a_n = f(\sum_{ijk} W_{nijk}x_{ijk} + b_n)$. Move the small window across the input image and get an output cube of size XxYxD, each window shares the same weight, e.g., number of parameters is 5x5xCxD for connection weights and D for bias. (n, or d, is output depth, i is kernel width, j is kernel height, k in input depth). **operations**: padding (affects output size), stride (downsamples the image), dilation (increase the receptive field on the original image without increasing the number of parameters). pooling layer: another building block apart from convolutional layer. make feature maps smaller, neurons in the next layer can see a larger region of image. e.g., max pooling. LeNet-5, I, C1, S2, C3, S4, C5, F6, O. **Deep neural network, success** because: hardware (efficient use of graphical processing units GPUs), Data (large datasets, ImageNet), Algorithm (Improvement of optimisation: ReLU, network architecture: AlexNet, VGG, ResNet, Transformer). optimisation: stochastic gradient descent, $W = W - \alpha\frac{\partial L_B}{\partial W}, b = b - \alpha\frac{\partial L_B}{\partial b}$. <u>problems:</u> <u>1. exploding gradient</u>: the gradient becomes very large, preventing the algorithm from converging) solve by **gradient clipping**, clip by value: $g_i = v_{min}\ if\ g_i < v_{min}, v_{max}\ if\ g_i > v_{max}, g_i$. clip by norm, $g_i = \frac{g}{||g||}v\ if\ ||g|| > v$, g otherwise <u>2. vanishing gradient</u>: the gradient becomes vanishingly small, preventing the weights from changing their values. **ReLU,** $f(z) = 0\ if\ z < 0, z\ if\ z \geq 0. f'(z) = 0, z < 0; 1, z \geq 0$. the gradient will not vanish when z is very large. Leaky ReLU: $f(z) = 0.01z\ if\ z < 0, z\ if\ z \geq 0$. Parametric ReLU (PReLU), $f(z) = az\ if\ z < 0, z\ if\ z \geq 0$. Exponential linear unit (ELU) $f(z) = a(e^z - 1)\ if\ z < 0, z\ if\ z \geq 0$. AlexNet, use ReLU, data augmentation (randomly crop 224x224 regions from 256x256 original images, horizontal reflection, perturb RGB intensities), training on multiple GPUs, 5Conv layers+3FC layers.

Vision transformer. Images have quite different format from language (pixels vs words). Divide picture to patches. Linear projection of flattened patches. Get features and fed to Transformer Encoder. Then to MLP Head, which is a multi-layer perceptron (i.e. fully connected layers) to predict class. The transformer consists of multi-head self-attention (MSA) layers.

Image segmentation (Pixel-wise classification) **unsupervised**, thresholding, K-means clustering, (parameters: assume each cluster is represented by its centre; association: each data point is assigned to the nearest cluster centre, $min\sum_{k=1}^K \sum_x \delta_{x,k}(x - \mu_k)^2$). <u>algo:</u> Initialize the cluster centre μ_k, for each iteration: compute $\delta_{x,k}$ for each data point, assigning x to the nearest cluster centre μ_k. update μ_k according to the membership $\delta_{x,k}$. repeat until $\delta_{x,k}$ no longer changes or the maximum number of iterations is reached. Feature can be more than just intensities. Clustering can be performed by colour similarity, position + color similarity, other features. x becomes a feature vector instead of a scalar. <u>cons:</u> performs a hard assignment, $\delta_{x,k}$ is either 0 or 1. GMM Gaussian mixture model, soft assignment by assuming a Gaussian distribution for each cluster. $P(y_j = k|x_j, \pi_k, \mu_k, \sigma_k) = \pi_k \cdot \frac{1}{\sqrt{2\pi}\sigma_k}\exp(-\frac{(x-\mu_k)^2}{2\sigma_k^2})$. <u>algo:</u> Initialize the parameters π_k, μ_k, σ_k, for each iteration, compute P, update π_k, μ_k, σ_k, repeat until… assign class $c = argmax\ P(y_j = c|x_j, \pi, \mu, \sigma)$. **PPT background removal**, GrabCut <u>algo:</u> estimate GMM parameters for two clusters (foreground and background), perform segmentation with smoothness constraint and user edit constraint. |**Supervised** segmentation, CNN Expensive to apply the network multiple times (the output is a probability vector) to all pixels. We want an output a pixel-wise probability map, after the last convolutional layer. All layers are convolutional layers, it is a **fully convolutional network**. Recover the probability map to the original image size by umsampling (transposed convolution). $\begin{bmatrix}w_1 & w_2 & w_3 & 0\\ 0 & w_1 & w_2 & w_3\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix} = \begin{bmatrix}y_1\\y_2\end{bmatrix}, \begin{bmatrix}w_1 & 0\\ w_2 & w_1\\ w_3 & w_2\\ 0 & w_3\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix}$. (Removing the fully connected layers, adding convolutional and transposed convolutional layers) train: define a classification loss at each pixel, e.g., **cross entropy**, the segmentation loss is the average classification loss for all pixels, by optimising the segmentation loss, train the network. The following integrate multi-scale features, instead of only local and global features: **FCN** (4 0.5x, 1 16x) other can implement multi-scale features: **DeepLabv3+** (4 0.5x, 1spatial pyramid pooling, 4x, **skip connection** concatenates local with global features 4x) (**image pyramid**, representation of an image at multiple scales, extract features at multiple scales), **U-net** (5 0.5x, 5 2x, skip connection at each level). **recent advances**: **HRNet** (4 resolution levels of features, high-resolution (local) features contribute to the learning of low-resolution (global) features. vice-versa)) **UNet++** (features of multiple scales are fused multiple times)

UNETR (Vision Transformer can learn features for image patches. Aggregate local features learnt by convolutions and global features learnt by transformer). **Semantic segmentation**, assigns a class label to each pixel, does not care whether these pixels form a single object. **Instance segmentation**, aims to define segmentation for each instance, focuses on countable things, ignores uncountable stuff, sky, road. **Panoptic segmentation**, unify things and stuff, providing segmentation for each thing (instance) and segmentation for stuff. **How to train segmentation**? It's like pixel-wise classification. At each pixel, define a classification loss (cross entropy). Segmentation loss is the average classification loss for all pixels. Train the network by optimising the segmentation loss.

object detection. (Apply classification model at different regions of an image for detection.) (Predict a set of bounding boxes and labels for each object of interest). 2 tasks: classification, and localization (predict bounding box coordinates) idea: Slide a window across the feature map. Utilise convolutional features for classification and localization. one-stage vs two-stage detection methods. tradeoff between accuracy and speed. **Two-stage object detection, Faster R-CNN stage 1** a region proposal network (RPN) is applied to convolutional feature map to propose possible regions of interest. Perform binary classification (0 not interesting, 1 otherwise). Predict the object size (x,y,w,h):(location, width and height). **Anchors** (boxes of varying sizes) **aspect ratio**. loss $L(p,t)=\sum_{i=1}^{n_{anchor}} L_{cls}(p_i,p_j^*)+\lambda\cdot\sum_{i=1}^{n_{anchor}} p_i^*\cdot L_{loc}(t_i,t_i^*)$, classification loss + localization loss **stage 2** a detection network, use features in ROI (proposed region of interest). ROI pooling, each various sizes, normalizes the size, mapping features to standard fixed size, and then provide to classifier. For each ROI, multi-class classifier predicts its label class and refine the bounding box estimate. $L(p,t)=L_{cls}(p,y)+\lambda\cdot 1_{y\geq 1}L_{loc}(t,t^*)$. **PRN vs Detection network**. input 3x3 window on convolutional feature map / a proposed region, thus contains more accurate features. RPN class-agnostic, only checks whether it is a ROI. / DN classifies the region into several classes. RPN use many anchors because 不知道物体看起来什么样 / DN already know a rough size from the proposal network, no need anchors. RPN move 3x3 sliding window, classifies it to be something interesting (binary classification) and proposes a region, DN takes the region's feature pools it into a fixed size window, classifies it with multi-class classifier, refines the bounding box. **One-stage object detection**, YOLO, **SSD** (Single shot multibox detector). change ground truth p_j^* binary into multi-class. Less accurate because feature map always 3x3 while 2-stage: feature map better matches the object size.

Recent Classification. 结构 **VGG** Visual Geometry Group (deeper, only uses 3x3 layers, which increases the depth of the network and non-linearity, training based on error backpropagation layer by layer, but difficult for error to propagate to layers far from output and update weights there.), **ResNet** (add shortcut connection to enable flow of gradients, formed by stacking some residual unit onto a shallow network), **Vision Transformer**, **Self-attention**: a way to model the dependencies between the tokens. $Attention(Q,K,V)=softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$. For n patches, dimension of Q, K, V is (dxn), $O(d^2n) or\ O(n^2d)$. **ConvNeXt** 训练 **self-supervised learning** (contrastive learning, mased auto-encoding) **data** modalities: **vision-language modelling**

Motion. **optic flow** is the motion of brightness patterns in videos. Output is a dense motion field, which describes the displacement vector for each pixel in the image. **3 assumptions**: brightness constancy (a pixel has constant brightness across time.) small motion (between frames, motion is small) spatial coherence (pixels move like their neighbours). $I(x+u,y+v,t+1)=\mathrm{I(x,y,t)}, also \approx \mathrm{I(x,y,t)}+\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v+\frac{\partial I}{\partial t}$. Combining, $I_xu + I_yv + I_t = 0$. (**Optic flow constraint equation**). Use spatial coherence (Flow is constant within a small neighbourhood) to solve from underdetermined to over-determined. $[I_x(p)\ I_y(p)]\begin{bmatrix}u\\v\end{bmatrix}=-I_t(p), x=$ estimate unknown using the least square method $argmin||Ax-b||^2, x=(A^TA)^{-1}A^Tb, x=\begin{bmatrix}u\\v\end{bmatrix}, A^TA=\sum_p\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix}(also\ seen\ in\ Harris\ detector), A^Tb=-\sum_p\begin{bmatrix}I_x & I_t\\ I_y & I_t\end{bmatrix}, cond(A^TA)=\frac{|\lambda_{max}|}{|\lambda_{min}|}$. For flat or edges, λ_{min} is close to 0, condition number is large, calculating $(A^TA)^{-1}$ becomes numerically sensitive to small perturbations, so corner easier to track than edge. **Aperture problem** 由于观察者的视角有限,我们可能无法获得完整的信息,从而限制了对物体运动或边界方向的精确估计(motion of a line is ambiguous because the motion component parallel to the line cannot be referred based on the visual input). **Lucas-Kanade method**: 1. compute the image gradients I_x, I_y and I_t. sum over a small neighbour 2. For each pixel, calculate A^TA, A^Tb and $\begin{bmatrix}u\\v\end{bmatrix}$. Image pyramid (a multi-scale representation of an image). The motion is large in the origin resolution but is small in the downsampled resolution. So Multi-scale Lucas-Kanade method can address the challenge of estimating large displacements. Coarse-to-fine motion estimation. The final flow field is obtained by summing up all incremental flows. steps:1. for scale l from coarse to fine, initial guess at scale l by upsampling the flow estimate at previous scale l+1 $u^{(l)}=2u^{(l+1)}, v^{(l)}=2v^{(l+1)}$. Compute the warped image $J^l_{warped}=J^l(x+u^{(l)},y+v^{(l)})$. For image I^l and J^l_{warped}, compute I_x, I_y, I_t. $I_t=J^l_{warped}-I^l$. Estimate the incremental flow using $\begin{bmatrix}u^\delta\\v^\delta\end{bmatrix}=(A^TA)^{-1}A^Tb$. Update the flow at this scale: $u^{(l)}=2u^{(l+1)}+u^\delta$.(same for v) motion for each pixel → an object of interest **Object tracking** 1. **Lucas-Kanade tracker** (estimate motion (u,v)(defined for one sum over pixels within template image object) from the template image I to the image J in the next time frame. Optimisation: cost function $min_{u,v}E(u,v)=\sum_x\sum_y[I(x,y)=J(x+u,y+v)]^2.\begin{bmatrix}u\\v\end{bmatrix}=-(\sum_x\sum_y\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix})^{-1}\sum_x\sum_y\begin{bmatrix}I_x & I_t\\ I_y & I_t\end{bmatrix}$. Sum over pixels within template image instead of over a small neighbourhood. cons: brightness constancy assumption may not always hold, only used pixel intensities, does not learn discriminative features for the template. 2. **Correlation filter method**, $(f*h)[m]=\sum_{m=-\infty}^{\infty}f[m]h[n+m]$utlise features extracted by CN. Aim to maximise correlation between template features and image features in the next time frame. Correlation more robust to illumination changes than sum of squared difference. For 2D, find correlations maximum between template features and features in search window. Features are learnt from CNNs. Correlation filter: **Siamese network**: comparing a template (representation) of the object to be tracked with candidate regions in subsequent frames. The network parameters are trained to predict ground truth score map(needs define). the maximal score indicates the location of the object. φ is used to extract features from both images and then a cross relation * is applied to find the similarity between the two images. If the similarity is above the threshold, then it implies the compared object is the same, thus this object is tracked.

camera model pinhole camera, homogeneous coordinate $(X,Y,Z,1)\to(\frac{fX}{Z},\frac{fY}{Z},1)$. the mapping for a pinhole camera is a perspective projection. 3D to 2D: $\begin{bmatrix}fX\\fY\\Z\end{bmatrix}=\begin{bmatrix}f000\\0f00\\0010\end{bmatrix}\cdot\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$. image coordinate may differ from origin from the principal point p (principal point offset). So we have camera to image: $\begin{bmatrix}fX+p_xZ\\fY+p_yZ\\Z\end{bmatrix}=\begin{bmatrix}f\ 0\ p_x\ 0\\0\ f\ p_y\ 0\\0\ 0\ 1\ 0\end{bmatrix}\cdot\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$. world to camera to image $x=\begin{bmatrix}f&0&px&0\\0&f&py&0\\0&0&1&0\end{bmatrix}\cdot\begin{bmatrix}R&-RC\\0&1\end{bmatrix}\cdot X$ camera matrix $P=\begin{bmatrix}f&0&px\\0&f&py\\0&0&1\end{bmatrix}\cdot[R\quad -RC]$. 9 degrees-of-freedom (3 intrinsic, 6 extrinsic (3 R 3 C)). pixel on camera sensors, $P=\begin{bmatrix}ax&0&px\\0&ay&py\\0&0&1\end{bmatrix}\cdot[R\quad -RC]$. 10 DoF. skew, $P=\begin{bmatrix}ax&s&px\\0&ay&py\\0&0&1\end{bmatrix}\cdot[R\quad -RC]$. 11 DoF=5i6e. $X_{Cam}=X_{world}-C\ or\ R(X-C)$ **camera calibration**, given $\{X_i,x_i\}$, x=PX, estimate P. steps: Analyse the unknowns and equations. Write down an equation system and solve it. Formulate it as an optimisation problem and solve it. $\begin{bmatrix}X^T&0&-X^Tx\\0&X^T&-X^Ty\end{bmatrix}\begin{bmatrix}p1\\p2\\p3\end{bmatrix}=0, optimisation: p=argmin||Ap||^2, ||p||^2=1, perform\ \boldsymbol{SVD}\ to\ A: A=U\Sigma V^T$. (Singular Value Decomposition, p is the column of V that corresponds to the smallest singular value. Rearrange p's element to form P) Use **reference object** to get $\{X_i,x_i\}$. (Place a reference object in the scene, identify correspondences between points in the image and in the scene. Compute the mapping from scene to image).

tutorials:

Gaussian white noise, $Y_i=I+n_i$. the mean and variance of the noise in image Y: noise in combine image $e=Y-I=\frac{1}{N}\sum_{i=1}^N Y_i-I=\frac{1}{N}\sum_{i=1}^N(I+n_i)-I=\frac{1}{N}\sum_{i=1}^N n_i$ so $E[e]=\frac{1}{N}\sum_{i=1}^N E[n_i]=0. Var(X)=E(X^2)-E^2(X)$ so variance $var[e]=E[(e-E[e])^2]=E[e^2]=E\left[\left(\frac{1}{N}\sum_{i=1}^N n^i\right)^2\right]=\frac{1}{N^2}E[\sum_{i=1}^N n_i^2+\sum_{i=1}^N\sum_{j\neq i}n_in_j]=\frac{1}{N}\sigma^2$ **loss function of sum of squared difference**, $E(\beta)=Y^TY-2\beta^TX^TY+\beta^TX^TX\beta, \frac{\partial E}{\partial\beta}=-2X^TY+2X^TX\beta, let\ it=0, \beta=(X^TX)^{-1}X^TY. since\ \frac{\partial^2E}{\partial\beta^2}=2X^TX\ is\ a\ positive\ matrix, therefore\ it\ is\ a\ local\ minimum.$ It is also how to solve over-determined system in the form of $\min||Am-b||^2, m=(A^TA)^{-1}A^Tb$ **proof det and trace**: $\det(M)=\det(P\Lambda P^T)=\det(P)\det(\Lambda)\det(P^T)$. since P is formed by eigenvectors, $\det(P)=1, dep(P^T)=1, so \det(M)=\det(\Lambda)=\lambda_1\lambda_2$. trace(AB)=trace(BA), so $trace(M)=trace(PP^T\Lambda)=trace(\Lambda)=\lambda_1+\lambda_2$ **derivative of sigmoid function** $f'(x)=f(x)(1-f(x))$ **sigmoid problem**: vanishing gradient problem. When f(x) saturates at either 0 or 1, f'(x) becomes nearly 0. **prove heat diffusion equation**: $\frac{\delta G}{\delta\sigma}=\frac{1}{2\pi\sigma^3}\left(\frac{x^2+y^2}{\sigma^2}-2\right)e^{-\frac{x^2+y^2}{2\sigma^2}}, \frac{\delta G}{\delta x}=-\frac{1}{2\pi\sigma^4}xe^{-\frac{x^2+y^2}{2\sigma^2}}, \frac{\delta^2G}{\delta x^2}=\frac{1}{2\pi\sigma^4}\left(\frac{x^2}{\sigma^2}-1\right)e^{-\frac{x^2+y^2}{2\sigma^2}}, so\ \frac{\delta^2G}{\delta x^2}+\frac{\delta^2G}{\delta y^2}=\frac{1}{2\pi\sigma^4}\left(\frac{x^2+y^2}{\sigma^2}-2\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$. it follows that $\frac{\delta G}{\delta\sigma}=\sigma\left(\frac{\delta^2G}{\delta x^2}+\frac{\delta^2G}{\delta y^2}\right)=\sigma\nabla^2$. **prove DoG and LoG$_{norm}$**: $\frac{\delta G}{\delta\sigma}\approx\frac{G(k\sigma)-G(\sigma)}{k\sigma-\sigma}\ and\ \frac{\delta G}{\delta\sigma}=\sigma\nabla^2$, it follows that $\mathrm{DoG(x,y,\sigma)}=G(k\sigma)-G(\sigma)\approx(k-1)\cdot\sigma^2\nabla^2G$ **GELU and ReLU:** GELU resembles the shape pf ReLU. ReLU has a constant zero value for x<0 and its gradient will be 0. ReLU is piecewise linear. **properties of a probability vector**: non-negative and its elements sum to 1. $\frac{\partial p_i}{\partial c_j}$ for softmax: $if\ i=j, \frac{\partial p_i}{\partial c_j}=\frac{e^{c_i}\sum_k e^{c_k}-e^{c_i}e^{c_j}}{(\sum_k e^{c_k})^2}=p_i\cdot(1-p_j). if\ i\neq j, \frac{\partial p_i}{\partial c_j}=\frac{-e^{c_i}e^{c_j}}{(\sum_k e^{c_k})^2}=p_i\cdot(-p_j)$ using Kronecker to donate, we have $\frac{\partial p_i}{\partial c_j}=p_i\cdot(\delta_{ij}-p_j)\ where\ \delta_{ij}=1\ if\ i=j, 0\ if\ i\neq j$ **tables formula** $r_l=r_{l-1}+s_{l-1}(k_l-1), \frac{input-k}{stride}+1(padding=0)$ **aspect ratio** = 宽:高 $precision=\frac{TP}{TP+FP}, recall=\frac{TP}{TP+FN}$ TP, correctly detected positive(左正), TN(右负),FP wrongly detected 左负, FN, missed to be detected 右正. **video denoising algorithm**: 1. estimate the optic flow between adjacent time frames using the Lucas-Kanade method. 2. for each image to be denoised, warp its adjacent frames to this image according to the optic flow field. 3. perform denoising by averaging the current frame with the warped adjacent frames. **a rotation matrix**: $\begin{bmatrix}cos\theta & -sin\theta\\ sin\theta & -cos\theta\end{bmatrix}$ **LoG** 是一种常用的图像边缘检测算法，它通过先对图像进行高斯平滑，然后再计算其二阶导数（拉普拉斯算子），从而实现边缘检测。缺点：计算量大：高斯平滑和拉普拉斯导致边缘检测速度较慢。尺度选择困难: LoG 算法中的高斯滤波器的尺度选择对于边缘检测的性能至关重要。但是，选择适当的尺度需要在多个尺度上进行尝试和验证，这增加了算法的复杂性。边缘定位不准确：由于图像的高斯平滑操作，LoG 算法会使边缘位置产生模糊。此外，对于比较细的边缘或噪声边缘，LoG 算法的定位精度可能会受到影响。**For Siamese network for object tracking, what kind of data can be used for φ:** labeled images, such as images of objects annotated with their locations, as well as video frames annotated with the same object in different time frames. In addition, the network can be trained using synthetic data generated from a simulation environment. **How to robustly track an object in video if its scale changes**: by training the network with different scale of input by using label images or video frames that have been resized to different scales. **proof of separate Gaussian** $f[x,y]*h[x,y]=\sum_i\sum_j f[x-i,y-j]h[i.j]=\sum_i\sum_j f[x-i,y-j]\cdot\frac{1}{2\pi\sigma^2}e^{-\frac{i^2+j^2}{2\sigma^2}}=\sum_i\left(\sum_j f[x-i,y-j]\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{j^2}{2\sigma^2}}\right)\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{i^2}{2\sigma^2}}=\sum_i f*h_y[x-i]\cdot\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{i^2}{2\sigma^2}}=f*h_y*h_x$ **Median filtering**: before corruption, each 3x3 neighbourhood has a uniform intensity c. Less than half pixels are corrupted. Of these, about half will be salt (very bright) and remaining will be pepper (very dark). The majority pixels still have the intensity c. Therefore, the median is x. After 3x3 median filtering, the intensities of a uniform region can be almost perfectly restored. **why a classifier is not working well:** Insufficient or Poor-quality Data. Model Complexity is too simple, it may underfit the data, failing to capture important patterns. Too complex, it may overfit the training data, capturing noise rather than true relationships. Class imbalance occurs when one class has significantly fewer samples than the others. In such cases, the classifier may be biased towards the majority class and perform poorly on the minority class. **How to design the kernel size K of Gaussian?** Usually it is an odd number, like 3, 5, 7 so that it has a clear centre. Larger the kernel, the smoother to filter but the CC increases. **Coordinates:** $X_{Cam}=X_{world}-C\ or\ R(X-C), \begin{pmatrix}2\\2\\4\end{pmatrix}=\begin{pmatrix}0\\0\\0\end{pmatrix}-C, X_{cam}=\begin{pmatrix}I&-C\\0&1\end{pmatrix}X_{world}$. **image matching pipeline**: feature detection (DoG) – feature description (SIFT) – interest point matching.

- Edge detection
  - Canny edge detection
  - Hough Transform (vote)
- Interest Point Detection
  - Harris detector (corner)
  - Scale adapted Harris detector
  - LoG
  - DoG
- Feature Description
  - Simple
  - SIFT algo
  - HOG
- Keypoint matching $Am+b=0.\ m=(A^TA^{-1})A^Tb$
- Image Classification
  - KNN
  - SVM
  - MLP
  - CNN
  - Deep neural network
    - ReLU
  - Vision Transformer
- Image Segmentation
  - unsupervised
    - thresholding
    - K-means clustering
    - GMM
  - supervised
    - CNN
    - FCN
    - DeepLabv3+
    - U-net
- Object Detection
  - 2-stage Faster R-CNN
  - 1-stage SSD
- Motion
  - optic flow
  - Lucas-Kanade method
  - Multi-scale Lucas-Kanade method
- Object Tracking
  - Lucas-Kanade Tracker
  - Correlation filter method: Siamese network
- Camera model
  - pinhole camera
  - camera calibration

1024 KB = 1 MB