SPATIAL FILTERING

Image Filtering

Idea: Use the information coming from the neighboring pixels for processing

• computes a function of a local neighborhood at each pixel position

Called “Local operator,” “Neighborhood operator,” or “Window operator”

– Enhance images (geliştirme)

Noise reduction, smooth, resize, increase contrast, recolor, artistic effects, etc.

– Extract features from images (özellikleri ayıklama)

Texture(doku), edges, distinctive(ayırt edici) points, etc.

– Detect patterns

Template matching, ex: eye template

Filter = frequency domain processing

* Accept or reject certain frequency components

Signals

• A signal is composed of low and high frequency components

Low frequency components = signal is change slowly – smooth part

High frequency components = edge,corner

Low-frequencies
(coarse-scale details)

High-frequencies
(fine-scale details)

Types of noise

Salt and peper: black and white points

Impulse: white points

Gaussian: variations in intensity drawn from gaussian normal distribution

Brightness of each pixel is decreased or increased

Filtering

• Processing done on a function

– can be executed in continuous form (e.g. analog circuit)

– but can also be executed using sampled representation

Linear filtering

• Filtered value is the linear combination of neighboring pixel values.

– linearity: filter(f + g) = filter(f) + filter(g)

convolution

simple averaging with weighted averages(each sample get its own weight)

Tüm elemanların ortalamasını alıyor, bir elemana eşitliyor

Assumptions:

*yan yana olan pixellerin çok farklı olmayacağı varsayılıyor. (smooth-not black white)

*noise process her pixelde farklı işliyor, independent from neighbours

Convolution pesudocode:

```
s = 0
for j = -r to r
        s = s + a[j]b[i-j]
return s
```

• Box filter – Simple and cheap

• Tent filter – Linear interpolation - weight linear azalıp artıyor

• Gaussian filter – Very smooth antialiasing filter

Discrete filtering in 2D

```
Func (filter2d a, filter2d b, int i, int j)
s = 0
r = a.radius
for i' = -r to r do
        s = s + a[i'][j']b[i – i'][j – j']
return s
```

convolution = i-u (a*b) i+rden i-rye gidiyor

correlation = i+u (soldan sağa yukardan aşağı)

correlation filtering

**template matching – bulduğunu parlak yapıyor

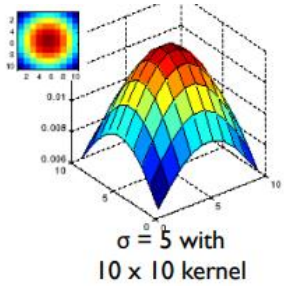
Smoothing by averaging

Bigger window(filter) size -> blur image(daha fazla komşunun ortalaması alınıyor)

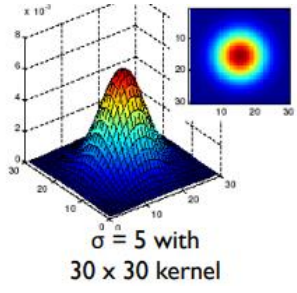
Low-pass filters = smoothlaştırır düşük frekans

Gaussian filter Resmin her yerine ulaşır – infinite support

Seperability example



σ = 5 with
10 x 10 kernel



σ = 5 with
30 x 30 kernel