Institut für Mess- und Regelungstechnik
mit Maschinenlaboratorium
Karlsruher Institut für Technologie
(KIT)
Prof. Dr.-Ing. C. Stiller

Solutions for exam "Machine Vision" February 20, 2023

Question 1 (5 points)

Provide the 4-by-4 binomial filter.

Solution

$$\frac{1}{(2^3)^2} \cdot \begin{pmatrix} 1 \\ 3 \\ 3 \\ 1 \end{pmatrix} * \begin{pmatrix} 1 & 3 & 3 & 1 \end{pmatrix} = \frac{1}{64} \begin{pmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{pmatrix}$$

Question 2 (6 points)

The figures below represent a gray level image and a kernel. Apply the convolution between the kernel and the gray level image. Do not use any padding (leave out boundary pixels).

Note down your calculation steps.

Gray Level Image

20	30	50	80	200
10	20	80	70	110
10	40	50	50	160

Kernel

0	0	0
0	0	1
0	2	4

Solution

2P for each correct value.

$$150 = ((4 * 20) + (2 * 30) + (0 * 50) + (1 * 10) + (0 * 20) + (0 * 80) + (0 * 10) + (0 * 40) + (0 * 50))$$

$$240 = ((4 * 30) + (2 * 50) + (0 * 80) + (1 * 20) + (0 * 80) + (0 * 70) + (0 * 40) + (0 * 50) + (0 * 50))$$

$$440 = ((4 * 50) + (2 * 80) + (0 * 200) + (1 * 80) + (0 * 70) + (0 * 110) + (0 * 50) + (0 * 50) + (0 * 160))$$

Solution

	150	240	440	

Question 3 (8 points)

Calculate the line that fits a set of points using the total least squares approach. The set of points is (0,0), (1,0), and (1,1), where the point coordinates are given as (x_1, x_2).

Remark: You can calculate the eigenvalue by zeroing the characteristic polynom:
$det(B - I\lambda) = 0$,
where the determinant of a 2x2 matrix is calculated by:
$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$.

Solution

We use the formulae from slide 04/27-30. We obtain

$$\begin{aligned} N &= 3 \\ \sum_i x_{i,1} &= 2 \\ \sum_i x_{i,2} &= 1 \\ \sum_i (x_{i,1})^2 &= 2 \\ \sum_i (x_{i,1} x_{i,2}) &= 1 \\ \sum_i (x_{i,2})^2 &= 1 \end{aligned}$$

Hence, we have to find the Eigenvalues of the matrix

$$B = \begin{pmatrix} 2 - \frac{4}{3} & 1 - \frac{2}{3} \\ 1 - \frac{2}{3} & 1 - \frac{1}{3} \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix}$$

By zeroing the charecteristic polynomial we obtain

$$det(B - \lambda I) = (\frac{2}{3} - \lambda) - (\frac{1}{3})^2 = 0 \quad \text{if and only if} \quad \lambda \in \{\frac{1}{3}, 1\}$$

We determine the Eigenvector of B with respect to the smaller Eigenvalue $\lambda = \frac{1}{3}$

$$\begin{pmatrix} \frac{2}{3} - \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} - \frac{1}{3} \end{pmatrix} \cdot \vec{n} = 0 \quad \Rightarrow \quad \vec{n} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

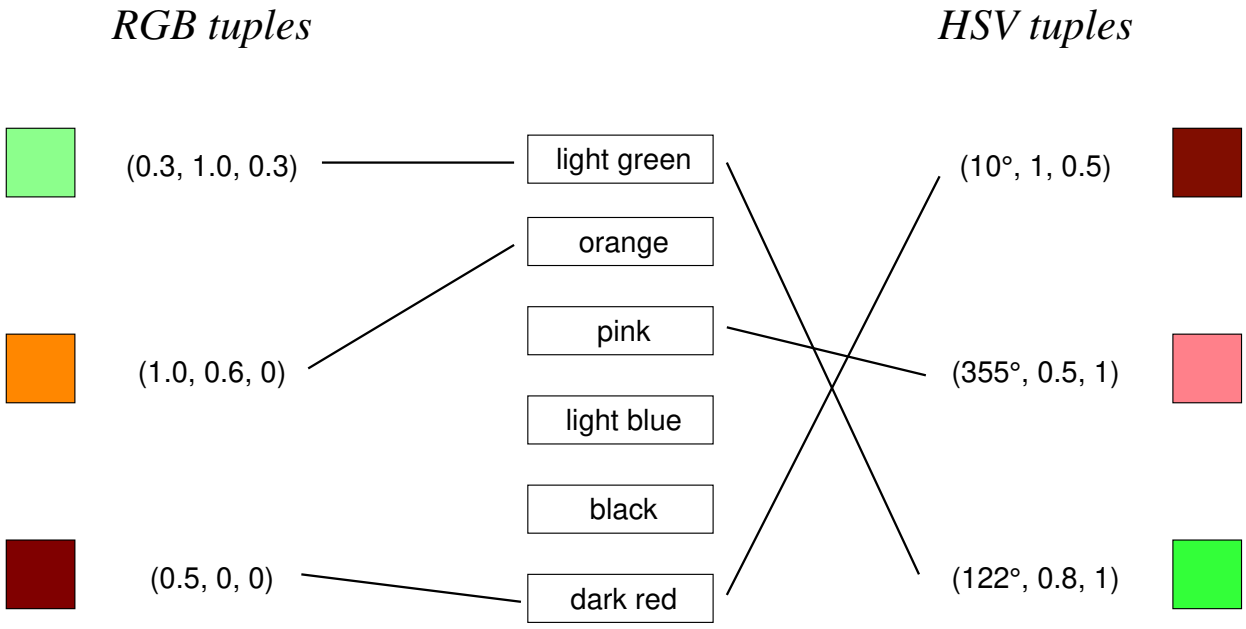
We obtain c as

$$c = -\langle \vec{n}, \frac{1}{N} \sum_i \vec{x_i} \rangle = -\frac{1}{6}\sqrt{2}$$

Question 4 (6 points)

Below you find six words that describe color, three RGB tuples (min:0, max:1) on the left and three HSV tuples on the right. Connect each RGB tuple with the word that describes its color. Connect each HSV tuple with the word that describes its color. Mind that some words might be connected with several RGB and HSV tuples while other words might not be connected to any.

Solution



Question 5 (5 points)

Assume that during a camera calibration process the 3d positions of the markers have not been measured. However, the distance of two detected markers is known as $d = 25$ cm. The markers are located in the image at $\vec{p_1}$ and $\vec{p_2}$. The calibration board is assumed to be parallel to the image plane. The intrinsic parameters of the camera are given by matrix $\mathbf{A}$. For simplification, a pinhole camera model is assumed.

Determine the distance to the calibration board.

$$\mathbf{A} = \begin{pmatrix} 800 & 0 & 1000 \\ 0 & 800 & 500 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\vec{p_1} = \begin{pmatrix} 300 \\ 381 \end{pmatrix}, \quad \vec{p_2} = \begin{pmatrix} 350 \\ 381 \end{pmatrix}$$

The points $\vec{p_1}$ and $\vec{p_2}$ are given in $\begin{pmatrix} u \\ v \end{pmatrix}$ format.

Solution

$$zu = \mathbf{A}x$$
$$zu_1 = \mathbf{A}x_1$$
$$zu_2 = \mathbf{A}x_2$$

$$z * u_1 = 800x_1 + 1000z$$
$$z * 300 = 800x_1 + 1000z$$
$$0 = 800x_1 + 1000z - z300$$

$$z * u_2 = 800x_2 + 1000z$$
$$z * 350 = 800x_2 + 1000z$$
$$0 = 800x_2 + 1000z - z350$$

$$800x_1 + 1000z - z300 = 800x_2 + 1000z - z350$$
$$800x_1 - 800x_2 - z300 + z350 + 1000z - 1000z = 0$$
$$800(x_1 - x_2) + z(350 - 300) = 0$$
$$800(x_1 - x_2) + z(350 - 300) = 0$$
$$z = -800\frac{(x1 - x2)}{(350 - 300)}$$
$$z = -800\frac{(25cm)}{(50)}$$
$$z = -800\frac{1}{2}cm = 400cm$$

Question 6 (2+4 points)

In level set methods for image segmentation the Mumford-Shah-based segmentation approach uses the evolution term

$$\frac{\partial \vec{x}}{\partial t} = \frac{\nabla \phi}{||\nabla \phi||} \cdot \big(-\beta\kappa - \alpha - \lambda_1 (g - \bar{g}_{foreground})^2 + \lambda_2 (g - \bar{g}_{background})^2 \big)$$

where g determines the gray or color value of the image, $\kappa = \nabla \left(\frac{\nabla \phi}{||\nabla \phi||} \right)$ denotes the local curvature of the boundary, and $\alpha \in \mathbb{R}, \beta, \lambda_1, \lambda_2 \in \mathbb{R}_{\geq 0}$ are tuning parameters.

(a) Assume that we choose $\lambda_1 = \lambda_2 = 1$ and $\alpha = 0$. How does the choice of β influence the result? Compare the cases $\beta = 0$ and $\beta \gg 1$

(b) Assume that we know that the foreground segment should cover approximately $k \in \mathbb{N}$ pixels. Denote with n the present size of the foreground segment in pixels. Extend the Mumford-Shah evolution term to foster foreground segements with the desired size $n \approx k$ over segments with very different sizes $n \not\approx k$. Explain your solution briefly.

Solution

(a) β influences the smoothness of the segment boundary. For $\beta = 0$ the resulting segment will have a ragged boundary. For $\beta \gg 1$ the smoothness term will be very influential so that the resulting segment will have a very smooth boundary.

(b) The idea is to prefer shrinking the foreground segment when it is too large and expanding it when it is too small. We can achieve this behavior with the dynamics

$$\frac{\partial \vec{x}}{\partial t} = \gamma \cdot (k - n) \cdot \frac{\nabla \phi}{||\nabla \phi||} \text{ with } \gamma \geq 0$$

γ is a parameter with which we can tune the influence of this evolution term. If $n < k$ the vector $\frac{\partial \vec{x}}{\partial t}$ points to the background so that the segment boundary will be expanded. If $n > k$ the vector $\frac{\partial \vec{x}}{\partial t}$ points to the foreround so that the segment boundary will be shrunk. Combining this term with the Mumford-Shah formula yields the dynamics

$$\frac{\partial \vec{x}}{\partial t} = \frac{\nabla \phi}{||\nabla \phi||} \cdot \big(-\beta\kappa - \alpha - \lambda_1 (g - \bar{g}_{foreground})^2 + \lambda_2 (g - \bar{g}_{background})^2 + \gamma \cdot (k - n) \big)$$

Question 7 (5 points)

Prove or disprove whether the following functions can serve as kernel functions in a support vector machine.

(a) $f(\vec{x}, \vec{y}) = e^{||\vec{x}||+||\vec{y}||}$ where $||\vec{x}||$ denotes the Euclidean norm of vector $\vec{x}$.

(b) $g(x,y) = x^2y + y$

Solution

f is a valid kernel since it can be written as inner product

$$f(\vec{x},\vec{y}) = e^{||\vec{x}||} \cdot e^{||\vec{y}||} = \phi(\vec{x}) \cdot \phi(\vec{y})$$

with the non-linear transfom $\phi : \vec{x} \mapsto e^{||\vec{x}||}$.

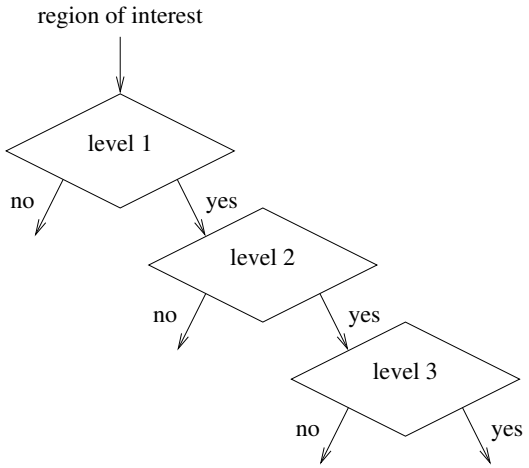
g is not a kernel function since it is not symmetric in its arguments and thus cannot be explained as inner product in feature space.

Question 8 (6 points)

Build a cascade classifier with three levels to detect certain objects in an image. The structure of the cascade is shown in the illustration below. For that purpose you can use the six classifiers described in the table below. Assign to each of the three levels the best classifier from the table. The cascade should be as accurate as possible and it should not require more computational effort than necessary to achieve that accuracy.

For each classifier briefly explain why you have chosen the classifier for a specific level.

classifier	precision	recall	execution time per region of interest
A	82%	55%	0.001 ns
B	49%	98%	8.9 µs
C	53%	98%	6.3 µs
D	92%	87%	1.2 ms
E	99%	28%	0.01 ns
F	34%	100%	0.2 ns



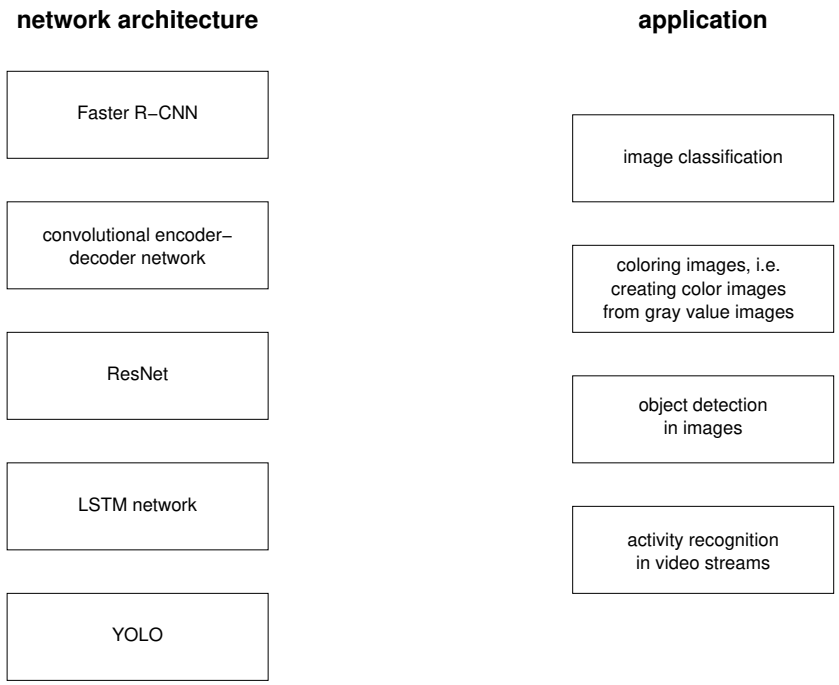
Solution

F (level 1), C (level 2), D (level 3)

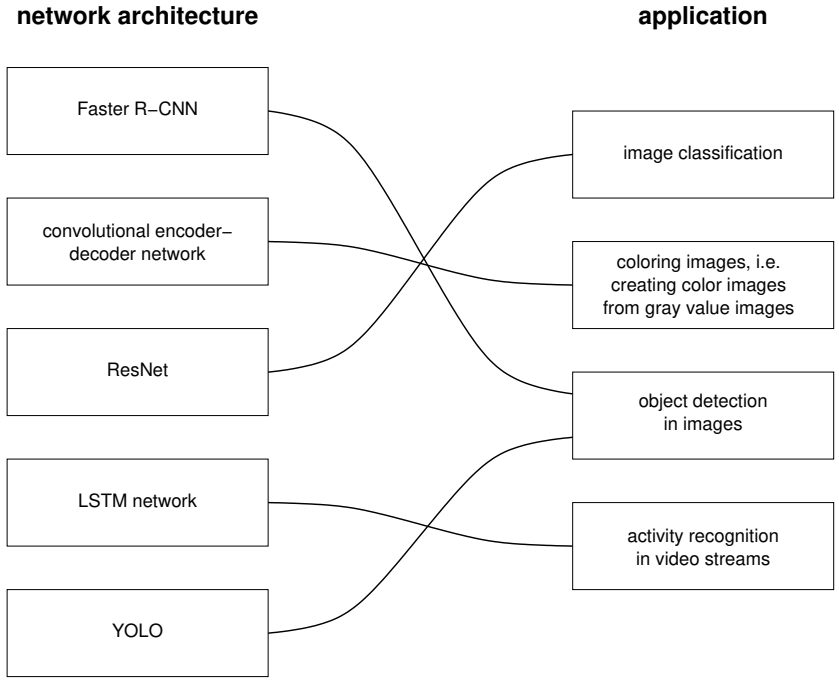
Question 9 (5 points)

Which of the following deep network architectures can be used for which applications? Draw a line between each network architecture and the respective application.

Remark: The given networks should be used without any additional heads or modifications.



Solution



Question 10 (8 points)

During the backpropagation of a multi-layer-perception its weights are changed to optimize an error term.
Implement the Python function `change_weights` that implements this process using gradient descent with momentum.

The function should have exactly 5 arguments, the weights of one layer (`weights`), the gradient of the error term with respect to these weights (`gradients`), the previous momentum term (`momentum`), and the hyperparameters epsilon (`epsilon`) and alpha (`alpha`). The function should update the weights and the momentum term and return both.
Provide type hints at your function definition.

Remark: If you use any kind of array or tensor in your solution, use Numpy arrays.

```
import numpy as np
```

Solution

```
def change_weights(
  weights: np.ndarray,
  gradients: np.ndarray,
  momentum: np.ndarray,
  epsilon: float,
  alpha: float,
):
  momentum = alpha * momentum - epsilon * gradients
  weights = weights + momentum

  return weights, momentum
```

Gesamtpunkte: 60