Дигитална обрада звука и слике 2

Саша Малков
Математички факултет
2023/2024

Дигитална обрада звука и слике

Тема 1
Дигитална обрада слике
(наставак)

Локалне трансформације

- Слика представља матрицу тачака
- Појединачне тачке се мењају независно од осталих тачака
- Сваки канал се мења за себе

Слика је матрица пиксела

- Пикселе слике добијамо са:
  ```
  pixels = img.load()
  ```
- Пикселима приступамо као елементима матрице
- Сваки пиксел је тројка `(r,g,b)`
  ```
  r,g,b = pixels[x,y]
  pixels[x,y] = (r,g,b)
  ```

Слика је матрица пиксела (2)

- Све пикселе слике обрађујемо помоћу две петље:

```
for i in range( img.width ):
   for j in range( img.height ):
      ... pixels[i,j] ...
```


Локалне трансформације (2)

- Алтернатива експлицитном коришћењу матрице пиксела је употреба метода:

```
img.point( lambda x: f(x) )
```

  - за сваку тачку посебно...
  - ...и за сваки канал посебно...
  - ...постојећа вредност x се замењује са $f(x)$
  - ако је $f(x)$ мање од 0, уписује се 0, ...
  - ... а ако је веће од 255 уписује се 255

Локалне трансформације (2)

- Функција `img.point(f)` ради као:

```
img1 = img.copy()
pixels = img1.load()
for i in range(img1.width):
   for j in range(img1.height):
      r, g, b = pixels[i,j]
      r = f(r)
      g = f(g)
      b = f(b)
      pixels[i,j] = (r,g,b)
return img1
```

- Уз описано понашање за вредности ван опсега [0,255]

Праг

- Операција "праг" (енгл. *threshold*)
- Локална трансформација пиксела
- Све вредности изнад прага се замењују максималном вредношћу (255), а све вредности испод прага минималном (0)

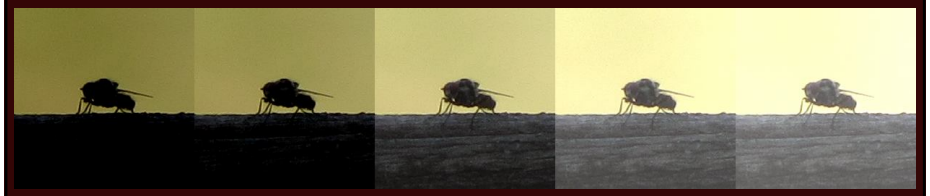
Праг (2)

```
tc = img.point( lambda x: 255 if x>100 else 0)
gs = img.convert("L")
tgs = gs.point( lambda x: 255 if x>100 else 0)
tile( img, tc, gs, tgs ).show()
```


Осветљење

- Операција "осветљење" (енгл. *brightness*)
- Локална трансформација пиксела
- Све вредности се повећавају за дату вредност

Осветљење (2)

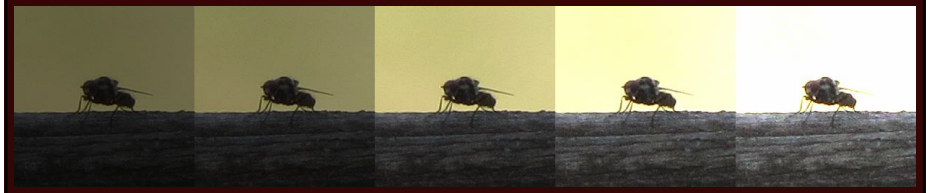


```
tile(
    img.point( lambda x: x - 75),
    img.point( lambda x: x - 50),
    img,
    img.point( lambda x: x + 50),
    img.point( lambda x: x + 75)
).show()
```


Контраст

- Операција "контраст" (енгл. *constrast*)
- Локална трансформација пиксела
- Све вредности се множе датим коефицијентом

Контраст (2)



```
tile(
    img.point( lambda x: x * 0.5 ),
    img.point( lambda x: x * 0.66 ),
    img,
    img.point( lambda x: x * 1.5 ),
    img.point( lambda x: x * 2 )
).show()
```


Линеарна промена динамичког опсега

- Комбинација контраста и осветљења
- Локална трансформација пиксела
- Све вредности се множе датим коефицијентом и повећавају за дати број

```
def contrast_brightness( img, c, b ):
    return img.point( lambda x: x * c + b )

def img_range( img, min, max ):
    return contrast_brightness( img,
           255.0/(max-min),
           -min * 255.0/(max-min)
    )
```

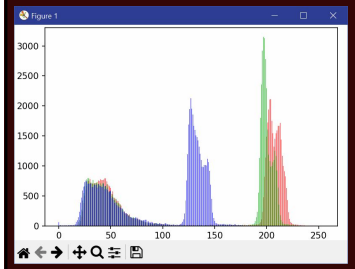

Линеарна промена динамичког опсега (2)

```
tile(
    img,
    contrast_brightness( img, 1.4545, -23 ),
    img_range( img, 16, 192 ),
    contrast_brightness( img, 2.37, -76 ),
    img_range( img, 32, 140 )
).show()
```


Хистограм

- У претходним примерима је потребно да се установе границе динамичког распона
  - преброји се колико којих вредности има
  - img.histogram()
    - рачуна за сваки канал колико којих вредности има
    - за *RGB* враћа низ од 256x3 вредности
      - првих 256 вредности је број појављивања вредности 0-255 у црвеном каналу
      - других 256 вредности је број појављивања вредности 0-255 у зеленом каналу
      - трећих 256 вредности је број појављивања вредности 0-255 у плавом каналу

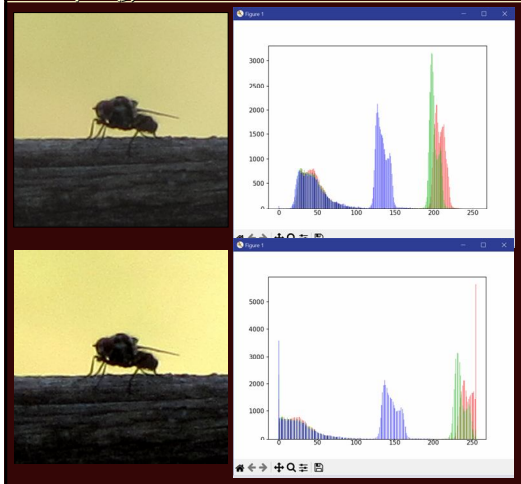
Хистограм (2)



- Приметимо три региона:
  - 15-100, тамна област, сви канали
  - 120-150, средња област, плави канал
  - 190-225, светла област, црвена и зелена

```
h = img.histogram()
plot.bar( range(0,256), h[0:256], color="#ff0000", alpha=0.5 )
plot.bar( range(0,256), h[256:512], color="#00aa00", alpha=0.5 )
plot.bar( range(0,256), h[512:768], color="#0000ff", alpha=0.5 )
plot.show()
```


Хистограм (3)



- Слика и хистограм пре примене контраста

- Слика и хистограм после примене контраста:

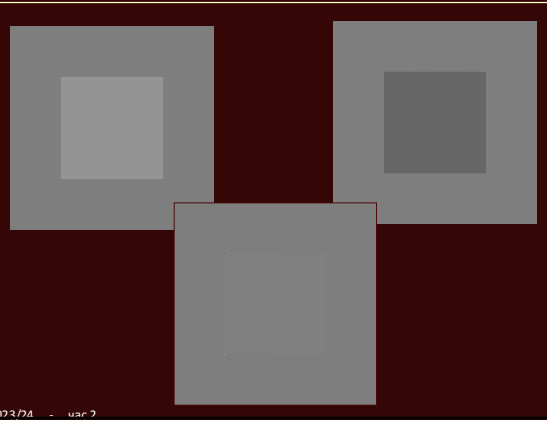
```
img = img_range(img, 25, 215)
```


Хистограм (2)

```
tile(
    img,
    img_range( img, 15, 100 ),
    img_range( img, 120, 150 ),
    img_range( img, 190, 225 ),
    img_range( img, 15, 225 )
).show()
```


Гама

- У идеалном случају, ако је 0 црна, а 255 бела, онда би 127,5 требало да буде сива која има тачно средњи интензитет
- У пракси то није тако, како због природе извора светла, тако и због осетљивости ока



Гама (2)

- Средина обојена различитим интензитетима
- Руб је комбинација истог броја белих и црних тачака

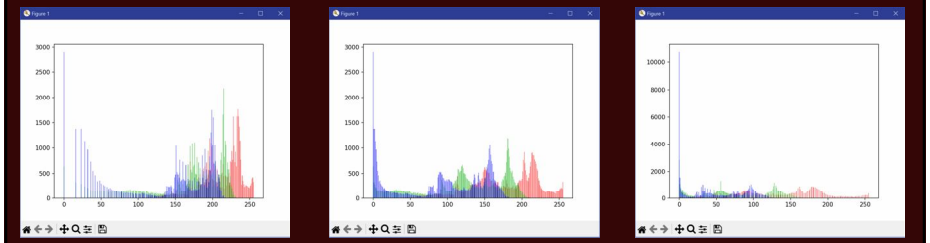
Гама-корекција

- Примењује се функција: x^{α}
  - претпоставља се да је распон 0-1
  - функција пресликава 0 у 0 и 1 у 1, а средина интервала се помера на неку страну, у зависности да ли је α веће или мање од 1
- α>1
  - повећава се контраст светлих делова слике
  - слика се генерално потамњује
- α<1
  - повећава се контраст тамних делова слике
  - слика се генерално посветљује

Гама-корекција (2)

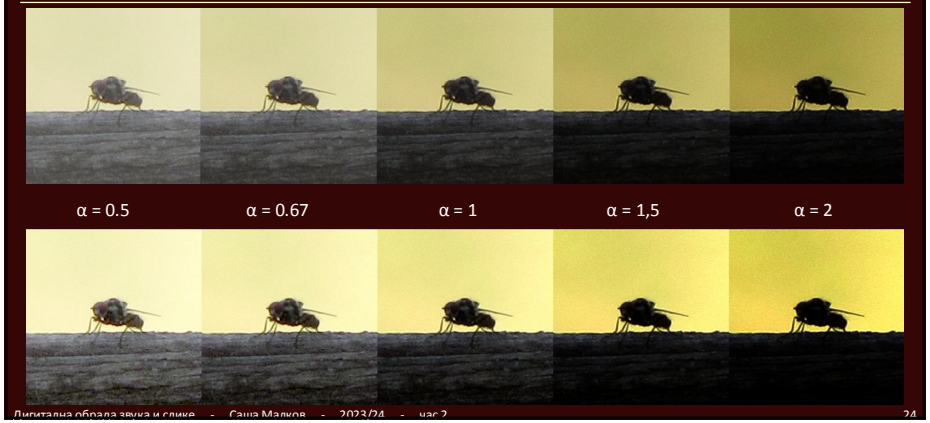
α = 0.33 α = 0.5 α = 1 α = 2 α = 3

Гама-корекција (3)



α = 0.5 α = 1 α = 2

Гама-корекција (2)



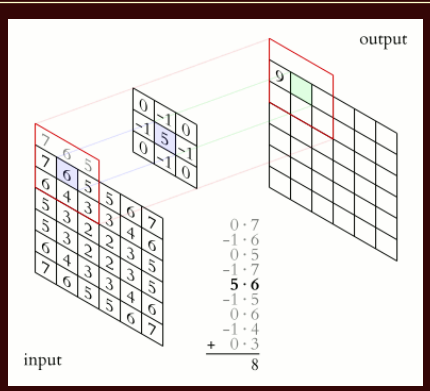
α = 0.5 α = 0.67 α = 1 α = 1,5 α = 2

Филтери

- Филтер рачуна вредност тачке на основу вредности тачака у окружењу
- Разликују се по начину рачунања

Конволутивни филтери

- Врста филтера
  - често се назива "језгро" (енгл. *kernel*)
- Поступак:
  - околне тачке се множе вредностима матрице
  - добијене вредности се сабирају
  - на крају се додаје *bias*



Слика, *Wikipedia*

Конволутивни филтери (2)

- *ImageFilter.BoxBlur*
  - замућење слике, смањује разлику између тачке и суседних тачака
  - сви коефицијенти квадратне матрице су једнаки, а сума им је 1
- *ImageFilter.GausianBlur*
  - замућење слике, смањује разлику између тачке и суседних тачака
  - коефицијенти матрице одговарају густини нормалне расподеле у 2Д, тако да им је сума 1
- *ImageFilter.SHARPEN*
  - изоштравање слике, повећава разлику између тачке и суседних тачака
  - централни коефицијент је позитиван а остали су негативни, сума је 1

Слика, *Wikipedia*

Конволутивни филтери – *BoxBlur*

```
tile(
    img,
    img.filter( ImageFilter.BoxBlur(1) ),
    img.filter( ImageFilter.BoxBlur(3) ),
    img.filter( ImageFilter.BoxBlur(5) )
).show()
```


Конволутивни филтери – *GaussianBlur*

```
tile(
    img,
    img.filter( ImageFilter.GaussianBlur(1) ),
    img.filter( ImageFilter.GaussianBlur(3) ),
    img.filter( ImageFilter.GaussianBlur(5) )
).show()
```


Конволутивни филтери – *Sharpen*

```
tile(
    img,
    img.filter( ImageFilter.DETAIL ),
    img.filter( ImageFilter.SHARPEN )
).show()
```


Конволутивни филтери – *Замућење*

```
tile( img,
      img.filter( ImageFilter.Kernel( (3,3), [1, 2, 1,
                                              2, 12, 2,
                                              1, 2, 1] )),
      img.filter( ImageFilter.Kernel( (3,3), [1, 2, 1,
                                              2, 8, 2,
                                              1, 2, 1] )),
      img.filter( ImageFilter.Kernel( (3,3), [1, 2, 1,
                                              2, 4, 2,
                                              1, 2, 1] ))
).show()
```

Конволутивни филтери – *Изоштравање*

```
tile( img,
      img.filter( ImageFilter.Kernel( (3,3), [-1, -2, -1,
                                              -2, 30, -2,
                                              -1, -2, -1] )),
      img.filter( ImageFilter.Kernel( (3,3), [-1, -2, -1,
                                              -2, 20, -2,
                                              -1, -2, -1] )),
      img.filter( ImageFilter.Kernel( (3,3), [-1, -2, -1,
                                              -2, 15, -2,
                                              -1, -2, -1] ))
).show()
```


Минимум, Максимум, Медијана

- Тачка добија вредност најмање/највеће/средишње вредности из скупа околних тачака
- Минимум и максимум се користе за проширивање тамних или светлих елемената
- Медијана се користи за елиминисање шума

Минимум, Максимум

```
tile(
    img,
    img.filter( ImageFilter.MinFilter(3) ),
    img.filter( ImageFilter.MinFilter(5) ),
    img.filter( ImageFilter.MaxFilter(3) ),
    img.filter( ImageFilter.MaxFilter(5) )
).show()
```


Медијана

```
tile(
    img,
    img.filter( ImageFilter.MedianFilter(3) ),
    img.filter( ImageFilter.MedianFilter(5) )
).show()
```

Литература за тему

- **Pinta**
  https://www.pinta-project.com/
- **Paint.NET**
  https://www.getpaint.net/index.html
- **Pillow Documentation**
  https://pillow.readthedocs.io/en/stable/index.html
- Stephen Gruppetta, **Image Processing With the Python Pillow Library,**
  https://realpython.com/image-processing-with-the-python-pillow-library/