1B/4

**Engineering Tripos Part IIB** FOURTH YEAR

Module 4F12: Computer Vision

# Image Structure 2022 Solutions

1. (a) i. **Solution:**
The central motivation is *data reduction/compression*: raw pixel data is impractical to process directly for current hardware, particularly for real-time applications.
*Desirable properties* for generic features include:
(1) They should preserve the useful information in an image (e.g. 2D shape of objects);
(ii) They should discard redundant/nuisance information (e.g. lighting conditions);
(iii) They should be as generic as possible (so the same features are useful across a wide range of applications). [10%]

(b) i. **Solution:**
To reduce the effects of high-frequency noise, we can convolve the image with a 2D Gaussian (i.e. low-pass) filter.
Mathematically, this corresponds to producing a smoothed image:

$$S(x,y) = \sum_u \sum_v G_\sigma(u,v) I(x-u, y-v)$$

where

$$G_\sigma(u,v) = \frac{1}{2\pi\sigma^2} \exp\Big(-\frac{u^2+v^2}{2\sigma^2}\Big)$$

This can be implemented efficiently by using the *separable* property of the Gaussian:
Using $G_\sigma(x,y)$ as above to denote a 2D Gaussian with standard deviation $\sigma$ along both axes and let $g_\sigma(x)$ denote a 1D Gaussian. The separable property of the Gaussian ensures that

$$G_\sigma(x,y) * I(x,y) = g_\sigma(x) * [g_\sigma(y) * I(x,y)]$$

Consequently, convolution with the discrete 2D kernel of 25 elements can be replaced by two convolutions with 1D kernels, each of size 5 elements.
*Savings*:
1. The output image size after filtering via "valid" convolution is $316 \times 236$. The total cost (in multiplications) for using 2D Gaussians is $316 \times 236 \times 5 \times 5 = 1,864,400$
2. To perform the convolution with 1D Gaussians requires two steps. It is slightly cheaper to perform the vertical 1D Gaussian first (this results from the fact that the image is not square). The output size from the vertical 1D Gaussian convolution is $320 \times 236$, and requires $5 \times 1 \times 320 \times 236 = 377,600$ multiplications to produce. The output from the second (horizontal) 1D Gaussian will have size $316 \times 236$ and will required $1 \times 5 \times 316 \times 236 = 372,880$ multiplications to produce. In total, the cost of both stages of the 1D approach is $750,480$. [15%]
3. The overall count of multiplications saved is $1,864,400 - 750,480 = 1,113,920$.

ii. **Solution:**
We can derive a discrete $3 \times 3$ filter which approximates the 2-D Laplcian by considering Taylor expansion at a given location $(x,y)$ the image, first in one dimension:

$$I(x+\delta, y) = I(x) + \delta \frac{\partial}{\partial x} I(x,y) + \frac{\delta^2}{2}\frac{\partial^2}{\partial x^2} I(x,y) + \mathcal{O}(\delta^3)$$

We then consider the cases when $\delta = 1$ and $\delta = -1$:

$$I(x+1,y) = I(x,y) + \frac{\partial}{\partial x} I(x,y) + \frac{1}{2}\frac{\partial^2}{\partial x^2} I(x,y) + \mathcal{O}(\delta^3)$$
$$I(x-1,y) = I(x,y) - \frac{\partial}{\partial x} I(x,y) + \frac{1}{2}\frac{\partial^2}{\partial x^2} I(x,y) + \mathcal{O}(\delta^3)$$

Adding one to the other and neglecting higher order terms yields:

$$I(x+1,y)+I(x-1,y)\approx 2I(x,y)+\frac{\partial^2}{\partial x^2}I(x,y)$$
$$\implies \frac{\partial^2}{\partial x^2}I(x,y)\approx I(x+1,y)-2I(x,y)+I(x-1,y)$$

A similar relation holds for the $y$ direction. Thus we have that:

$$\nabla^2 I(x,y)=\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial x^2}\right)I(x,y)\approx$$
$$I(x+1,y)-2I(x,y)+I(x-1,y)+I(x,y+1)-2I(x,y)+I(x,y-1)$$

This can be implemented with a $3\times 3$ discrete filter of the form:

$$\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

*Implementation with 1D filters.* Since the Laplacian represents the sum of two 1D kernels computing second derivatives, we can instead convolve $I(x,y)$ with the following two filters

$$\partial_{x^2}=\begin{bmatrix}1 & -2 & 1\end{bmatrix}\quad \partial_{y^2}=\begin{bmatrix}1\\ -2\\ 1\end{bmatrix}$$

and sum their outputs:

$$\nabla^2 I(x,y)=\frac{\partial^2}{\partial x^2}I(x,y)+\frac{\partial^2}{\partial x^2}I(x,y)$$

The 1D filter approach requires fewer multiplications: each Laplacian response requires $3+3=6$ multiplications (vs $3\cdot 3=9$ multiplications for the $3\times 3$ kernel). However, under the assumptions of the question (the filters are applied sequentially), it requires more memory: After applying the first 1D kernel (e.g. $\frac{\partial^2}{\partial x^2}$) we must keep hold $I(x,y)$ and $\frac{\partial^2}{\partial x^2}I(x,y)$ in memory while beginning to compute $\frac{\partial^2}{\partial x^2}I(x,y)$ in order to generate the final result that sums the 1D filter responses in each position.
*Note*: given the size of the image ($320\times 240$), we neglect (as insignificant) the memory required to store the additional 3 filter values required by the $9=3\cdot 3$ 2D kernel over the $6=3+3$ filter values associated with the two 1D kernels. [15%]

iii. **Solution:**
While there are multiple viable algorithms, the course focused on a simple implementation of the Canny Edge detector, so this is the solution described below.
1. The image $I(x,y)$ is first filtered with a 2D Gaussian kernel, $G_\sigma$, to produce a smoothed image $S(x,y)$.
2. The gradient of $S(x,y)$ is then computed at every pixel (i.e. $\nabla S=\nabla(G_\sigma * I)$).
3. Non-maxima suppression is applied over the magnitude of the gradients (placing "edgels" at locations where $|\nabla S|$ is greater than nearby values of $|\nabla S|$ in directions $\pm\nabla S$.
4. Edgels are thresholded, so that only those with a $|\nabla S|$ above some threshold are retained.
5. Double-thresholding is applied to determine "strong" and "weak" edges, before hysteresis is employed to suppress weak edges (Note: this stage was not discussed explicitly in lectures).
The operations of computing gradients and removing high-frequency noise (i.e. steps 1 and 2)

are *computed efficiently* by combining them into a single operation via the *derivative theorem of convolution* that ensures the *associativity* of differentiation and convolution:

$$\nabla S = \nabla(G_\sigma * I) = \begin{bmatrix} \frac{\partial(G_\sigma * I)}{\partial x} \\ \frac{\partial(G_\sigma * I)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial G_\sigma}{\partial x} * I \\ \frac{\partial G_\sigma}{\partial y} * I \end{bmatrix}$$

[15%]

(c) i. **Solution:**
Edges have two primary limitations that renders them ineffective as keypoints for correspondences:
(1) They only allow localistaion in one dimension - the direction normal to the edge. They do not allow localisation parallel to the edge (the "aperture problem");
(2) Edges cannot be used to determine the scale of a feature. [10%]

ii. **Solution:**
(1) The descriptor is computed at the location of a keypoint - this provides *translation invariance.*
(2) The descriptor is computed from $16 \times 16$ pixels at the characteristic scale determined by the corresponding keypoint - this provides *scale invariance.*
(3) The gradient orientations of the descriptor are stored relative to the dominant orientation of the keypoint - this provides *rotation invariance.*
(4) The gradients computed at each pixel in the patch are weighted a Gaussian (with $\sigma$ equal to half the width of the descriptor window i.e. $\sigma = 8$ pixels for $16 \times 16$ windows) - this provides a degree of *partial occlusion invariance* by reducing the influence of image gradients far from the centre that are most susceptible to misregistration.
(5) Histograms are computed over *cells* of $4 \times 4$ pixels. As a result, an image gradient can shift by up to 4 pixels while still contributing to the same histogram - this provides a small degree of *local positional invariance.*
(6) Gradient orientations are aggregated in histograms with relatively coarse bins (8 orientation bins per histogram) - this provides a small degree of further *local rotational invariance.*
(7) The SIFT descriptor is normalised to unit length - this provides invariance to *affine changes in illumination.*
(8) Each value in the descriptor vector is then further truncated to a maximum value of 0.2 and then renormalised. This provides a small degree of invariance to non-linear illumination changes (e.g. due to camera saturation).
Typical failures scenarios include: (i) large changes in viewpoint, (ii) background clutter at object boundaries. [20%]

iii. **Solution:**
There are many possible nuisance factors. Reasonable answers could include:
1. Partial occlusion (caused by any of: (i) snow obscuring features of the building, such as roof markings or windows, (ii) greater/fewer numbers of tourists at different times of year could add further occlusion).
2. Differences in light source position (for images taken at similar times on the same day, the sun will not have moved much, but photos across seasons could have significantly greater variation in sun position).
3. Significant differences in illumination and contrast (e.g. from winter snow reflection and/or shadows thrown by the summer sun).
4. Differences in reflectance properties of the surfaces (dry building walls vs moist walls from snow). [15%]

2 (a) (i) Assumptions: pin-hole camera, central planar projection with no non-linear distortion.

Coordinates: $[X, Y, Z]^\top = \left[\frac{X_1}{X_4}, \frac{X_2}{X_4}, \frac{X_3}{X_4}\right]^\top$, $[u, v]^\top = \left[\frac{x_1}{x_3}, \frac{x_2}{x_3}\right]^\top$. [10%]

(ii) $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{T}]$, where $\mathbf{K} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ - $3 \times 3$ upper triangular (4 DoF, 5 parameters), $\mathbf{R}$ - $3\times3$ orthogonal (3 DoF, 3 parameters if parametrised using Euler angles), $\mathbf{T}$ - $3 \times 1$ vector encoding camera position (3 DoF, 3 parameters).

Here $(u_0, v_0)$ - principal point, $f$ - focal length, $(k_u, k_v)$ - pixel sizes, $(\alpha_u = k_u f, \alpha_v = k_v f)$ - image scaling factors, $\frac{\alpha_v}{\alpha_u}$ - aspect ratio.

Total: 11 parameters, 10 DoF. [15%]

(iii) Setting up homogeneuous system of equations. One 2D - $(u_i, v_i)$ to 3D - $(X_i, Y_i, Z_i)$ correspondence gives two equations: $u_i = \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p14}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p34}$ and $v_i = \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p24}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p34}$. Rearranged as:

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \mathbf{p}^\top = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

where $\mathbf{p} = [p_{11}, p_{12}, \cdots, p_{34}]^\top$ is a $12 \times 1$ column vector. For N pairs we have: $\mathbf{Ap} = \mathbf{0}$, where $\mathbf{A}$ is a $2n \times 12$ matrix.

Solving system of equations. Since scale of $\mathbf{p}$ does not matter (11 DoF) we can solve this by orthogonal least squares. The solution is the eigenvector of $A^\top A$ corresponding to the smallest eigen value. This solution minimizes $|\mathbf{Ap}|$ such that $|\mathbf{p}| = 1$ and can be derived by considering Rayleigh's Quotient: $\lambda_1 \leq \frac{\mathbf{p}^\top \mathbf{A}^\top \mathbf{Ap}}{\mathbf{p}^\top \mathbf{p}} \leq \lambda_n$. Need $n \geq 6$ correspondences to obtain a solution.

Robust estimation. **Step 1** - use RANSAC: (1) - randomly sample 6 pairs of correspondences, (2) - compute $\mathbf{p}$, (3) - compute inliers, (4) - repeat to maximize inliers.

**Step 2** - use the result of previous step as initialisation to a non-linear optimization step which minimizes re-projection error: $\text{argmin}_{\mathbf{p}} \sum_{i=1} (u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2$, where $(\hat{u}_i, \hat{v}_i)$ are estimated from the model.

Recovering $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{K}$. Since $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{T}] = [\mathbf{KR}|\mathbf{KT}]$, we can obtain $\mathbf{K}$ and $\mathbf{R}$ by RQ decomposition of $\mathbf{P}[:3, :3]$. Then $\mathbf{T} = \mathbf{K}^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$. [20%]

(iv) Projection matrix. $\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \mathbf{K}\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix}\left[\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{T} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$

Hence,

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} fk_ur_{11} & fk_ur_{12} & fk_ur_{13} & fk_uT_x + u_0Z_c^{av} \\ fk_vr_{21} & fk_vr_{22} & fk_vr_{23} & fk_vT_y + v_0Z_c^{av} \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

After scale normalisation - linear projection equations: $\begin{bmatrix} u \\ v \end{bmatrix} = [2\times 4]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$

Advantages. (1) - can be calibrated from 4 points instead of 6 points, (2) - calibration process is better conditioned (less sensitive to noise than non-linear full perspective calibration.

Viewing conditions. Assumes that the variation in depth of the objects is small compared to the distance of the camera to the scene. [15%]

(b) (i) $\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \mathbf{K}\,[R|T]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$, where $\mathbf{H}$ is a $3\times 3$ non-singular (full rank) matrix in the general case (9 parameters, 8 DoF). Rewriting in homogeneous coordinates: $\tilde{\mathbf{w}}' = \mathbf{H}\tilde{\mathbf{w}}$, where $\tilde{\mathbf{w}}$ is a homogeneous 2-D coordinate of a point on the floor plane and $\tilde{\mathbf{w}}'$ - corresponding pixel coordinate. [10%]

(ii) Homogeneous 2-D line equation: $\tilde{\mathbf{l}}^\top\tilde{\mathbf{w}} = 0$. From $\tilde{\mathbf{w}}' = \mathbf{H}\tilde{\mathbf{w}}$ we have $\tilde{\mathbf{w}} = \mathbf{H}^{-1}\tilde{\mathbf{w}}'$ (since $\mathbf{H}$ is full rank). The line equation can be rewritten as $\tilde{\mathbf{l}}^\top(\mathbf{H}^{-1}\tilde{\mathbf{w}}') = 0$. Hence the homogeneous line, $\tilde{\mathbf{l}}'$, in the image is $\tilde{\mathbf{l}}' = \mathbf{H}^{-\top}\tilde{\mathbf{l}}$. [10%]

(iii) Estimating **H**. Let $\tilde{\mathbf{l}} = [a,b,c]^\top$ and $\tilde{\mathbf{l}}' = [a',b',c']^\top$. Since $\tilde{\mathbf{l}} = \mathbf{H}^\top\tilde{\mathbf{l}}'$, we have $\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{bmatrix}\begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$. This gives two equations for a single line correspondence and can be rearranged as follows: $\begin{bmatrix} a'c & 0 & -a'a & b'c & 0 & -b'a & c'c & 0 & -c'a \\ 0 & a'c & -a'b & 0 & b'c & -b'b & 0 & c'c & -c'b \end{bmatrix}\mathbf{h}^\top = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, where $\mathbf{h} = [h_{11}, h_{12}, \cdots, h_{33}]^\top$ is a $9\times 1$ column vector. For N pairs we have: $\mathbf{Ah} = \mathbf{0}$,

where $\mathbf{A}$ is a $2n \times 9$ matrix (8DoF, need 4 line correspondences, no three lines should be parallel). Homogeneous system of equations is solved and matrix $\mathbf{H}$ is robustly estimated in the similar way as in Part(a)(iii).
Advantages of using lines. (1) - lines may be estimated more accurately at the sub-pixel level. (2) - it is hard to localise points at sub-pixel level and some type of tyles may not have clear sharp (as opposed to round) corners.
Image rectification. Rectified image is obtained by retrieving pixel intensities for a point $\tilde{\mathbf{v}}$ in the new image from $\mathbf{H}^{-1}\tilde{\mathbf{v}}'$ in the source image. Bilinear interpolation can be used to reduce discretisation effects. [20%]

3 (a) Convolutional stage.

$$a_{i,j,c_{out}} = \left(\sum_{k,l,c_{in}} w^{c_{out}}_{k,l,c_{in}} x_{i-k,j-l,c_{in}}\right) + b^{c_{out}}$$

Here, $c_{in}$ - number of input channels, $c_{out}$ - number of output channels, $w^{c_{out}}_{k,l,c_{in}}$ - convolution kernel weights for output channel $c_{out}$.

(1) - Many image properties are translation invariant. (2) - reduces number of parameters required compared to fully connected layers. (3) - low-level features are likely to be local.

Non-linear stage.

$$y_{i,j,c} = f(x_{i,j,c})$$

$g$ - (point-wise) non-linear function. E.g. ReLU - $f(x) = max(0,x)$ or sigmoid - $f(x) = \frac{1}{1+e^{-x}}$.

(1) - allows a (convolutional) neural network to learn complex (non-linear) decision boundaries. (2) - a single hidden layer neural network is a universal approximator.

Pooling/subsampling stage.

$$z_{i,j,c} = \max_{k,l}(x_{i-k,j-l}) \text{ (e.g. max-pooling)}$$

Note that a stride may be applied to pooling operations and other pooling operations like average pooling could be used.

Pooling stage is useful to perform image subsampling in order to (1) encourage learning of feature hierarchies and to (2) reduce the number of paramters required. Also it introduces (3) a slight translation invariance. [20%]

(b) Let

$$a^{1,(n)}_i = \sum_k w_k x^{(n)}_{i-k}, \qquad y^{1,(n)}_i = f(a^{1,(n)}_i)$$

$$a^{2,(n)}_i = \sum_l v_l y^{1,(n)}_{i-l}, \qquad y^{2,(n)}_i = f(a^{2,(n)}_i)$$

$$a^{(n)}_i = \sum_d W_{i,d} y^{2,(n)}_d, \qquad y^{(n)}_i = \text{softmax}(a^{(n)}_1, a^{(n)}_2, \cdots)$$

Objective function - $G(\{\mathbf{y}^{(n)}\}, \{w, v, W\}$.

$\frac{\partial G}{\partial y_i^{(n)}}$ and $\frac{\partial y_i^{(n)}}{\partial a_t^{(n)}}$ are known.

$$\frac{\partial G}{\partial w_k} = \sum_n \sum_i \frac{\partial G}{\partial y_i^{(n)}} \sum_t \frac{\partial y_i^{(n)}}{\partial a_t^{(n)}} \frac{\partial a_t^{(n)}}{\partial w_k} = \sum_n \sum_i \frac{\partial G}{\partial y_i^{(n)}} \sum_t \frac{\partial y_i^{(n)}}{\partial a_t^{(n)}} \sum_d W_{t,d} f'(y_d^{2,(n)}) \frac{\partial a_d^{2,(n)}}{\partial w_k} =$$

$$= \sum_n \sum_i \frac{\partial G}{\partial y_i^{(n)}} \sum_t \frac{\partial y_i^{(n)}}{\partial a_t^{(n)}} \sum_d W_{t,d} f'(y_d^{2,(n)}) \sum_l v_l \frac{\partial y_{d-l}^{1,(n)}}{\partial w_k} =$$

$$= \sum_n \sum_i \frac{\partial G}{\partial y_i^{(n)}} \sum_t \frac{\partial y_i^{(n)}}{\partial a_t^{(n)}} \sum_d W_{t,d} f'(y_d^{2,(n)}) \sum_l v_l f'(a_{d-l}^{1,(n)}) x_{d-l-k}.$$

[20%]

(c) (i) It is reasonable to assume that the number of output channels increases further away from the input to CNN. Hence $C_1 = 2^n$, $C_2 = 2^{n+1}$, $C_3 = 2^{n+2}$, where $C_i$ - the number of output channels in the $i$th convolutional layer. Position of pooling layer does not have an effect on the number of parameters in the network. Hence,

| Layer | Output shape | No. of params. |
|---|---|---|
| INPUT | $H \times H \times 3$ | 0 |
| CONV1 | $H \times H \times C_1$ | $K \cdot K \cdot 3 \cdot C_1 + C_1$ |
| CONV2 | $H \times H \times C_2$ | $K \cdot K \cdot C_1 \cdot C_2 + C_2$ |
| POOL | $\frac{H}{2} \times \frac{H}{2} \times C_2$ | 0 |
| CONV3 | $\frac{H}{2} \times \frac{H}{2} \times C_3$ | $K \cdot K \cdot C_2 \cdot C_3 + C_3$ |
| FC | $L$ | $\left(\frac{H^2}{4} \cdot C_3 + 1\right) \cdot L$ |

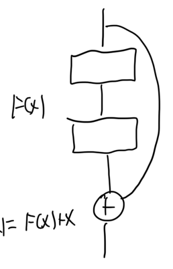
Number of parameters, $P = K^2 3 C_1 + C_1 + K^2 C_1 C_2 + C_2 + K^2 C_2 C_3 + C_3 + \left(\frac{H^2}{4} C_3 + 1\right) L$. Since $K = 5$, $H = 32$, $L = 100$, we have $1703812 = 76C_1 + C_1(50C_1 + 2) + C_1(200C_1 + 4) + 1024000C_1 + 100$. Hence, $C_1 = 16 = 2^4$ and $n = 4$. [20%]

(ii) Number of parameters would be large.

(1) - $3 \times 3$ convolutions could be used instead of $7 \times 7$ convolutions as in VGG-16.

(2) - A single $7 \times 7$ convolution can be approximated as three $3 \times 3$ convolutions applied sequentially.

Slow/inefective learning due to vanishing gradients of inherently difficult learning task.



(1) - simply stacking convolutional, non-linear and pooling layers performs worse both on train and test data after certain depth. (2) - ResNet-34 uses residual connections which enable the network to fit a potentially simpler residual value $F(x)$ instead of directly fitting the desired value $H(x) = F(x) + x$.
Changing weight distribution. During training, updates of the weights of the preceding layer may significantly change the distribution of the input values to the succeeding layers, especially for very deep networks.
(1) - ResNet-34 uses batch normalisation to normalise outputs of intermediate layers as follows.
$\mu_k = \frac{1}{M}\sum_m y_k^{(m)}$ - mean of mini-batch of $M$ images. $\sigma_k^2 = \frac{1}{M}\sum_m (y_k^{(m)} - \mu_k)^2$ - mini-batch variance. $\hat{y}^{(m)} = \frac{y_k^{(m)} - \mu_k}{\sqrt{\sigma_k^2 + \epsilon}}$ - normalized output. Finally, scaled and shifted output: $y_k'^{(m)} = \gamma \hat{y}_k^{(m)} + \beta$, where $\gamma$ and $\beta$ are learnable parameters. [25%]

(iii) Objective function. Tripplet loss:

$$L = \sum_{m=1}^{M} \max\left(0, ||\mathbf{e}_A^{(m)} - \mathbf{e}_+^{(m)}||_2^2 - ||\mathbf{e}_A^{(m)} - \mathbf{e}_-^{(m)}||_2^2 + \Delta\right).$$

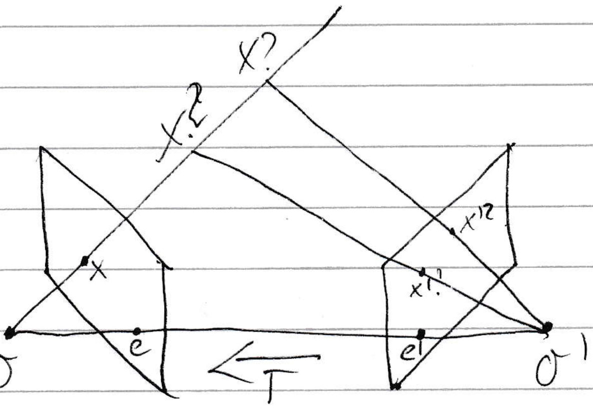
Here $\Delta$ - margin parameter, $M$ - number of triplets in a batch, $\mathbf{e}_A$ - CNN features for the anchor sample, $\mathbf{e}_+$ - CNN features for the positive sample, $\mathbf{e}_-$ - CNN features for the negative sample.

Changes in the training procedure. (1) Replacing the categorical cross entropy objective function with the tripplet loss. (2) Sampling tripplets of images (anchor, positive, negative) instead of a single image when training. An image is considered to be positive if it contains a road sign of the same class and negative if it contains a road sign of a different class than the anchor image. (3) Note, augmented versions of anchor image could be used as examples of positive samples.
Changes in the testing procedure. (1) Instead of directly predicting class labels, one would first need to compute CNN features of the test image and compare them with the features of images of road signs in the database. The class of the most similar image (or the majority class of k closest images) can be used as the output class label of the test image. (2) Note in this case if one wants to apply the road sign recognition system in another country, one can simply extend the database with CNN features of newly collected images of previously unseen classes of road signs. No (re-)training is required. [15%]

Q~~4~~9(a)(i)

IF WE OBSERVE POINT p IN ONE IMAGE THEN ITS POSITION p' IN OTHER IMAGE MUST LIE ON A LINE.

$p'^T E p = 0$



DERIVATION:

(1) $X_c' = RX_c + T$ [CAMERA CENTERED POSITIONS OF X]

(2) $T \times X_c' = T \times (RX_c + T)$
$T \times X_c' = T \times RX_c$

(3) $X_c' \cdot (T \times X_c') = X_c' \cdot (T \times RX_c)$
$0 = X_c' (T \times R\cancel{X_c})$

(4) $0 = X_c'^T [T]_x R X_c$, HERE $[T]_x = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$

FOR RAY p WE HAVE $\lambda p = X_c$

HENCE $(\lambda' p'^T) E (\lambda^* p^*) = 0$
$p'^T E p^* = 0$

[20%]

Q4 (a) (ii)

$$\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0/f \\ 0 & k_v & v_0/f \\ 0 & 0 & \frac{1}{f} \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix}$$

$$\begin{bmatrix} fu \\ fv \\ f \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix}$$

HENCE HOMOGENEOUS COORDINATES $\tilde{w}$ OF RAY P CAN BE EXPRESSED AS $\tilde{w} = Kp$

HENCE $p = K^{-1}\tilde{w}$ $\quad p' = K'^{-1}\tilde{w}'$

$p'^T E p = 0 \Rightarrow (K'^{-1}\tilde{w}')^T E K^{-1}\tilde{w} = 0 \Rightarrow$ [15%]

$\Rightarrow \tilde{w}'^T K'^{-T} [T]_\times R K^{-1}\tilde{w} = 0$ SINCE $K' = K \Rightarrow \tilde{w}'^T K^{-T} [T]_\times R K^{-1}\tilde{w} = 0$

(a)(iii) $$[x' \; y' \; 1] \begin{bmatrix} f^{-1} & 0 & 0 \\ 0 & f^{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 2d \\ 0 & 0 & +d \\ -2d & -d & 0 \end{bmatrix} \begin{bmatrix} f^{-1} & 0 & 0 \\ 0 & f^{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

$$[x' \; y' \; 1] \begin{bmatrix} 0 & 0 & 2d \\ 0 & 0 & d \\ -2d & -d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

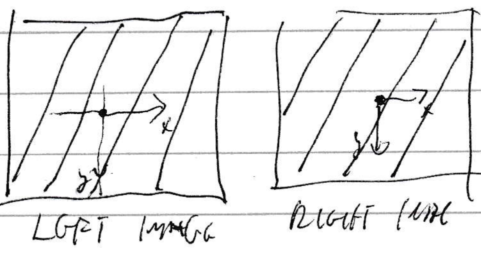
$$[-2d \quad d \quad x'2d + y'd] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

[15%]

$$[-2d \quad -d \quad + x'2d + y'd] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

$$-2dx - dy + x'2d + y'd = 0$$

$$+2x + y = 2x' + y'$$



Q4(b)(i)

STEP 1: FUNDAMENTAL MATRIX ESTIMATION FROM CORRESPONDENCES

NEED 8 CORRESPONDENCES FOR A LINEAR METHOD OR 7 CORRESPONDENCES FOR A NON-LINEAR METHOD

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

1 POINT 1 CONSTRAINT

$$\begin{bmatrix} u_i' u_i & u_i' v_i & u_i' & v_i' u_i & v_i' v_i & v_i' & u_i & v_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ \vdots \\ f_{33} \end{bmatrix} = 0$$

SOLVE $Af = 0$ BY MINIMISING $|Af|$ SUBJECT TO $|f| = 1$

$$\lambda_1 \le \frac{f^T A^T A f}{f^T f} \le \lambda_2$$

SOLUTION - ~~SMALLEST~~ EIGEN VECTOR CORRESPONDING TO SMALLEST EIGEN VALUE OF $A^T A$

~~MAKE~~ PROJECT TO F RANK 2 BY SVD $(\lambda_3 = 0)$

RUN RANSAC FOR ROBUST ESTIMATION

~~FOLLOW BY NON-LINEAR OPTIMISATION OF 7 PARAMETERS~~

FOLLOW BY NON-LINEAR OPTIMISATION TO MINIMIZE GEOMETRIC ERROR.

Q~~3~~4(e)(i) CONTINUED

STEP 2. RECOVER R,T FROM F

SINCE K IS KNOWN

$E = K^T F K = T_x R = U \Lambda V^T$

$\hat{T}_x = U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T \qquad R = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T$

$\hat{T}_x$ - UNKNOWN SCALE    R - ORTHONORMAL MATRIX

4 SOLUTIONS ~~FOR~~ FOR $\pm T$ AND $R^T, R$    $P = K[I|0]$ $P' = K[R|T]$

AMBIGUITIES RESOLVED BY ENSURING THAT VISIBLE POINTS LIE IN FRONT OF THE TWO CAMERAS.

Q~~3~~ (e)(ii) IF T=0 — NO TRANSLATION, PURE ROTATION THEN CAN NOT USE EPIPOLAR CONSTRAINTS

UNDER PURE ROTATION

$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad H = K R K^{-1}$

* - 3D COORDINATES CAN NOT BE RECOVERED AS PIXEL DEPTH COULD BE ARBITRARY UNDER PURE ROTATION

1 POINT - 2 CONSTRAINTS. ~~TOTAL~~
MINIMUM 4 CORRESPONDENCES NEEDED

ESTIMATION → SIMILAR TO (e)(i)    [15%]

RECOVER $R = K^{-1} H K$. WE KNOW THAT T=0
SINCE R MAY NOT BE A PROPER ROTATION MATRIX, SVD CAN BE USED TO OBTAIN THE CLOSEST ROTATION MATRIX.

Q~~4~~5 (c)

STEP 1.

(1) TAKE 2 IMAGES AND OBTAIN F
(2) DECOMPOSE F INTO R,T BY USING KNOWLEDGE OF K
(3) USE THIS TO PROVIDE INITIAL 3D COORDINATE ESTIMATES OF SPARSE IMAGE FEATURES
(4) PROJECT THESE COORDINATES INTO OTHER IMAGES AND ~~USE~~ MINIMISE THE REPROJECTION ERROR ITERATIVELY BY UPDATING 3D STRUCTURE AND CAMERA POSES. THIS IS CALLED BUNDLE ADJUSTMENT.

(5) DENSE 3D STRUCTURE CAN BE OBTAINED BY ESTABLISHING DENSE MATCHES ON PAIRS OF IMAGES WITH ESTIMATED CAMERA POSES AND TRIANGULATING.

TRIANGULATION IS PERFORMED AS FOLLOWS. EACH 3D POINT GIVES 2 EQUATIONS IN 3 UNKNOWNS

$$u = \frac{su}{s} = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

$$v = \frac{sv}{s} = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

4 EQUATIONS IN X,Y,Z. SOLVE BY LEAST-SQUARES.

[15%]

# Engineering Part IIB 2022
# Module 4F12 (Computer Vision) Assessor's Report

1. **Gaussian smoothing, edge detection and SIFT**. Attempted by 88/90 Part IIB candidates, average mark 13.5/20.

   A popular question. When answering Part (b-i), many struggled with low-level details, particularly with computing dimensions for convolution outputs, and understanding trade-offs between memory and computation. However, the majority of candidates did well with applying the key ideas of the course (invariance and nuisance factors) conceptually in (c-iii).

2. **Perspective projection and camera calibration**. Attempted by 83/90 candidates, average mark 13.9/20.

   Students demonstrated a particularly good knowledge of perspective projection and the key steps required for calibration with a known 3D object. Parts (a-i) and (a-iv) were answered exceptionally well. Marks were lost in Parts (b-ii) and (b-iii). Many students were not successful in deriving a 2-D homogenous representation of a line under homography transformation.

3. **Image classification with convolutional neural networks**. Attempted by 23/90 candidates, average mark 13.2/20.

   Most candidates that attempted this question made excellent progress. Many of the marks were lost to mistakes in Part (b) requiring to compute the derivative of the objective function with respect to model parameters and in Part (c-iii) requiring to propose an image classification approach based on image retrieval.

4. **Epipolar geometry and stereo vision**. Attempted by 76/90 candidates, average mark 13.3/20.

   Parts covering epipolar geometry (a) and fundamental matrix estimation (b) from point correspondences were mostly well answered with occasional marks lost for lack of precision or detail. Candidates displayed a particularly good understanding of essential matrix decomposition. Very few candidates noticed that fundamental matrix can not be estimated under pure rotation in Part (b-ii). Many candidates struggled to explain the key steps of structure from motion in Part (c).