

# HW 10: Zero Crossing Edge Detection

## Source Code

---

All questions are written in Python code, please refer to the file “main.py”.

All images will be stored in the folder “res” (automatically create a new folder).

In accordance with the **FAQ** of course website:

- All parts of the question are written from scratch, except for plotting images

## Answer

---

### 1. Laplacian's Mask 1, Laplacian's Mask 2, Minimum Variance Laplacian

```
def laplacianMask(source, kernel, threshold):
    padded = padding(source, 1)
    result = np.zeros(source.shape, dtype=int)
    for i in range(1, padded.shape[0]-1):
        for j in range(1, padded.shape[1]-1):
            # Region of interest
            box = []
            for x in range(3):
                for y in range(3):
                    xdest = i + x - 1
                    ydest = j + y - 1
                    box.append(padded[xdest][ydest])
            # Multiply RoI with kernel
            total = 0
            for x in range(len(kernel)):
                total += (kernel[x] * box[x])
            total = round(total, 2) # to avoid floating point precision error
            # Insert to mask
            if total >= threshold:
                result[i-1][j-1] = 1
            elif total <= -threshold:
                result[i-1][j-1] = -1
            else:
                result[i-1][j-1] = 0
    return result

def problemABC(source, kernel, threshold):
    mask = laplacianMask(source, kernel, threshold)
    result = zeroCrossing(mask)
    return result
```



Left -> Right: Laplacian's Mask 1, Laplacian's Mask 2, Minimum Variance Laplacian

## 2. Laplace of Gaussian, Difference of Gaussian

```
def lgMask(source, kernel, threshold):
    padded = padding(source, 5)
    result = np.zeros(source.shape, dtype=int)
    for i in range(5, padded.shape[0]-5):
        for j in range(5, padded.shape[1]-5):
            # Region of interest
            box = []
            for x in range(11):
                for y in range(11):
                    xdest = i + x - 5
                    ydest = j + y - 5
                    box.append(padded[xdest][ydest])

            # Multiply RoI with kernel
            total = 0
            index = 0
            for x in range(11):
                for y in range(11):
                    total += (kernel[x][y] * box[index])
                    index += 1

            # Insert to mask
            if total >= threshold:
                result[i-5][j-5] = 1
            elif total <= -threshold:
                result[i-5][j-5] = -1
            else:
                result[i-5][j-5] = 0
    return result

def problemDE(source, kernel, threshold):
    mask = lgMask(source, kernel, threshold)
    result = zeroCrossing(mask)
    return result
```



Left -> Right: Laplace of Gaussian, Difference of Gaussian

### 3. Zero Crossing

```
def zeroCrossing(mask):  
    padded = padding(mask, 1)  
    result = np.zeros(mask.shape, dtype=int)  
    for i in range(1, padded.shape[0]-1):  
        for j in range(1, padded.shape[1]-1):  
            if padded[i][j] != 1:  
                result[i-1][j-1] = 255  
            else:  
                cross = 1  
                for x in range(3):  
                    if cross == 1:  
                        for y in range(3):  
                            xdest = i + x - 1  
                            ydest = j + y - 1  
                            if padded[xdest][ydest] == -1:  
                                cross = 0  
                if cross == 1:  
                    result[i-1][j-1] = 255  
            else:  
                result[i-1][j-1] = 0  
    return result
```