Kernel Operations

NOAH SHERRY

Input Images

image1.png

image2.png

Box Blur: 5x5

- Develop and use a 5x5 box blur kernel.
- Kernel: [[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1]]
- Implementation:

```
return cv2.filter2D(img, -1, kernel * (1/25))
```

Box Blur: Split Kernels

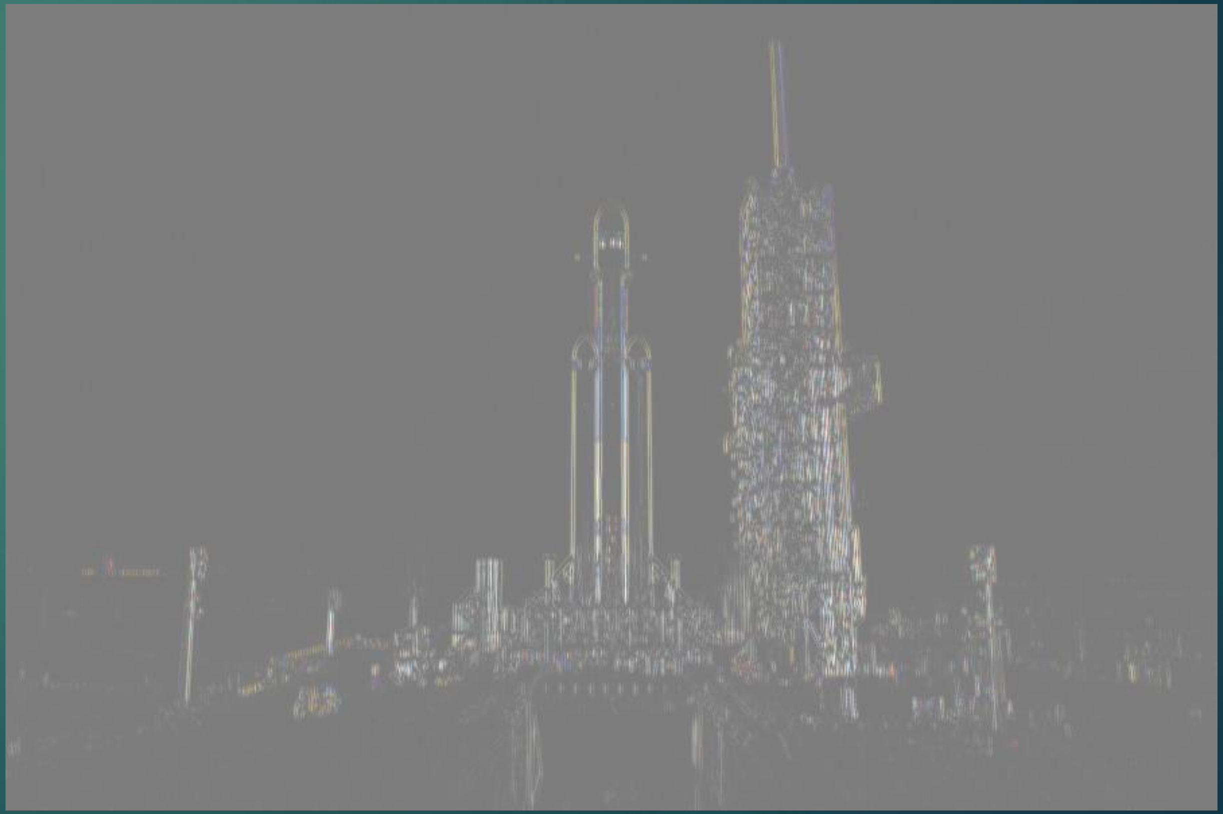
- Develop and use 1x5 & 5x1 box blur kernels.
- 1x5 Kernel: [1,1,1,1,1]
- 5x1 Kernel: [[1],[1],[1],[1],[1]]
- Implementation:

```
kernel_0 = np.float32(
        [[1],[1],[1],[1],[1]]
    ) * (1/5)
    kernel_1 = kernel_0[:,None]
    kernel_mix = kernel_0.dot(kernel_1)
    return cv2.filter2D(img, -1, kernel_mix)
```

Box Blur: Diff

- Create an image difference of the images produced from the previous slides.
- Implementation:

```
image1 = img1*1.0
    image2 = img2*1.0
    dif = np.abs(image1-image2)
    dif -= dif.min()
    dif += dif.max()
    dif += 255
    return np.uint8(dif)
```



Gaussian Blur: 5x5

- Develop and use a 5x5 gaussian blur kernel.
- Kernel: [[1,4,6,4,1],[4,16,24,16,4],[6,24,36,24,6],[4,16,24,16,4],[1,4,6,4,1]]
- Implementation:

```
return cv2.filter2D(img, -1, kernel * (1/256))
```

Gaussian Blur: Split Kernels

- Develop and use 1x5 & 5x1 Gaussian blur kernels.
- 1x5 Kernel: [1,4,6,4,1]
- 5x1 Kernel: [1],[4],[6],[4],[1]
- Implementation:

```
kernel_0 = np.float32(
        [[1],[4],[6],[4],[1]]
    ) * (1/16)
    kernel_1 = kernel_0[:,None]
    kernel_mix = kernel_0.dot(kernel_1)
    return cv2.filter2D(img, -1, kernel_mix)
```

Gaussian Blur: Diff

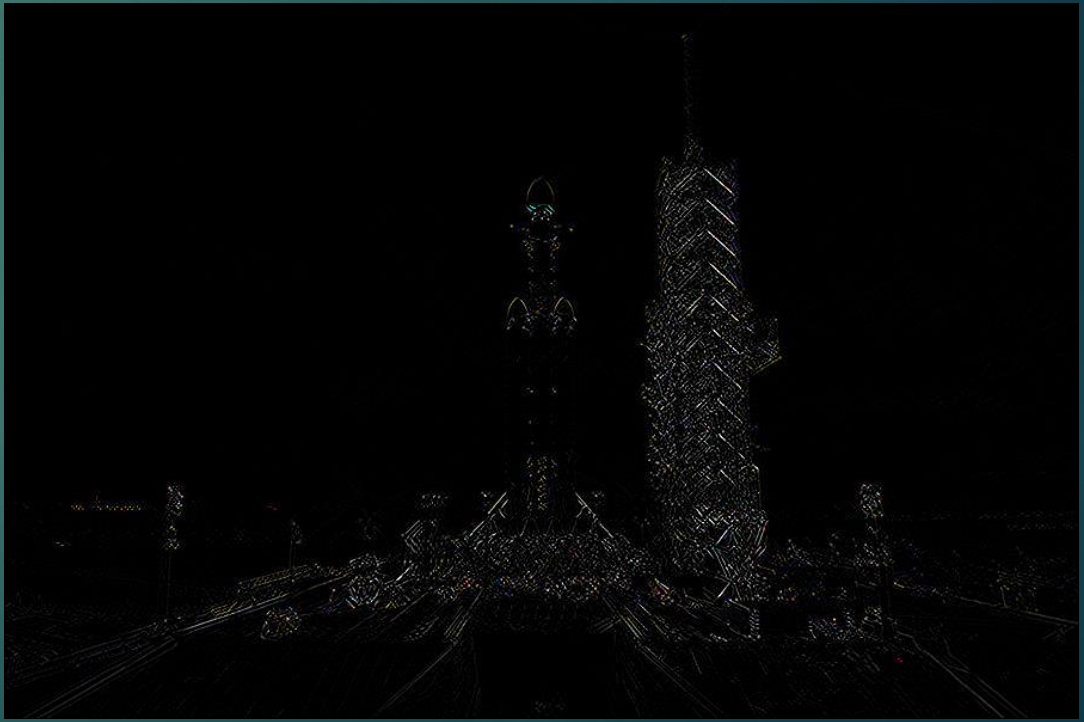
- Create an image difference of the images produced from the previous slides.
- Implementation:

```
image1 = img1*1.0
    image2 = img2*1.0
    dif = np.abs(image1-image2)
    dif -= dif.min()
    dif += dif.max()
    dif += 255
    return np.uint8(dif)
```

Edge Filter

- Develop and implement 3x3 diagonal edge detection kernel
- Kernel: [[1,0,-1],[0,0,0],[-1,0,1]]
- Implementation:

```
kernel = np.float32([[1,0,-1],[0,0,0],[-1,0,1]])
return cv2.filter2D(img, -1, kernel)
```



Sharpen Filter

- Develop and implement a 5x5 sharpen kernel.
- Kernel: [[-1,-1,-1,-1,-1],[-1,2,2,2,-1],[-1,2,8,2,-1],[-1,2,2,2,-1],[-1,-1,-1,-1,-1]]
- Implementation:

```
kernel = np.float32([[-1,-1,-1,-1,-1],[-1,2,2,2,-1],[-1,2,8,2,-1],[-1,2,2,2,-1],[-1,-1,-1,-1,-1]])
return cv2.filter2D(img, -1, kernel * (1/8))
```