ST24 Computer Vision (COMP60006 @ Imperial College London) by ck21

Image Formation, Filtering & Edge Detection

Image Formation. Images are represened as an array of pixels (a set of 3 numbers - RGB). Objects have colour because of reflected light - modelled with Bidirectional Radiance Function BRDF, $f_r(\theta_i, \gamma_i, \theta_r, \gamma_r, \lambda) = \frac{dL_r}{dE_i}$. Cameras sense light via Charged Coupled Device or CMOS. Full RGB Colour Space - gamut of human vision. sRGB is triangle subset of human gamut - cannot produce all human visible colours.
Moving Average Filter. Every value in kernel is the same. $[[\frac{1}{9}, \frac{1}{9}, \frac{1}{9}], [\frac{1}{9}, \frac{1}{9}, \frac{1}{9}], [\frac{1}{9}, \frac{1}{9}, \frac{1}{9}]]$ is 3×3. To perform signal processing using the kernel, set the center pixel to index 0. Scan over each pixel in the image and make the center pixel equal to the $\circ$ of both. Generally work in grayscale as there is only 1 channel. RGB - R, G then B. Complexity with image $N \times N$ and kernel $K \times K$ is $O(N^2, K^2)$. We can separate the filter using **Convolution**. To convolute 2 matrices, flip the kernel along the x and y axis, and scan over each pixel with the kernel. $\circ$ and Σ all values to give new value for centre. After separation, $O(N^2, K)$.
Gaussian Filter. $h(i,j) = \frac{1}{2\pi\sigma^2} e^{\frac{i^2+j^2}{2\sigma^2}}$. Small values outside $[-k\sigma, k\sigma]$ can be ignored (k=3 or 4). Also separable; $h(i,j) = h(i) \star h(j) \cdot h(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$.
High Pass Filter - Identity (just 1 in the center) + High Freq (Identity - Moving Avg).
Edge Detection. Looks at a brightness gradient. For discrete we can use finite difference $f'[x] = f[x+1] - f[x] = f[x] - f[x-1] = \frac{f[x+1]-f[x-1]}{2}$. **Prewitt Filter**: $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}_x, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}_y$
Sobel Filter. $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}_x, \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}_y$. Both are separable - $[1 \quad 1 \quad 1]^T \star [1 \quad 0 \quad -1]$ and $[1 \quad 2 \quad 1]^T \star [1 \quad 0 \quad -1]$. Input $f : g_x = f \star h_x, g_y = f \star h_y, g = \sqrt{g_x^2 + g_y^2}, \theta = arctan(\frac{g_y}{g_x})$. If h is Gaussian, $\frac{dh}{dx} = \frac{-x}{\sqrt{2\pi\sigma^3}} e^{-\frac{x^2}{2\sigma^2}}$. Canny Edge Detection: 1. Perform Gaussian Smoothing, 2. Use Sobel Filter to calculate gradient magnitude and direction. 3. Apply Non-Maximum Suppression to get a single response for each edge. $M(x,y) = M(x,y)$ if local maximum, 0 otherwise. 4. Perform Hysteresis Thresholding to find potential edges. Define t_{low}, t_{high}, if less than low reject. If high, accept as edge. If between weak edge: if connected to existing edge accept, if not reject. Learning based edge detection: ML assumes paired data (images x and manual edge maps y). Find model with parameters θ that maps x to y s.t. $y = f(x|\theta)$. Integrates from multiple scales with coarse-scale edges to form a final output. Canny Edge Detection uses single scale for edge detection, controlled by σ.
Hough Transform. Lines can be parameterised many ways. $y = mx + b$, $\frac{x}{a} + \frac{y}{b} = 1$, $xcos\theta + ysin\theta = p$ where $\theta_{angle}, p_{distance}$. Hough transforms image space (edge map) to parameter space (a line). Each edge votes for possible models in the parameter space. One way is to fit a line model to the edge points $(x1, y1), (x2, y2)...min(m,b)\Sigma_i(y_i - (mx_i + b))^2$. Parameter space is too large for m and b as both are infinite. Thus, we can use normal form (θ and p). Whilst p can still be infinite, $\theta \in [0, \pi)$. The transform will look different (sinusoidal). To perform, we do the following: (1) Initialise bins $H(p, \theta) = 0$. For each edge point (x, y) : for $\theta = [0, \pi)$, calculate $p = xcos\theta + ysin\theta$ and accumulate $H(p, \theta) = H(p, \theta) + 1$. Find (p, θ) where $H(p, \theta)$ is a local maximum and larger than threshold. Similar to Canny. Reduces false positives. The detected lines are given by $p = xcos\theta + ysin\theta$. For circles: (1) Init bins $H(a, b, r) = 0 \forall r \in [r_{min}, r_{max}]$. Let θ be gradient dir, calculate $a = x - ycos\theta, b = y - rsin\theta$.

Interest Point Detection

Interest Points. Useful for processing and analysis - classification, matching, retrieval. Corners, blobs where local structure rich - AKA key points, landmarks, low level feats.
Harris Detector. Corners are intersections of edges. Looking at gradient magnitude of a single pixel is not enough - could be along an edge. Use a small window to tell us if it is a corner. Flat - no intensity change in either direction. Edge - intensity change in 1 direction. Corner both directions. Define window W and window function w - 1 if in window, 0 if not, where we have sum of squared differences; intensity in shifted window and original intensity $E(u,v) = \Sigma_{x,y \in W} w(x,y)[I(x+u, y+v) - I(x,y)]^2$. $w(x,y)$ may use Gaussian to put more focus on the centre of the window. $I(x+u, y+v) = I(x,y) + uI_x(x,y) + vI_y(x,y) + ...I_x$ is image derivative along x, same for y. $E(u,v) = [u,v]\Sigma_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$. If M=Middle Section (everything encompassed by uv either side), we get simple cases $M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ for flat region, $M = \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}$ for edge (large on one of diagonal), $M = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$ for corner. If M is not diagonal matrix, we can use Eigendecomposition, where λ_x is an eigenvalue of M. $M = P \wedge P^T, \wedge = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$. P is a matrix of eigenvectors of M. If R tells us whether a pixel is a corner, then R can be defined in a number of ways. $R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$ - H&S, where k is a small number i.e. 0.05. $R = min(\lambda_1, \lambda_2)$ from Kanade, $R = \frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2} + \epsilon$ from Noble.
Invariance. Harris Detector is rotation invariant. Not invariant to scale. Circles scaled down may be detected as corners. At a given scale, we can see whether the Harris detector gives the highest point. If it looks most like a corner at scale σ, there should be a high response. Scale adapted Harris Detector: $M = \Sigma_{x,y} w(x,y) \sigma^2 \begin{bmatrix} I_x^2(\sigma) & I_x(\sigma)I_y(\sigma) \\ I_x(\sigma)I_y(\sigma) & I_y(\sigma) \end{bmatrix}$. Use this at different scales, determine the scale with largest response. We now have a scale dimension, so the interest point is (x, y, σ). Algo: $\forall\sigma$ perform Gaussian smoothing, calc x y derivatives $I_x(\sigma), I_y(\sigma)$ respectively. Compute M at each pixel and calculate R. Detect interest points which are local maxima across scale and space with $R >$ Threshold.
Laplacian of Gaussian. Performs Gaussian smoothing followed by Laplacian Operator. $\Delta f = \nabla^2 = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}$. Laplacian Filter = $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$. Second Derivative more sensitive to noise: LoG Filter is $f \star (\frac{\delta^2 h}{\delta x^2} + \frac{\delta^2 h}{\delta y^2})$. Since we know 2D gaussian, we have $\frac{\delta^2 h}{\delta x^2} + \frac{\delta^2 h}{\delta y^2} = -\frac{1}{\pi\sigma^4}(1 - \frac{x^2+y^2}{2\sigma^2}) e^{-\frac{x^2+y^2}{2\sigma^2}}$. LoG can be a good interest point detector. Needs scale mulitiplication - 2nd der, so twice. $LoG_{norm}(x,y,\sigma) = \sigma^2(I_{xx}(x,y,\sigma) + I_{yy}(x,y,\sigma))$.
Difference of Gaussian. Defined as difference of gaussians with different scales (suggested $k = \sqrt{2}$). $DoG(x,y,\sigma) = I \star G(k\sigma) - I \star G(\sigma)$. Approximates Normalised LoG: $DoG(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G(x,y,\sigma)$.

Feature Description

Pixel Intensity. Simply intensity of the pixel. Sensitive to lighting conditions. Patch intensity: take some window representing local pattern. Images similar intensity range, roughly aligned, work well. Sensitive to absolute intensity values. Not rotation invariant. Gradient Orientation: not sensitive to intensity but still rotation-variant.
Histogram. Take intensity histogram of patch. Bin intensity values, generate histogram from values. Robust to rotation and scale. Sensitive to intensity.
SIFT 1: Detection of scale-space extrema. Search across scales and pixel locations, looking for interest points. DoG filter, scales continuously multiplied by $\sqrt{2}$. Point of interest if local extremum along both scale and space. Both minima and maxima. If interest point is very bright, DoG $<$ 0. vv for dark blob.
SIFT 2: Keypoint Localisation. Initial SIFT simply locates key points at scale-space extrema. Improved version fits model onto nearby data improving accuracy. Quad func can be fitted onto DoG response of neighbouring pixels, location and scale for extremum of QF can be estimated. Denote DoG response as $D(x,y,\sigma)$ or $D(x)$ where $x = (x,y,\sigma)^T$. From Taylor expansion we get $D(x + \Delta x) = D(x) + \frac{\partial D}{\partial x}^T_{(1)} \Delta x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2}_{(2)} \Delta x$. First derivative estimated with finite difference: $\frac{\partial D}{\partial x} = \frac{D(x+1,y,\sigma) - D(x-1,y,\sigma)}{2}$. Second derivatives / Hessian also estimated with finite difference. Gives QF for Δx, the shift to refined estimate. Therefore, to find a refined extrema for the function: $\frac{\partial D(x+\Delta x)}{\partial \Delta x} = \frac{\partial D}{\partial x} + \frac{\partial^2 D}{\partial x^2}\Delta x = 0 => \Delta x = \frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x}$. Shifting initial by Δx gives us refined estimate. This also gives us refined scale, since it is 3D with a scale-dimension.
SIFT 3: Orientation Assignment. Attempts to assign consistent orientation to each keypoint. Feature descriptors can be represented relative to this orientation for rotation invariance. We need to know orientation of the feature then we can perform sampling in a rotated coordinate system when we calculate features. We can create orientation histogram with 36 bins of 10°: each pixel in neighbourhood votes for orientation bin. We now have (x, y, σ) and orientation Θ. we draw samples using window size prop to σ rotated by θ.
SIFT 4: Keypoint Descriptor. We need to describe the local image content. Histogram of Gradient Orientations. Made from calculated grad mags and orientation for sampling points. Calculated relative to dominant θ. In practice subregions are used, each with 4×4 samples. For each subregion, construct orientation histo with 8 bins 45° each. Also represented with 8 arrows, length of arrow representing grad mag in a given dir. In Lowe's impl, 16 subregions are used. This gives descriptor dim of $128 = 16 \times 8$. Each keypoint described with feature vector containing 128 elems. Robust to rotation due to consideration of dominant orientation. Similarly robust to scale since samples drawn from window prop to scale. Robust to intensity since grad orientations used for feat desc.
Keypoint Matching. Matched by identifying nearest neighbours (Euclidean distance of SIFT descriptors). Each KP in image A identisifies NN in database of KPs for image B. Suppose we find find kp (x,y) in A that corrresponds with (u,v) in B. We assume relation via affine transformation: $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$. With many pairs, we can extend the eqn $\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \dots \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \dots \end{bmatrix}$. This can be written as a linear system, where m is the only unknown $Am = b$. We can obtain the least-sq soln to lin system with Moore Penrose Inverse, minimise $||Am - b||^2; m = (A^T A)^{-1} A^T b$. We have spatial transform between 2 images.
RANSAC. Squared difference sensitive to outliers. Points deemed corresponding that aren't. To ensure robust solution, we can use RANdom SAmple Consensus. (1) Randomly sample some points. (2) Fit line along sampled points. (3) Find number of outliers within threshold of line. (4) Terminate if enough outliers found or we have reached certain iteration no.
SIFT Acceleration. FPGA. Feature detection: separable filtering, downsampling or cropping Gaussian Kernel. Feature description: evaluate grad orientation faster. IP matching: appx NN, use lower dim feature vector.
Speeded-Up Robust Features. Only computes gradients along ver and hor dir using Haar wavelets. SURF still uses the same subregions and sample points but does not calculate gradients for arbitrary orientations. Two simple filters d_y and d_x onto sample points. Summing pixel intensities with a weight of 1 or -1 is very fast, since its either add or sub. For each subregion, sub Haar wavelet responses over sample points, the descriptor for given region defined by 4 elems: $\Sigma d_x, \Sigma d_y, \Sigma|d_x|, \Sigma|d_y|$. If all 4 low, thne homogenous region. If zebra-stripe like, sum 0 but sum $|d|$ large in correct dir. If gradually increasing, then both sums in correct direction will be high. SURF still uses 16 subregions, with 64 dimensionality for each interest point. $\approx 5\times$ faster than SIFT due to simpler Haar wavelets. SURF performs as well as SIFT in matching.
BRIEF. We can further shorten the descriptor by quantisation or binarisation. Haar wavelets compare local region to another, giving float difference. BRIEF will give binary value $(1 if q > p) : \tau(p,q) = 1$ if $I(p) < I(q)$; 0 otherwise. BRIEF samples n_d pairs randomly for bin testes, random pattern determined once, same pattern applied to all interest points. $N_d = 256$ then we get 256 tests, 1 bit output = 32 bytes, SIFT uses 128, SURF 64. Fast due to bitshift. Uses less memory and compares only 2 num. Avoids calc Euclidean dist, use Hamming dist (num bits diff). Efficient calc as XOR descriptors and take bit count. Not ROT or SCALE invariant. Assumes images taken are only translational-movement. Image sim = num matching points.
Histograms of Ordered Gradients HoG. Extend idea of feature descriptors of large region or whole image. HoG divides large region into dense grid of cells, describes each cell and concatenates to form global description. Divide image into equally space cells each 8×8 px. 4 cells form block. Calc grad orientation hist for block (descr of block). Desc vector norm to form locally norm desc. Small err value for stability - $v_{norm} = \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}}$. Block shifted along horizontally by one cell. Some overlaps. HoG desc formed by concatenating all norm desc. HoG used to perform feat extr from imput img to feat repres. Feat Extr tranforms img to low dim vec for easier comparison or matching. Classifier user to give output label for feat desc.

Image Classification

. **KNN**. Dist metric - Euclidean dist (hypotenuse). 784 dim for MNIST. Diff scales, better to normalise feat vec. Norm to Gaussian Dist allowing dims to be treated fairly. Cos dist: $D(x,y) = \frac{x,y}{|x||y|} = \frac{x_1 y_1 + \dots x_n y_n}{\sqrt{\Sigma_{i=1}^n x_i^2}\sqrt{\Sigma_{i=1}^n y_i^2}}$. LP norm:

Manhattan $p=1$ Euclidean $p=2$. $D_p(x,y)=(\Sigma_{i=1}^{n}|x_i-y_i|^p)^{\frac{1}{p}}$. Use $k=1$ works well enough though still inaccurate. $k=32.4\%$ error. KNN no training step simple multiclass capable. Expensive (store training data, high search cost). Comp cost for M test images with N training is $O(MN)$. Want fast test time. Pixel intensities fine for simple preprocessed images but not scale or rot invariant.

SVM. Linear classifier 2D $w_1x_1+w_2x_2+b=0$. Assign class based on $c=1$ if $w_1x_1+w_2x_2+b\geq 0$ else -1. Know w and b we discard training data. Faster in test time. SVM - determine max margin hyperplane. Only need innermost point, SV. SV to fulfil $w\cdot x+b=1$ or -1. Max margin between $wx+b=1$ and $wx+b=-1$. Derive distance metric - $n=\frac{w}{||w||}$. SVM optimises $\min_{w,b}||w||^2+C\Sigma_{i=1}^{N}\xi_i$ s.t. $\forall i:\xi_i\geq 0; y_i(w\cdot x_i+b)\geq 1-\xi_i$. $\xi_i=0$ when correctly classified or between 0 and 1 within margin. Otherwise distance to margin. C is regn term (small C easily fulfilled constraints, large C tigher margin). Slack variable as $\xi_i=max(0,1-y_i(w\cdot x_i+b))_{hingeloss}$. Both convex - know min, local min is global min. Non-neg weighted sum of two convex func also convex. $\forall x_1,x_2\in R,\forall t\in[0,1][f(tx_1+(1-t)x_2)\leq tf(x_1)+(1-t)f(x_2)]$. Line joining $x_1,f(x_1)$ and $x,f(x_2)$ will lie above function curve. 2 optm approaches. GD. Hinge loss not differentiable. Use subgradient - $\nabla_w h=-y_ix_i$ if $y_i(w\cdot x_i+b)<1$ else 0. Similar for $\nabla_b h$ but focus on w for derivation. Each iteration, move along negative of the gradient by step length $\eta:\omega^{(k+1)}=\omega^{(k)}-\eta(2\omega^{(k)}+C\Sigma_{i=1}^{N}\nabla_w h)$. Also Lagrangian Duality. Test time, classify using $c=1$ if $f(x)=w\cdot x+b\geq 0$ else -1. Works well if data linearly separable, if not transform featuers. Transform using kernel. Linear, polynomial, gaussian kernel. Multiclass - 1 v rest. Acc $=\frac{TP+TN}{Total}$, Recall $=\frac{TP}{TP+FN}$, Prec $=\frac{TP}{TP+FP}$

Convolutional Neural Networks

CNNs. Assume input of 2D image and encode properties like local connectivity and weight sharing into the arch, reducing param no and making computation more efficient. When we look at image, we first look at regions, process this info then combine the info from different regions to form global understanding. CNNs - each neuron sees a receptive field in the layer before it. Local connectivity.

Conv Layer. Consider input of $X\times Y\times C$. If neuron required $5\times 5\times C$ cuboid of input, would have $5\times 5\times C\times 1$ (1 bias). IO of neuron: $z=\Sigma_{i,j,k}W_{i,j,k}x_{i,j,k}+b; a=f(z)$. Add more neurons to form $D\times 1\times 1$ output (D=depth) $-a_n=f(\Sigma_{i,j,k}W_{i,j,k}x_{i,j,k}+b_n)$. Window can be moved across input, gives output of $X\times Y\times D$. All windows use same weights. Conv kernel W has 4 dims: output depth, kernel width and height, input depth: $a_d=f(\Sigma_{i,j,k}W_{d,i,j,k}x_{i,j,k}+b_d)$. Output feature map 3 dims: out depth, width and ht. Params for layer: padding (zeroes around edge so output larger). Output shape: $X_{out}=x_{in}+2p-k+1$. Stride (moving more than 1 px at a time) gives stride $s\times s: X_{out}=\left\lfloor\frac{x_{in}+2p-k}{s}\right\rfloor+1$. Dilation (increase region size without downsampling or more params). Use 3×3 kernel look at square of 5×5 and take every other pixel when convoluting. Pooling Layer: take block of pixels from image and takes single value from (max or avg). Can choose pooling window size.

DNNs. Gradient clipping: $g_i=v_{min}$ if $g_i<v_{min}$; v_{max} if $g_i>v_{max}$. g_i otherwise. Or clip by L2 norm: $g=\frac{g}{||g||}v$ if $||g||>v$; g otherwise. Vanishing problem caused on either extreme of sigmoid. Sigmoid Derivative: $f'(z)=\frac{e^{-z}}{(1-e^{-z})^2}=f(z)(1-f(z))$. Propagated Derivative: $\frac{\delta J}{\delta z^{(l)}}=((W^{(l)})^T\frac{\delta J}{\delta z^{(l+1)}})\cdot f'(z^{(l)})$. Can be addressed with different activation functions. ReLU: $f(z)=0$ if $z<0$ else z. Leaky ReLU: $f(z)=0.01z$ if $z<0$ else z. PReLU: $f(z)=az$ if $z<0$ else z (a learnt in training). Exponential ReLU: $f(z)=a(e^z-1)$ if $z<0$ else z.

Recent CNN Architectures. AlexNet similar to LeNet but deeper. Data Aug by cropping random 224×224 regions from OG 256×256 images. Horizontal reflections and pertubing RGB vals. VGG only uses 3×3 conv kernels. Conv of 2 3×3 is 5×5, 3 is 7×7. Deeper and reduces no of param. more non-linearity, AF after every small kernel. Receptive Field - $r_l=r_{l-1}+s_{l-1}(k_l-1)$ k - no filter

Segmentation

Intro. More detailed understanding that BB (Label for each pixel - generates seg mask). Instance seg gives each instance unique label. Thresholding simplest method of segmentation. Converts greyscale image to binary label map. Label defined as $f(x)=1$ if $I(x)\geq$ threshold, 0 otherwise. No training data, only parameter is threshold. Relies on intensity histogram split into 2 classes.

K-means. More than 2 classes. Simple clustering by estimating params. Each cluster represented by centre, each px intensity associated with nearest centre. Optimal centres minimise intra-class variance. C_k data point assoc with $k: min\Sigma_{k=1}^{K}\sigma_{x\in C_k}(x-\mu_k)^2$. Reformulate with delta denoting membership. $min\Sigma_{k=1}^{K}\sigma_{x\in C_k}\delta_{x,k}(x-\mu_k)^2$. If we know δ we can work out μ vv. Can be done iteratively. To determine no of clusters k, calculate class-variance for each k. Can be performed on other features like colour-similarity.

Gaussian Mixture Model GMM. GMMs allow for soft assignment by assuming each cluster is Gaussian distr. Prob that x_j in $C_k=>P(y_j=k|x_i,\pi_k,\mu_k,\sigma_k)=\pi_k\cdot\frac{1}{\sqrt{2\pi\sigma_k}}e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$. Updating similar to K-means. $\pi_k=\frac{\Sigma_j P(y_j=k)}{N}, \mu_k=\frac{\Sigma_j P(y_j=k)\cdot x_j}{\Sigma_j P(y_j=k)}, \sigma_k^2=\frac{\Sigma_j P(y_j=k)\cdot(x_j-\mu_k)^2}{\Sigma_j P(y_j=k)}$. Class assigned by max prob.

CNNs. Seg map manually annotated. Assume we have this data. Each pixel in feat map of AlexNet encodes info about 16×16 region in input image/ Obtain classification results from feat map. Conv replaces FC layers with conv layers enabling network to produce pixel-wise prob map (each class has heat map). Depth of final network represents no. of classes. Upsampling must be performed to obtain pixel-wise prediction since feat map is downsampled. Transposed conv can achieve this. Conv 1D $\begin{bmatrix}w_1&w_2&w_3&0\\0&w_1&w_2&w_3\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=\begin{bmatrix}y_1\\y_2\end{bmatrix}$. Transposed can be $\begin{bmatrix}w_1&0\\w_2&w_1\\w_3&w_2\\0&w_3\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix}$. Classification can be turned into seg by replaceing FC with Conv and Transposed Conv layers. Pixel-wise classification problem. Each pixel define classification loss (Cross-entropy) and seg-loss = average. Same network can segment and perform Obj Detection.

Motion

Optic Flow. Estimates movement of brightness and intensity in videos. Output is a flow field. Same dims as video and describes px discplacement. Video: 2D img seq captured over time. Func of (x,y,t) for each (x,y) at time t we want to estimate $(x+u,y+v)$ at $t+1$. Assumptions: brightness consistency (2 time frames from video, assume two corresponding points will have same greyscale intensities), small motion (objects won't move much per frame), spatial coherence (neighbouring pixels move similarly).

Optic Flow Constraint. $I(x+u,y+v,t+1)=I(x,y,t)$. FO taylor expand on LHS, based on small motion: $I(x+u,y+v,t+1)\approx I(x,y,t)+\frac{\delta I}{\delta x}u+\frac{\delta I}{\delta y}v+\frac{\delta I}{\delta t}=>\frac{\delta I}{\delta x}u+\frac{\delta I}{\delta y}v+\frac{\delta I}{\delta t}=0$. Alternative Notion: $I_xu+I_yv+I_t=0$.

Lucas-Kanade. 2 unknowns - u,v. Spatial Coherence: $\begin{bmatrix}I_x(p)&I_y(p)\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}=-I_t(p)$. Assume u,v same for small neighbourhood, then we have sys of lin eq $A=\begin{bmatrix}I_x(p_1)&I_y(p_1)\\I_x(p_2)&I_y(p_2)\\\dots&\dots\\I_x(p_N)&I_y(p_N)\end{bmatrix}, x=\begin{bmatrix}u\\v\end{bmatrix}, b=\begin{bmatrix}I_t(p_1)\\I_t(p_2)\\\dots\\I_t(p_N)\end{bmatrix}$. Unknowns estimated with Least Sq: $x=\underset{x}{argmin}||Ax-b||^2$. LS uses Moore Penrose method to find solution. $x=(A^TA)^{-1}A^Tb$. $cond(A^TA)=\frac{|\lambda_{max}|}{|\lambda_{min}|}$. Corner motion easier to track than edge.

Aperture Problem. Motion of line ambiguous through aperture. Calc image gradients I_x,I_y either with finite diff or sobel/prewitt. Finite difference between time frames I_t. For each px. Calc matrix for px in neighbourhood: $A^TA=\Sigma_p\begin{bmatrix}I_x^2&I_xI_y\\I_xI_y&I_y^2\end{bmatrix}, A^Tb=-\Sigma_p\begin{bmatrix}I_xI_t\\I_yI_t\end{bmatrix}$ and then optic flow for this pixel $\begin{bmatrix}u\\v\end{bmatrix}=(A^TA)^{-1}A^Tb$. Assumption fails when motion large. Use Multiscale or Multiresolution network. Motion will look smaller in downsampled image. Flow can be estimated using scale pyramids on time-frames. Flow field obtained by summing all incr flows. Multiscale Lucas-Kanade: for all scale l: (1) Upsample flow estimate from previous scale $u^{(l)}=u^{(l+1)}+u^{\delta}$, same for v. (2) Compute warped image $J^l_{warped}=J^l(x+u^{(l+1)},y+v^{(l+1)})$. (3) Compute partial der at new scale: $I_t=J^l_{warped}(x,y)-I^l(x,y)$. (4) Estimate Inc Flow: $\begin{bmatrix}u^{\delta}\\v^{\delta}\end{bmatrix}=(A^TA)^{-1}A^Tb$. (5) Update flow: $u^{(l)}=u^{(l+1)}+u^{\delta}$, same for v.

LK Tracker. Aims to estimate motion from template image I to J (next frame). Assume template moves by u,v and assume will become same as image J. $\underset{u,v}{min}E(u,v)=\Sigma_x\Sigma_u(I(x,y)-J(x+u,y+v))^2_{(1)}$.

Goal is to find how template moves frame to frame. Diff E w.r.t u,v and set to 0. $\frac{\delta E}{\delta u}=-2\times(1)\frac{\delta J}{\delta x}=0$, $\frac{\delta E}{\delta v}=-2\times(1)\frac{\delta J}{\delta y}=0$. Apply chain rule to obtain $\frac{\delta J}{\delta x}$ term (same for 2nd eqn). Use small motion assumption. Approximate J' with I'. $\frac{\delta E}{\delta u}=-2\Sigma_x\Sigma_y[-I_t-I_xy-I_yv]I_x=0$, $\frac{\delta E}{\delta v}=-2\Sigma_x\Sigma_y[-I_t-I_xy-I_yv]I_y=0$. Matrix Form: $\begin{bmatrix}u\\v\end{bmatrix}=-(\Sigma_x\Sigma_y\begin{bmatrix}I_x^2&I_xI_y\\I_xI_y&I_y^2\end{bmatrix})^{-1}\Sigma_x\Sigma_y\begin{bmatrix}I_x&I_t\\I_y&I_t\end{bmatrix}$. Compared to LK OF, we sum over pixels in template image where former sums over small neighbourhood. More complex - use parametric model for motion. $W(x,y;p)=\begin{bmatrix}m_1&m_2\\m_3&m_4\end{bmatrix}\begin{bmatrix}x&y\end{bmatrix}+\begin{bmatrix}t_x\\t_y\end{bmatrix}, \underset{u,v}{min}E(u,v)=$ $\Sigma_x\Sigma_y(I(x,y)-J(W(x,y;p)))^2$. LK classical; assumptions may not hold.

Correlation Filter. Max correlation between template and image feat. $(f\star h)[n]=\Sigma_{m=-\infty}^{\infty}f[m]h[n+m]$. Correlation more robust to illumination changes, compared to SSD. 2D images, search window in next time frame and box with same size as template. Calc correlation for each pos, taking max. Use more discriminative features learnt from CNNs. Siamese networks to compare features. Each pixel in score map relation to pos of search window. Train with Supervised Lng. Paired images can be extracted from vids and annotated. Defines Ground Truth for score map.

Tracking by Detection. Obj Tracking - performing obj detection at each frame. Tracking - typically have initial BB, only care about how single object moves without considering objects in BG. Since we are only interested in initial BB, we can learn specific features. Perform Online Lng (streamed video). Obj tracking - perform Data Aug to obtain more trng data. More samples from more time frames = more to train with. Majority of network can be pretrained with large datasets, only last few layers need online trng. Fewer parameters in cls and loc branches, trained with fewer samples.

Camera Models

Intro. Closely related to motion, mapping 3D world to 2D image. Denote 3D coords as X and 2D coords as x. Mapping described with camera matrix $P: x=PX$.

Pinhole Camera. Model how X is mapped to x on image plane. Image plane is rotated by 180° and put as virtual image plane in front of pinhole. Pinhole is camera origin. Distance from camera origin to image plane is focal length f. One dim, use similar triangles $(X,Y,Z)=>(f\frac{X}{Z},f\frac{Y}{Z})$. Homogenous coords mean $(x,y,1)$ is the same as (kx,ky,k) for any k. Homogenous coords - $(X,Y,Z,1)=>(fX,fY,Z)$.

Principle Point Offset. CCD array, image origin defined as bottom/top left, hence image coordiante system origin may differ from Principle point p - centre of image plane. To account - $(X,Y,Z,1)=>(f\frac{X}{Z}+p_x,f\frac{Y}{Z}+p_y,1)$ can be matrified.

World Coordinate System. If rotation, need to factor in 3D Rot matrix $R: X_{cam}=R(X-C)$ R Rot C Transl. Inhomogenous coords. Homogenous - $\begin{bmatrix}X_{cam}\\Y_{cam}\\Z_{cam}\\1\end{bmatrix}_{(x_{cam})}=\begin{bmatrix}R&-RC\\0&1\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}_{(X)}$

Camera Matrix. Map world coord X to image coord x. $x=\begin{bmatrix}f&0&p_x&0\\0&f&p_y&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}R&-RC\\0&1\end{bmatrix}X$. Camera matrix can be written more concisely as $P=\begin{bmatrix}f&0&p_x\\0&f&p_y\\0&0&1\end{bmatrix}\begin{bmatrix}R&-RC\end{bmatrix}=K\begin{bmatrix}R&-RC\end{bmatrix}=KR\begin{bmatrix}I&-C\end{bmatrix}$.

Total 9 DoF (3 intrinsic, 3 3D Rot, 3 3D Trans). Still representing image coordinates in same units as camera coordinates. Conv ratio k_x,k_y like pixels per mm - $(f\frac{X}{Z}+p_x,f\frac{Y}{Z}+p_y,1)=>(k_x(f\frac{X}{Z}+p_x),k_y(f\frac{Y}{Z}),1)$. Converts camera to a_x,a_y instead of f and x_0,y_0 in place of p_x,p_y. Can be increased to 11 DoF when considering skew.

Calibration. Assume we know 3D structure, estimate camera matrix. Given a set of points $\{X_i,x_i\}$ we want to estimate P. Mapping: $\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}p_1&p_2&p_3&p_4\\p_5&p_6&p_7&p_8\\p_9&p_{10}&p_{11}&p_{12}\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}=\begin{bmatrix}p_1^T\\p_2^T\\p_3^T\end{bmatrix}X$. 2nd term allows us to have $x=\frac{p_1^TX}{p_3^TX}, y=\frac{p_2^TX}{p_3^TX}$. Overdetermined matrix: $\begin{bmatrix}X_1^T&0&-X_1^Tx_1\\0&X_1^T&-X_1^Ty_1\\\dots&\dots&\dots\\X_n^T&0&-X_n^Tx_n\\0&X_n^T&-X_n^Ty_n\end{bmatrix}$. p is the nullspace of A. Add constraint $P=argmin_p||Ap||^2$. Solve with SVD for A $(U\Sigma V^T)$ - col V smallest sing val.