1. For each pixel in the image frame, multiply it with its corresponding kernel value, sum the products, and set the current pixel to this value.
2. For each pixel in the kernel, visit each pixel in the image, multiply their values, sum the products, and set the current pixel to this value.
3. For each row in the image, visit each pixel in the row, apply the kernel values to the neighboring pixels, and set the current pixel to this value.
4. For each row in the kernel, visit each pixel in the row, apply the kernel values to the neighboring pixels, and set the current pixel to this value.

Answer:

---

(9) Which one of the following kernels can be used to blur an image? Explain Why? (10 pts)

$$(A)\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix}\quad(B)\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\quad(C)\begin{bmatrix}1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\end{bmatrix}\quad(D)\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

Answer:

|

|

|

---

(9) Fill in the missing comments for each OpenCV function in the Sobel pipeline, explaining the purpose of each step. The (10 pts - 1 pt each)

Comment = #

```
#
#
img = cv2.imread('Graphics/face_conv.png')


#
#
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


#
#
kernelOne = np.array([[1, 2, 1], [2, 4, 2], [1, 2, 1]],dtype=np.float32) / 16


#
#
```

```
filterOneImage = cv2.filter2D(KernelOne, cv2.CV_64F, gray)


#
#
kernelX = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=np.float64)


#
#
KernelY = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=np.float64)


#
#
imageX = cv2.filter2D(filterOneImage, cv2.CV_64F, kernelX)

#
#
imageY = cv2.filter2D(filterOneImage, cv2.CV_64F, kernelY)


# Compute the magnitude of the gradients of imageX and imageY
mag = cv2.magnitude(imageX, imageY)


#
#
val, thresh = cv2.threshold(mag, 100, 200, cv2.THRESH_BINARY)

#
#
plt.imshow(thresh, cmap='gray')
```