4F12 Computer Vision (2023) — Solutions

Q1(a)

(i) Smoothed pixel:

$$S_\sigma(x,y) = \sum_{-n}^{n}\sum_{-n}^{n} I(x-u, y-v)\, g_\sigma(u)\, g_\sigma(v)$$ mark (1)

where $g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$ and is sampled at $(2n+1)$ locat

(size $2n+1$ pixels) (1).

(ii) Low-pass filter, $g_\sigma(x)$

Needed to remove high frequency noise which is amplified by differentiation. (1)

Consider Fourier transform to interpret as a low-pass filter

spatial domain $g_\sigma(x)$ $\rightleftharpoons$ frequency domain $k\, G_{\sigma^{-1}}(\omega) = e^{-\frac{\omega^2\sigma^2}{2}}$ (1)

where $\sigma_1 = \frac{1}{\sigma}$

This is a suitable low-pass filter — gain $\to 0$ as $\omega \to \infty$
— $kG_{\sigma^{-1}}(0) = 1$
$\therefore$ average intensity unchanged

$kG(0) = e^{-0} = 1$ or $\int_{-\infty}^{\infty} g_\sigma(x)\,dx = 1$

Q1(a) (iii) Image pyramid and scale space

Sample $S(x, y, \sigma)$ logarithmically $\left(\text{i.e. } \sigma_i = \sigma_0 2^{i/s}\right)$

(1) — $\sigma_0,\ \sigma_1 = \sigma_0 2^{\frac{1}{3}},\ \sigma_2 = \sigma_0 2^{\frac{2}{3}},\ \sigma_3 = 2\sigma_0 \ \ldots\ldots\ldots\ \sigma_{64\sigma_0}$

(1) — subsample after $s=3$ images since $\sigma_3 = 2\sigma_0$
Subsample to $\frac{1}{4}$ size to produce next octave

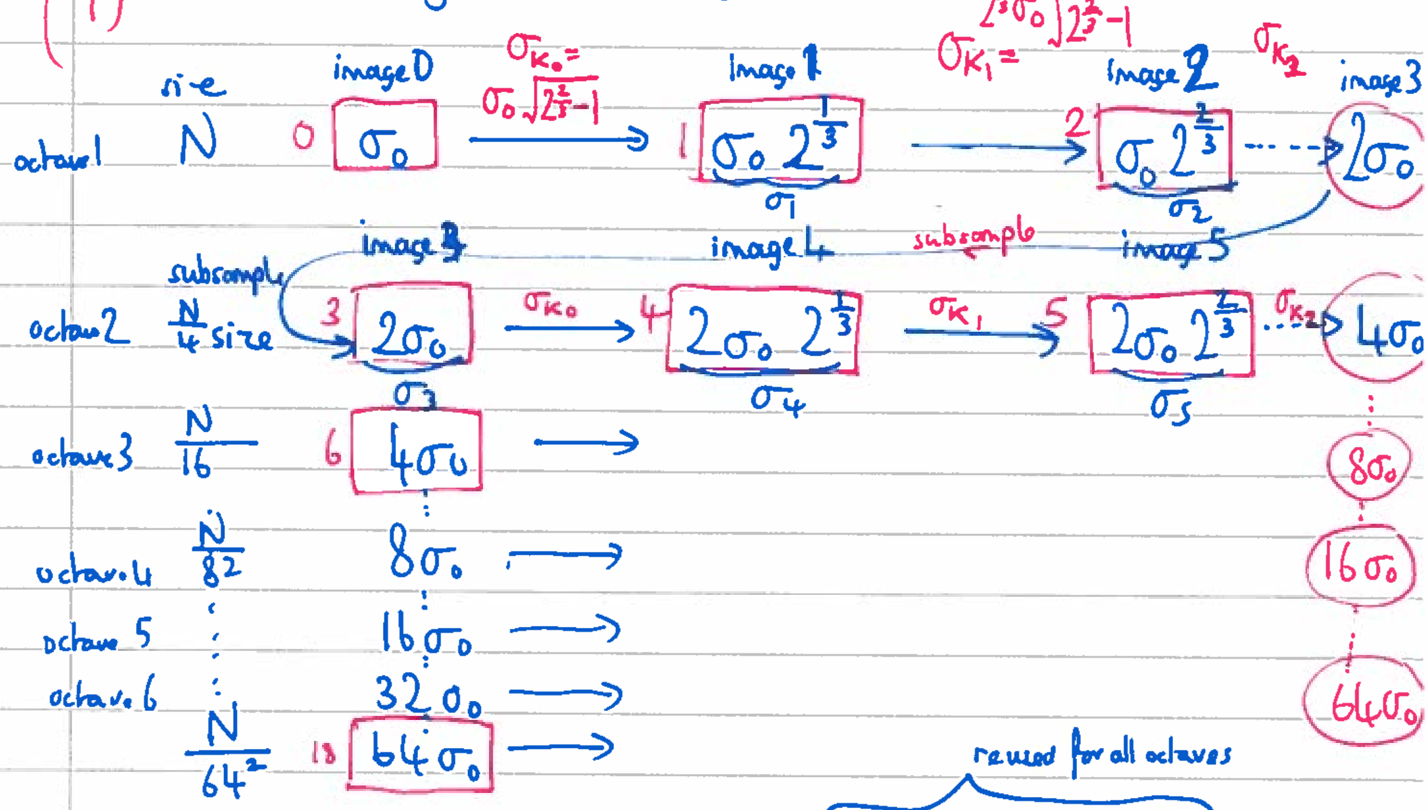
(1) — Within each octave use an incremental blur σ_{K_i}

$$\sigma_{i+1} = \sigma_i * \sigma_{K_i} \qquad \text{where } \sigma_{K_i} = \sigma_i \sqrt{2^{\frac{2}{3}} - 1}$$

incremental blur

These 3 filters are re-used in each octave (need 3 + σ_0).

(1) — Each octave has $s=3$ distinct images, last is subsampled to begin next octave



reused for all octaves

4 distinct filters are needed: σ_0, $\underbrace{\sigma_{K_0},\ \sigma_{K_1},\ \sigma_{K_2}}_{\text{reused for all octaves}}$

Q1 (b) Laplacian of a Gaussian

(i)

$$S(x+\Delta x, y+\Delta y) \simeq S(x,y) + \frac{\partial S}{\partial x}\Delta x + \frac{\partial S}{\partial y}\Delta y + \frac{\partial^2 S}{\partial x^2}\frac{\Delta x^2}{2} + \frac{\partial^2 S}{\partial y^2}\Delta y^2 + \ldots$$

For $\Delta x = 1$, $\Delta y = 1$

$$S(x+1,y) \simeq S(x,y) + \frac{\partial S}{\partial x} + \frac{1}{2}\frac{\partial^2 S}{\partial x^2} + O(\Delta x^3) \ldots$$

$$S(x-1,y) \simeq S(x,y) - \frac{\partial S}{\partial x} + \frac{1}{2}\frac{\partial^2 S}{\partial x^2} \ldots$$

$$\therefore \frac{\partial^2 S}{\partial x^2} \simeq s(x-1,y) - 2s(x,y) + s(x+1,y)$$ (2)

Similarly $\frac{\partial^2 S}{\partial y^2} \simeq S(x,y-1) - 2(x,y) + S(x,y+1)$

$\therefore$ Kernel for $\frac{\partial^2 S}{\partial x^2}$

1	-2	1

1D convolution

(1)

Kernel for $\frac{\partial^2 S}{\partial y^2}$

1
-2
1

1D convolution

(ii) $\nabla^2 S = \frac{\partial^2 S}{\partial x^2} + \frac{\partial^2 S}{\partial y^2} \simeq S(x-1,y) - 4S(x,y) + S(x+1,y) + S(x,y-1) + s(x,y+1)$

Kernel:

0	1	0
1	-4	1
0	1	0

2D convolution

(1)

Q1(b) Laplacian of a Gaussian

(i)

$$S(x+\Delta x, y+\Delta y) \simeq S(x,y) + \frac{\partial S}{\partial x}\Delta x + \frac{\partial S}{\partial y}\Delta y + \frac{\partial^2 S}{\partial x^2}\frac{\Delta x^2}{2} + \frac{\partial^2 S}{\partial y^2}\Delta y^2 + \ldots$$

For $\Delta x = 1$, $\Delta y = 1$

$$S(x+1,y) \simeq S(x,y) + \frac{\partial S}{\partial x} + \frac{1}{2}\frac{\partial^2 S}{\partial x^2} + O(\Delta x^3) \ldots$$

$$S(x-1,y) \simeq S(x,y) - \frac{\partial S}{\partial x} + \frac{1}{2}\frac{\partial^2 S}{\partial x^2} \ldots$$

$$\therefore \frac{\partial^2 S}{\partial x^2} \simeq s(x-1,y) - 2s(x,y) + s(x+1,y)$$ (2)

Similarly $\frac{\partial^2 S}{\partial y^2} \simeq S(x,y-1) - 2(x,y) + s(x,y+1)$

$\therefore$ Kernel for $\frac{\partial^2 S}{\partial x^2}$

1	-2	1

1D convolution

(1)

Kernel for $\frac{\partial^2 S}{\partial y^2}$

1
-2
1

1D convolution

(ii) $$\nabla^2 S = \frac{\partial^2 S}{\partial x^2} + \frac{\partial^2 S}{\partial y^2} \simeq S(x-1,y) - 4S(x,y) + S(x+1,y) + S(x,y-1) + s(x,y+1)$$

Kernel:

0	1	0
1	-4	1
0	1	0

2D convolution

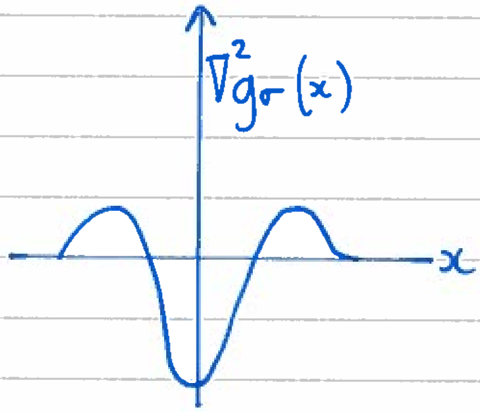
(1)

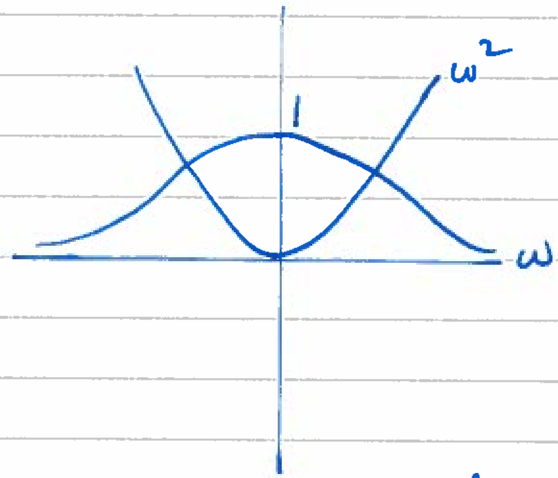
Q1(b)(iii) Band-pass filtering

Interpretation as a band-pass filter is done by considering the Fourier transform of the second derivative of a gaussian

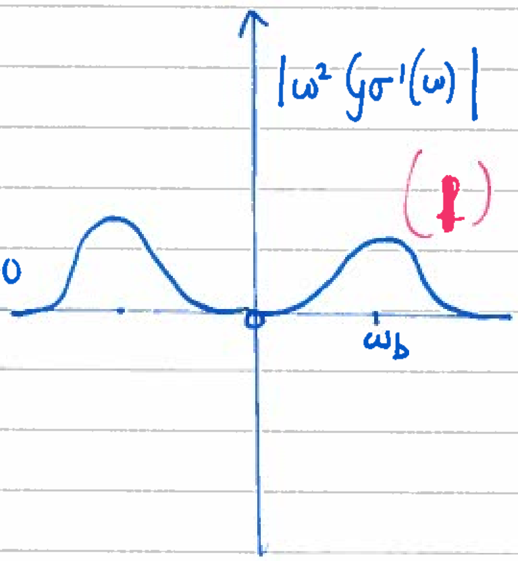
spatial domain in 1D | frequency domain in 1D

$$\nabla^2 g_\sigma(x) = \frac{\partial^2 g_\sigma}{\partial x^2} \overset{FT}{\rightleftharpoons} \omega^2 G_{\sigma'}(\omega) \quad (1)$$





combination of a LP and HP filter



i.e. band-pass filter since gain $= 0$ at $\omega = 0$ and gain $\to 0$ as $\omega \to \infty$

Q1(c)(i)

(i) Edge — intensity discontinuity, ∇S_σ is a maximum

— Approximately localised at zero-crossings of $\nabla^2 S$ (1)

— Look for transition/change in sign of $\nabla^2 G_\sigma * I(x,y)$

— Interpolate to find exact zero-crossing. (1)

(ii) Blob - circular region of uniform intensity

- use band-pass filter matched to central blob of $\nabla^2 G_\sigma(x,y)$ (1)

- Localize at max/min of $\sigma^2 \nabla^2 G_\sigma * I(x,y)$

scale-normalised

— Localize in image and scale by evaluating over scale-space 26 neighbours

— size of blob is given by $\sqrt{2}\,\sigma_i$ (1)

(iii) Edges - invariant to lighting, can localize in position and orientation
- encode 2D shape, i.e. contours
- used in SIFT descriptor → HOGs (1)

Blobs - can be localised in position and scale
- used to give keypoint location and size
- descriptor (128D SIFT) is built around 16x16 pixels using edges
- invariant to scale (1).

Q2(a)(i) Projection matrix

Image co-ordinate (u_i, v_i) can be represented by homogeneous vector

$$\underline{\tilde{\omega}} = \begin{pmatrix} su_i \\ sv_i \\ s \end{pmatrix}$$

World point (X_i, Y_i, Z_i) can be represented by homogeneous vector

$$\underline{\tilde{X}} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} \quad \text{where } X_i = \frac{X_1}{X_4}, \; Y_i = \frac{X_2}{X_4}, \; Z = \frac{X}{X}$$

∴ Perspective projection can be written:

$$su_i = p_{11}X_1 + p_{12}X_2 + p_{13}X_3 + p_{14}X_4$$

$$sv_i = p_{21}X_1 + p_{22}X_2 + p_{23}X_3 + p_{24}X_4$$

$$s = p_{31}X_1 + p_{32}X_2 + p_{33}X_3 + p_{34}X_4$$

or.

$$\underbrace{\begin{pmatrix} su_i \\ sv_i \\ s \end{pmatrix}}_{\tilde{\omega}_i} = \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}}_{\tilde{X}_i}$$

3x4 projection matrix

(2)

Q2(a)(ii)

Vanishing point $\equiv$ perspective projection of point at ∞ in direction of parallel lines

Z-axis lines at ∞ $\qquad \tilde{X} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$

Let $Z_i \to \infty$

$$\tilde{\underline{VP}}_z = \begin{bmatrix} P \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} p_{13} \\ p_{23} \\ p_{33} \end{bmatrix}$$

$$\text{or} \quad \left(p_{13}/p_{33} \,,\; p_{23}/p_{33} \right)$$

(2)

(iii) Horizon $\equiv$ Projection of parallel plane at infinity

$$\tilde{\underline{VP}}_X = \begin{pmatrix} p_{11} \\ p_{21} \\ p_{31} \end{pmatrix} \qquad \underline{VP}_Y = \begin{pmatrix} p_{12} \\ p_{22} \\ p_{32} \end{pmatrix}$$

$$\text{horizon} = \underline{L} = \tilde{\underline{VP}}_X \times \tilde{\underline{VP}}_Y$$

$$= \begin{vmatrix} i & j & k \\ p_{11} & p_{21} & p_{31} \\ p_{12} & p_{22} & p_{33} \end{vmatrix}$$

(2)

Q2(b) Camera calibration

(i) Recover unknown p_{jk} of projection matrix (12 parameters)

and decompose into camera position, T (3)

camera orientation, R (3)

camera intrinsics fk_u, fk_v, u_0, v_0 (4)

$$\begin{bmatrix} su_i \\ sv_i \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} & & \\ & p_{jk} & \\ & & \end{bmatrix}}_{3\times 4} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} su_i \\ sv_i \\ s \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} R & | & T \end{bmatrix}}_{3\times 4} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

(2)

(ii) Each image correspondence (u_i, v_i) gives 2 equations in 12 p_{jk} projection matrix parameters,

$N \geq 6$

Calibration object → large field of view (fills view)

→ large variation in depth (must NOT be coplanar)

→ features easy to localize accurately

(2)

2(b)(iii)

Each calibration point (X_i, Y_i, Z_i) gives 2 equations in unknown p_{jk}.

N points produce 2N equations to solve by least-squares and non-linear optimization.

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_i & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_i & -v_1 \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = 0$$

(2)

- Solve $\underset{2N\times 12}{[A]}\ \underset{12\times 1}{\underline{p}} = 0$ by least-squares or SVD

Find $\underline{p}$ (unit vector) which is smallest eigenvector of A^TA since

$$\lambda_1 \leq \frac{p^TA^TAp}{p^Tp} \leq \lambda_{12}$$

(2)

- Initial linear solution is then optimised by searching for $\underline{p}$ that minimizes the sum of the reprojection errors squared:

$$\min_{\underline{p}} \sum_i (u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2$$

where $\begin{pmatrix}\hat{u}_i \\ \hat{v}_i\end{pmatrix}$ are predicted projections $\underline{p}$

Q2(c) Rays and triangulation

i) Each point (u_i, v_i) constrains (X,Y,Z) to lie on a ray given by intersection of 2 linear equations (planes)

Plane 1: $p_{11}X + p_{12}Y + p_{13}Z + p_{14} = u_i\,(p_{31}X + p_{32}Y + p_{33}Z + p_{34})$

$$0 = (p_{11} - u_i p_{31})X + (p_{12} - u_i p_{32})Y + (p_{13} - u_i p_{33})Z + p_{14} - u_i p_{34}$$

(1)

Plane 2: $p_{21}X + p_{22}Y + p_{23}Z + p_{24} = v_i\,(p_{31}X + p_{32}Y + p_{33}Z + p_{34})$

$$0 = (p_{21} - v_i p_{31})X + (p_{22} - v_i p_{32})Y + (p_{23} - v_i p_{33})Z + p_{24} - v_i p_{34}$$

These are linear equations in (X,Y,Z) and are not parallel planes. They define a ray in space.

(1)

(ii) Write equations in form

$$\begin{array}{c c} & 4 \\ 2 & \left[\quad\quad\quad\right] \end{array} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0$$

(2)

Impossible to solve uniquely from 1 image (2 planes). Add another distinct viewpoint (must have a translation of camera) to give 2 more independent equations so we have

$$\begin{array}{c c} & 4 \\ 4 & \left[\quad\quad\right] \end{array} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0$$ which we can solve by least-squares

Geometrically $\Rightarrow$ triangulation.

(2)

3 (a) This network follows a typical architecture for a CNN. However, instead of using explicit pooling layers it uses convolution with stride two.

Convolutional stage.

Convolutional layers CONV1-4 extract translation invariant features from an image.

Non-linear stage.

The use of non-linear activation functions such as Rectified Linear Unit (ReLU) enables the network to learn complex (non-linear) decision boundaries.

Pooling/subsampling stage.

The convolutional layers CONV1 and CONV3 perform image subsampling in order to encourage learning of feature hierarchies and to reduce number of parameters. They act as learnable weighted average pooling layers.

Fully connected layer

The fully connected layer FC1 forms final features of our proposed network. Note that FC1 layer features are not translation invariant.

Parameter calculation

Detailed calculation of the output shape (OS) of each layer and the corresponding number of parameters (P).

(CONV1, $K = 5 \times 5$, $S = 2$, $C = 16$, $P = 2$, A = ReLU) - OS = $16 \times 16 \times 16$, $P = 5 \cdot 5 \cdot 1 \cdot 16 + 16 = 416$. (CONV2, $K = 3 \times 3$, $S = 1$, $C = 32$, $P = 1$, A = ReLU) - OS = $16 \times 16 \times 32$, $P = 3 \cdot 3 \cdot 16 \cdot 32 + 32 = 4640$. (CONV3, $K = 5 \times 5$, $S = 2$, $C = 64$, $P = 2$, A = ReLU) - OS = $8 \times 8 \times 64$, $P = 5 \cdot 5 \cdot 32 \cdot 64 + 64 = 51264$. (CONV4, $K = 3 \times 3$, $S = 1$, $C = 128$, $P = 1$, A = ReLU) - OS = $8 \times 8 \times 128$, $P = 3 \cdot 3 \cdot 64 \cdot 128 + 128 = 73856$. (FC1, $C = 128$, A = Linear) - OS = 128, $P = 8 \cdot 8 \cdot 128 \cdot 128 + 128 = 1048704$.
Total number of parameters: 1178880.

[20%]

(b) (i) In a Siamese network setup the same network is applied to a pair of training images points, I_1 and I_2. A 128-dimensional embedding vectors $f^1 \in R^{128}$ and $f^2 \in R^{128}$ are obtained for each image.

The contrastive loss is used to train this network. It is defined (for a single pair of inputs) as follows:

$$L = \frac{1}{2} s D^2 + \frac{1}{2}(1-s)\left(\max\left(0, \Delta - D\right)\right)^2,$$

where $D = \sqrt{\sum_d \left(f_d^1 - f_d^2\right)^2}$, $s = 0$ is a pair is of images of the same person and $s = 1$, otherwise. Δ is a margin parameter.

At test time the a single image is used to obtain its embedding vector and find the nearest neighbour in the database of the embedding vectors computed from face images of known individuals. [10%]

(ii) We have:
$a_{i,j,c} = \sum_{k,l,c'} w^c_{k,l,c'} x_{i-k,j-l,c'}$ and $y_{i,j,c} = \max\left(0, a_{i,j,c}\right)$ as well as $f_d = \sum_{i,j,c} w^f_{d,(i,j,c)} y_{i,j,c}$. Here (i,j,c) - is an single value index corresponding to the output unit $y_{i,j,c}$ in the CONV4 layer.

We also denote a^o,y^o,f^o, where $o \in \{1,2\}$ to be the intermediate outputs of the Siamese network for the first ($o = 1$) and second ($o = 2$) pair respectively.

Applying chain rule:

$$\frac{\partial L}{\partial w^c_{k,l,c'}} = \frac{\partial L}{\partial D} \sum_{o \in \{1,2\}} \sum_d \frac{\partial D}{\partial f^o_d} \sum_{i,j,\tilde{c}} \frac{\partial f^o_d}{\partial y^o_{i,j,\tilde{c}}} \frac{\partial y^o_{i,j,\tilde{c}}}{\partial a^o_{i,j,\tilde{c}}} \frac{\partial a^o_{i,j,\tilde{c}}}{\partial w^c_{k,l,c'}}.$$

Partial derivatives:

$\frac{\partial L}{\partial D} = 2Ds + (1-s)\max\left(0, \Delta - D\right) \cdot (-1) = \begin{cases} Ds & \text{if } D \geq \Delta, \\ D - \Delta(1-s) & \text{otherwise,} \end{cases}$

$\frac{\partial D}{\partial f^o_d} = \frac{1}{2}\frac{1}{D} \cdot 2 \cdot \left(f^1_d - f^2_d\right)(-1)^{1-o} = \frac{1}{D}\left(f^1_d - f^2_d\right)(-1)^{1-o}$,

$\frac{\partial f^o_d}{\partial y^o_{i,j,\tilde{c}}} = w^f_{d,(i,j,\tilde{c})}$ and $\frac{\partial y^o_{i,j,\tilde{c}}}{\partial a^o_{i,j,\tilde{c}}} = \mathbb{1}\left[a^o_{i,j,\tilde{c}} > 0\right]$ and $\frac{\partial a^o_{i,j,\tilde{c}}}{\partial w^c_{k,l,c'}} = x^o_{i-k,k-l,\tilde{c}} \mathbb{1}\left[\tilde{c} = c'\right]$.

We have $\sum_{o\in\{1,2\}} \sum_d \frac{\partial D}{\partial f_d^o} \sum_{i,j,\tilde{c}} \frac{\partial f_d^o}{\partial y_{i,j,\tilde{c}}^o} \frac{\partial y_{i,j,\tilde{c}}^o}{\partial a_{i,j,\tilde{c}}^o} \frac{\partial a_{i,j,\tilde{c}}^o}{\partial w_{k,l,c'}^c} =$
$= \frac{1}{D} \sum_d \left(f_d^1 - f_d^2\right) \sum_{i,j} w_{d,(i,j,c')}^f \left(\mathbb{1}\left[a_{i,j,c'}^1 > 0\right] x_{i-k,j-l,c'}^1 - \mathbb{1}\left[a_{i,j,c'}^2 > 0\right] x_{i-k,j-l,c'}^2\right) = \frac{1}{D} v_{k,l,c'}^c$.

Hence, $\frac{\partial L}{\partial w_{k,l,c'}^c} = \begin{cases} s v_{k,l,c'}^c & \text{if } D \geq \Delta, \\ (1 - \frac{\Delta}{D}(1-s)) v_{k,l,c'}^c & \text{otherwise.} \end{cases}$ [25%]

(iii) (1) It seems the network is failing to generalise, hence a couple of dropout layers should be used in between the convolutional networks. The dropout layer works by multiplying each input unit by 0 with a probability, α (e.g. $\alpha = 0.5$). At test time usually drop out is not used so the dropout layer should scale its input by multiplying it by α.

(2) The increasing brightness of images may have a multiplicative effect on the embedding vectors hence they should be normalised (ie. similar to SIFT feature descriptors). [15%]

(c) Advantages and disadvantages.

(1) **Advantage: modelling long range correlations.** The attention operation considers all the input pixels when producing each output value. This allows to the full size receptive field from the very start of the architecture. This may allow to easier capture the long range correlations, e.g. such as a specific distance between the nose and eyes. This information may be captured only after quite a few layers in a typical convolutional network, potentially requiring deeper networks with more parameters to solve the same problem.

(2) **Advantage: input to the transformer networks are unstructured sets.**
CNN's assume that the data is presented on some fixed grid. This may not be the case for many types of inputs. E.g. if one wanted to integrate other information along with the face image such as a recording of a voice of a person, it can be seamlessly performed by just concatenating the input of the rgb image pixels and chosen representation of sound. Also transformer architectures can be more efficient in leveraging sparse input (e.g. if one uses only edges or face landmarks for face recognition) since they would only use the number of input data points provided as opposed to the full size grid.

(3) **Disadvantage: quadratic time and space complexity in terms of input size.** A vanilla transformer network built of MHA and Layer Normalisation layers would have a quadratic cost in the input size (e.g. $(WH)^2$ which may be prohibitive for many

applications. The time/space complexity in convolutional networks is much smaller WHK^2 where K is the convolutional kernel size. This drawback of transformers can be address by for example projecting the input into a smaller dimensional space as done by the linear projection layer in the ViT. [10%]

(d) Pre-training and data
Since the dataset containing the group image segmentation/recognition information is relatively small, we first train the original network in Fig. 1. Note that the original training setup is likely produce a network which is sensitive to recognising faces that are not well centered. Hence we may include crops of the original images where the center pixel is near the edge of the face with the correct identity. The newly added part of the image can be filled in with random noise, uniform color (e.g. black) or crops of background from the provided dataset.

Architecture. The architecture shown in Fig. 1 is adapted by: replacing the final fully connected layer with the deconvolution layer without a non-linearity which upsamples the output from $8 \times 8 \times 128$ to $32 \times 32 \times 128$. This makes the network fully convolutional, hence - very efficient in producing per pixel 128-dimensional embedding vectors. During the finetuning stage the rest of the network of Fig.1 is initialised with pretrained weights and are frozen. Only the the deconvolution layer parameters are trained.

Test use. At test time we find the closest embedding in the database to each pixel in the image to produce a per pixel face identity label. To speed up this step the per-pixel embeddings may be clustered (e.g. using mean-shift) and only cluster centers used to retrieve the identity.

Objective function and fine-tuning. As with image-level face recognition task, the contrastive loss is used in the fine-tuning stage. However, it should be applied per pixel and not for the whole image.

Also there is a slight change in the sampling procedure for negative and positive pairs. In the fine-tuning stage for each pixel we take a positive or negative example by randomly sampling any pixel in the batch of image which has the same or different label id. The background pixels can be grouped into their own class in order to not only be able to recognise face pixels but also segment out a background class.

Colour normalization, rotation, horizontal flips, random crops and scales, colour jitter, randomized contrast and brightness and other data augmentation techniques can be employed.

Also it is important to note that all geometric (e.g. rotation, flip, etc.) augmentation steps performed on the input image should be applied on the corresponding target per-pixel class label images.

[20%]

Q4

(a) Epipolar constraint — derivation:

(i) Consider rigid-body motion of camera: $\underline{X}' = R\underline{X} + \underline{T}$

$$\underline{T}_x \underline{X}' = \underline{T}_x R \underline{X} + \underbrace{\underline{T}_x \underline{T}}_{\to 0}$$

Rays and baseline are coplanar

$$\underline{X}' \cdot \underline{T}_x [R] \underline{X} = 0$$

Define $\underline{T}_x = [T_x] = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$ and let $E = [T_x][R]$

$$\therefore \underline{X}' \underset{3\times3}{[T_x]} \underset{3\times3}{[R]} \underline{X} = 0$$

$$\underline{X}' E \underline{X} = 0 \quad \text{epipolar constraint} \quad (2)$$

In rays $\underline{p}' // \underline{X}'$ and $\underline{p} // \underline{X}$ where $\underline{p} = \dfrac{f\underline{X}}{Z}$

$$\therefore \underline{p}' E \underline{p} = 0 \qquad \underline{p}' = \frac{f\underline{X}'}{Z'}$$

In pixel co-ordinates $\underline{\omega} = K\underline{p}$ and $\underline{\omega}' = K'\underline{p}'$

$$\therefore \underline{\tilde{\omega}}'^T K'^{-T} E K^{-1} \underline{\tilde{\omega}} = 0 \quad (2)$$

$$\underline{\tilde{\omega}}'^T F \underline{\tilde{\omega}} = 0 \quad \text{where } \underline{\underline{F = K'^{-T} E K^{-1}}}$$

4(a)(ii)

Consider point in left view $\underline{\tilde{w}} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$ then epipolar constraint can be rewritten:

$$\underline{\tilde{w}}'^T \underline{\tilde{l}}' = 0 \quad \text{where} \quad \underline{\tilde{l}}' = F\underline{\tilde{w}}$$ (2)

$\underline{\tilde{l}}'$ is the line in the right view on which correspondence $(\underline{\tilde{w}}')$ must lie.

4(b) Viewing conditions for $\underline{\tilde{w}}' = \underset{3\times 3}{H} \underline{\tilde{w}}$

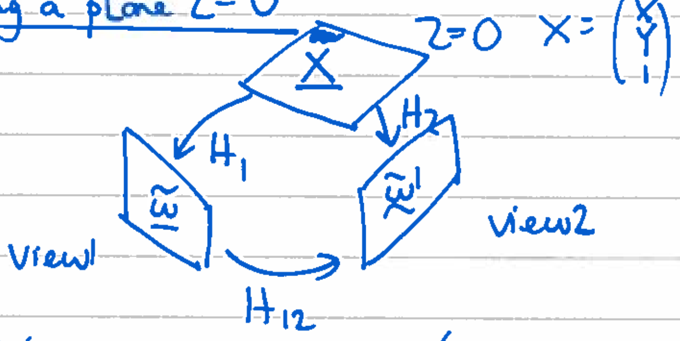
(i) Rotation about optical axis $\underline{T} = 0$

$$\underline{\tilde{w}} = K\underline{X} \quad \text{and} \quad \underline{\tilde{w}}' = KR\underline{X}'$$

$$\therefore \underline{\tilde{w}}' = \underbrace{KRK^{-1}}_{H}\underline{w}.$$ (1)

marks (2)

(ii) Viewing a plane $Z=0$



marks (2)

$$\underline{\tilde{w}} = H_1\underline{X} \qquad \underline{\tilde{w}}' = H_2\underline{X} \qquad \underline{\tilde{w}}' = \overbrace{H_2H_1^{-1}}^{H}\underline{\tilde{w}}$$ (2)

4(c) Estimating multi-view transformations, F and H

(i). Keypoints detected in each view and matched by comparing descriptors.

Keypoint detection:
- find blob centres and (scale) sizes by image pyramid (scale invariance)
- find dominate orientation → orientation invariance
- compute 16x16 gradients and 16 HOGs
- 128D vector normalised to unit vector

Keypoint description
- 128D descriptor (SIFT) $\underline{x}$
- robust to photometric changes $|\underline{x}| = 1$

(2)

Matching
- Find nearest neighbour match – search $\underline{x}_i$ using kd tree for nearest euclidean distance

$$d_{im} = \sum_{1}^{128} \left(\overline{\underline{x}_i} - \underline{x}_m^2\right) = \|\underline{\tilde{x}}_i - \underline{\tilde{x}}_m\|$$

- accept nearest neighbour match (shortest distance) if nearest neighbour $\underline{x}_1$ is much closer than next neighbour $\underline{x}_2$ using RATIO test:

$$\frac{|\underline{x}_1^2 - \underline{x}_m^2|}{|\underline{x}_2^2 - \underline{x}_m^2|} < 0.7$$

(2)

Q4(c)(ii) RANSAC

N = 4 or N = 8
Homography Fundamental matrix

- Randomly sample N points and their correspondences (NN in SIFT space)
- Compute F or H using min. pts (N=8 or 4)
- Check for inliers and count
- Accept solution if large # inliers

- Perform least-squares of inliers to estimate F or H

(2)

Q4(c)(iii)

Set up linear equations for inliers:

$$\underset{N\times 9}{A}\ \underset{9\times 1}{\underline{f}} = 0$$

where each row of A: $\left[\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \right] \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ \vdots \\ f_{33} \end{bmatrix}$

(2)

Solve by least-squares $\lambda_1 \leq \dfrac{\underline{f}^T A^T A \underline{f}}{\underline{f}^T \underline{f}} \leq \lambda_9$

to find smallest eigenvector, $\underline{f}$, corresponding to smallest eigenvalue λ_1.
This is linear estimate $\hat{\underline{F}}$ but doesn't enforce non-linear constraints. det F=(

Q4(c)(iii) cont..

Project $\hat{F}$ to nearest F' which has $\det F' = 0$ (ie rank 2)
(2 singular values)

Decompose F', $E = K'^T F' K$

Use SVD to get R and T (4 ambiguity, resolve by positive depth)

(2)

RC 19/5/2023