Görüntü İşleme

21.10.24

→ Negatif {

→ log {gama}

→ identity. kuvvet {γ dönüşümü} gama dönüşümleri

↳

Yoğunluk - Düzeyi Dilimleme

(intensity-level slicing)

↳ Uydudan alınan görüntüleri düzenleme

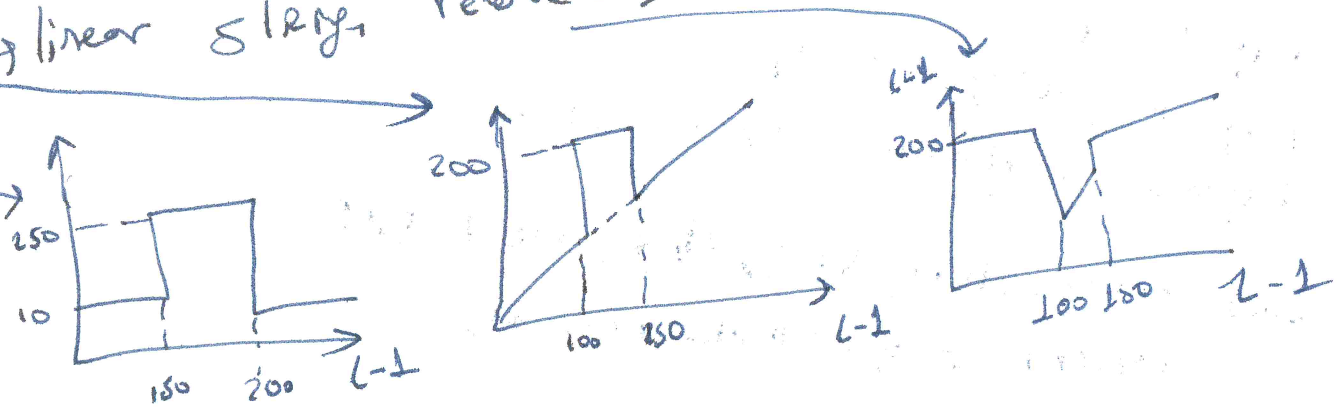
↳ X-ray fotoğraflarındaki kusurları görme

→ binary slicing ⇒ ışık yoğunluğuna sahip pikselleri düzelt.

↳ 255 ya da 0 sahip ol

→ linear slicing ⇒ bazı değerleri yükselt

→ linear slicing reverse ⇒ 150-250 arasında batsın.



kod tarafı →

↳ Visual Studio Code;

Open Folder
↳ HO8.
↳ images
L Main.py
L ~~py~~

α ⇒ terminal
↳ python -m venv .hobdy.

α ⇒ folder ⇒ Scripts
↳ Activate.ps1 aktif hale getir

α ⇒ pip install opencv-python.
α ⇒ pip install matplotlib

→ Main.py.

```
import py
import numpy.as np
import matplotlib.pyplot as plt.
def binary_slicing(img, A, B, alt, ust):
        img_out = np.fulllike(img, alt)
a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
b = np.full_like(a, 255)
# print(b) → hepsini 255 yapar.
seciler = a>4
# print(seciler) → [False, False, False, False, T, T, T, T])
c = a[seciler]
# print(c) → [5, 6, 7, 8, 9]
# a[seciler] = 255 ⇒ [1 2 3 4 255 255 255 255].
```

Secilen = np.logical_and (a>=4, a<=7)

print (secilen) → 4 ile 7 arasındaki degerler getirdi.

b = np.full_like (a, 0) değerler aldı.

b [secilen] = 255 ⟹ 0 ile 255 arasında

tekrar fonksiyon'a dönüş

↳ def binary_slicing (img, A, B, ~~alt~~ ~~üst, alt~~ alt, üst);

```
        img_out = np.full_like (img, alt)
        aralık = np.logical_and (img>A, img<B)
        img_out[aralık] = üst
        return img_out.
```

```
img_path = ''. imgs''
img = cv2.imread (img_path, 0)
img_bs = binary_slicing (img, 150, 250, 10, 255)

plt.imshow (img_bs, cmap="gray")
plt.show ()
```

siyah beyaz

bu ise arayda 10 yap değilse 255 yap.

```
def linear_slicing (img, A, B, üst):
        img_out = img.copy()
        aralık = np.logical_and(img>A, img<B)
        img_out[aralık] = üst
        return img_out
img_ls = linear_slicing (img, 150, 250, 255)
plt.imshow (img_ls, cmap="gray")
plt.show()
```

```
def linear_slicing_reverse( ~~X~~ img, A, B, cost):
        img_out = img.copy()
        ~~aralik_1 = np.logical_and( ~~ ~~a>=0, a<A~~)
    # aralik_1 = np.logical_and(img>=0, img< A)
    # aralik_2 = np.logical_and(img>B, img<=255)
    # aralik = np.logical_or(aralik1, aralik2)
        img_out[aralik] = cost
        return img_out
    aralik = np.logical_or(img<A, img>B)
```

```
~~img liner~~
img_lsr= ilinear_s_reverse(img, 150, 250, [illegible])
~~imread~~
 imshow(img_lsr, cmap
```

Tum resim yanyan koy

```
hstacked1= np. hstack((img, img_bs))
hstacked2 = np. hstack ((img_ls, img_lsr))
vstacked = np.vstack ((hstacked1, hstacked2))
 plt.imshow (vstacked, gray)
```

log_dn al