Дигитална обрада звука и слике

1

Саша Малков

Математички факултет

2023/2024

План курса

- Рад са дигиталним ресурсима
  - слика
  - звук
  - видео
  - комбиновање
- Коришћењем алата
- Коришћењем програмског језика *Python*

Дигитална обрада звука и слике

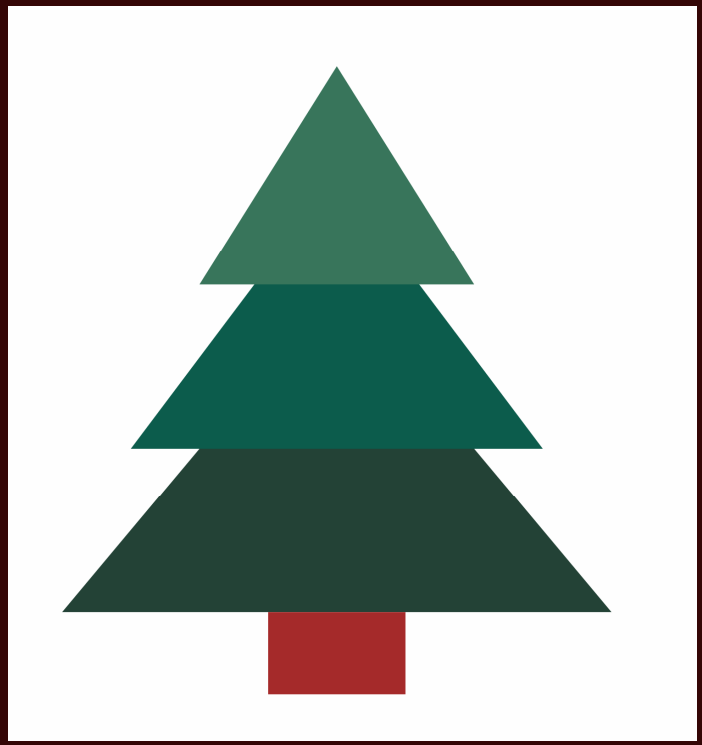
Тема 1

Дигитална обрада слике

Дигитална обрада слике

- Слике се чувају и обрађују у два основна облика:
  - растерске слике
  - векторске слике
- Обрада слике може да буде мануелна и аутоматизована
  - предмет и циљ обраде се разликује за растерске и векторске слике

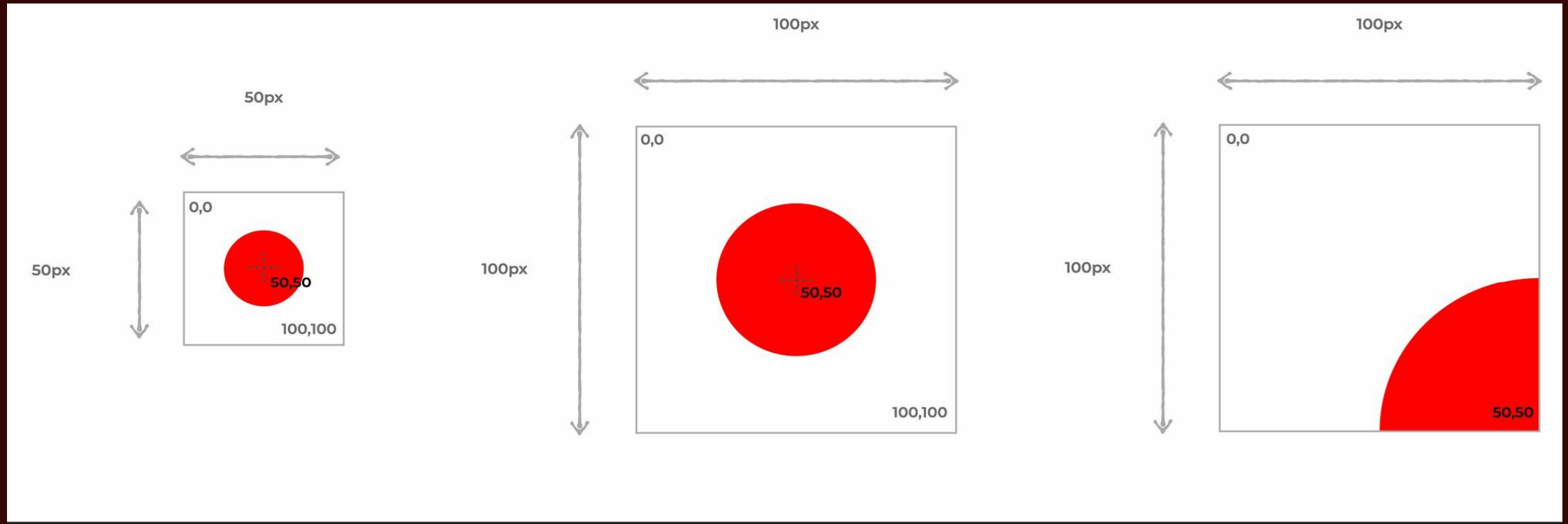
Векторска слика



- Слика се представља скупом елемената

```
...
<polygon points="0,0 80,120 -80,120"
   fill="#234236" />
<polygon points="0,-40 60,60 -60,60"
   fill="#0C5C4C" />
<polygon points="0,-80 40,0 -40,0"
   fill="#38755B" />
<rect x="-20" y="120" width="40" height="30"
   fill="brown" />
...
```

Векторска слика




```svg
<svg
  width="50" height="50"
  viewBox="0 0 100 100"
>
  <circle
    cx="50" cy="50"
    r="25" fill="red"
  />
</svg>
```

```svg
<svg
  width="100" height="100"
  viewBox="0 0 100 100"
>
  <circle
    cx="50" cy="50"
    r="25" fill="red"
  />
</svg>
```

```svg
<svg
  width="100" height="100"
  viewBox="0 0 50 50"
>
  <circle
    cx="50" cy="50"
    r="25" fill="red"
  />
</svg>
```

Обрада векторских слика

- Односи се на мењање садржаја слике
  - захтева разумевање структуре конкретне слике
  - мењају се облици, и карактеристике елемената
  - мењају се односи елемената
- Обично се ради мануелно, програмима за цртање

Обрада векторских слика (2)

- Може да се аутоматизује
  - додавањем програмских елемената програмима за цртање
  - независно од алата, на пример обрађивањем докумената у формату SVG

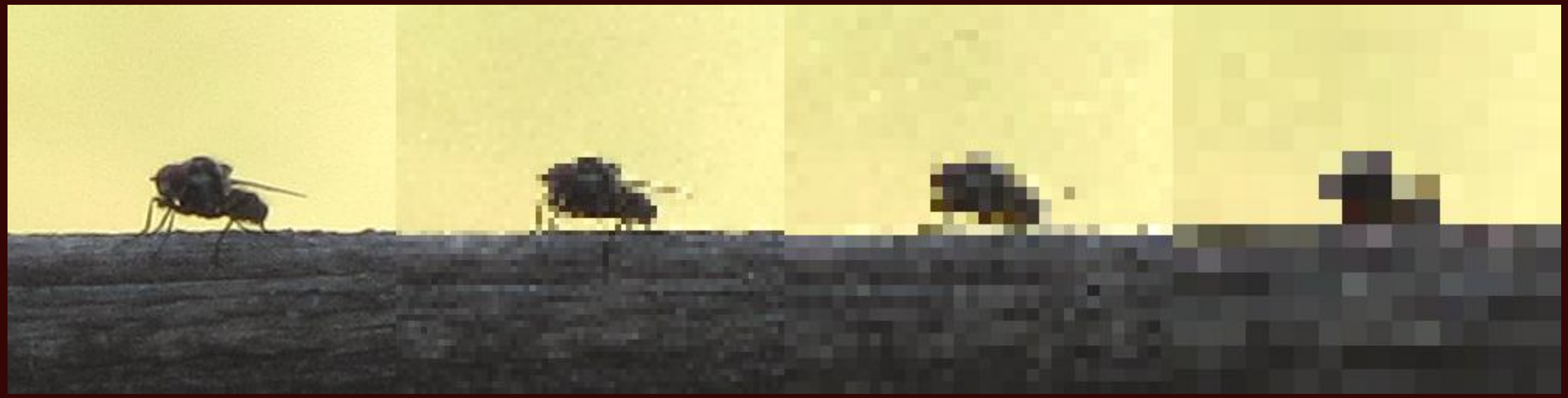
- Ради се углавном при припреми дигиталних садржаја
- Један од предмета обраде је конверзија у растерске слике

Алати за цртање векторских слика

- Има много алата, међу најпознатијим су:
  - *Adobe Illustrator*
  - *Corel Draw*
  - *Inkscape*

- Многи алати за растерско цртање имају неке елементе векторског цртања

Растерска слика



- Слика се представља као матрица тачака
  - за сваки "канал" по матрица
  - за црно белу слику – један канал
  - за слику у боји више канала – 3 за *RGB*, 4 за *CMYK*
  - додатни "алфа" канал за транспарентност

Обрада растерских слика

- Односи се на мењање визуалних карактеристика слике
  - захтева разумевање записа и формата записивања слике
- Може да се односи и на мењање садржаја и структуре
  - за разлику од векторских слика, елементи растерских слика обично нису означени
    - процес препознавања елемената слике је сложенији

Обрада растерских слика (2)

- Ради се мануелно, програмима за цртање и програмима за обраду слика
- Ради се аутоматски, применом одговарајућих библиотека
- Ради се и при припреми и при накнадној обради дигиталних садржаја
- Аутоматизује се много чешће од обраде векторских слика

- У даљем раду ћемо се углавном фокусирати на обраду растерских слика

Алати за цртање растерских слика

- Примарно су намењени за цртање
  - али могу да се користе и за обраду слика
- Има много алата:
  - *Adobe Fresco*
  - *Corel Painter*
  - *GIMP*
  - *Krita*
  - ...
- Многи алати за растерско цртање имају неке елементе векторског цртања

Алати за обраду растерских слика

- Примарно су намењени за обраду
  - али могу да се користе и за цртање

- Има много алата:
  - *Adobe Photoshop*
  - *Corel PaintShop*
  - *Paint.Net*
  - *Pinta*
  - ...

Фокус

- Разумевање техника обраде дигиталних слика
  - "Креда и табла"
- Примена техника помоћу алата за обраду слика
  - Paint.Net
  - Pinta
- Примена техника писањем програма на Пајтону
  - *Pillow (PIL)*
  - *OpenCV*

Paint.Net, Pinta

- ...

Библиотека *Pillow*

- Новија верзија старије библиотеке *PIL*
- Основне операције са сликама
- Умерено напредне операције са сликама
- Једноставна и лака за употребу

  - https://pillow.readthedocs.io/en/stable/
  - `py -m pip install pillow`

Библиотека *Pillow* – Читање и писање

```
import PIL
from PIL import Image
...
img = Image.open( filename )
img.load()
img.show()
img.close()
img.save( filename )
...
with Image.open( filename ) as img:
    ...
```

Трансформације слике

- Геометријске трансформације
- Локалне трансформације
  - Промене боја и динамичких карактеристика
- Филтри
- Комбиновање слика

Геометријске трансформације

- Промена величине
- Исецање дела слике
- Огледало
- Ротирање
- Конверзија формата и издвајање канала

Промена величине

- Непропорционална промена величине
- Слика се трансформише независно по ширини и висини
    - img.resize((w,h))
    - img.resize((w,h), resample)

Промена величине (2)

```
img1 = img.resize( (16,16), resample=Image.NEAREST )
tile(
    img,
    img1.resize( (250,250), resample=Image.NEAREST ),
    img1.resize( (250,250), resample=Image.BILINEAR ),
    img1.resize( (250,250), resample=Image.BICUBIC )
).show()
```

Промена величине (3)

- Пропорционална промена величине
- Слика се повећава или смањује уз задржавање односа димензија
  - img.reduce(n)
  - PIL.ImageOps.scale(img, factor)
  - PIL.ImageOps.scale(img, factor, resample)

Промена величине (4)

- Пропорционална промена величине са проширењем
  - PIL.ImageOps.pad
  - PIL.ImageOps.expand

Промена величине (5)

```
ImageOps.pad(
    ImageOps.pad(
        img,
        (450,250),
        color="#3333ff"
    ),
    (550,450),
    color="#33ff33"
)
```

Исецање дела слике

- Исеца се правоугаони део слике
  - img.crop(x0, y0, x1, y1)
  - PIL.ImageOps.crop(img, border)

Исецање дела слике (2)

```
tile(
    img,
    img.crop((0,0,150,150)),
    img.crop((50,50,150,150,)),
    img.crop((50,50,200,200)),
    img.crop((100,100,250,250))
).show()
```

Огледало

- Слика се трансформише у симетричну по хоризонталној или вертикалној оси
  - img.transpose(type)
  - PIL.ImageOps.mirror
  - PIL.ImageOps.flip

Огледало (2)

```
tile(
    img,
    img.transpose( Image.FLIP_LEFT_RIGHT ),
    img.transpose( Image.FLIP_TOP_BOTTOM ),
    img.transpose( Image.TRANSPOSE ),
    img.transpose( Image.ROTATE_180 )
).show()
```

Ротирање

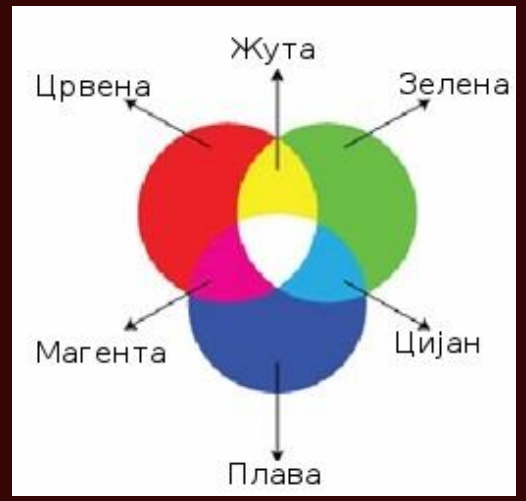
- Слика се ротира за дати угао
- img.rotate(angle, resample, expand)

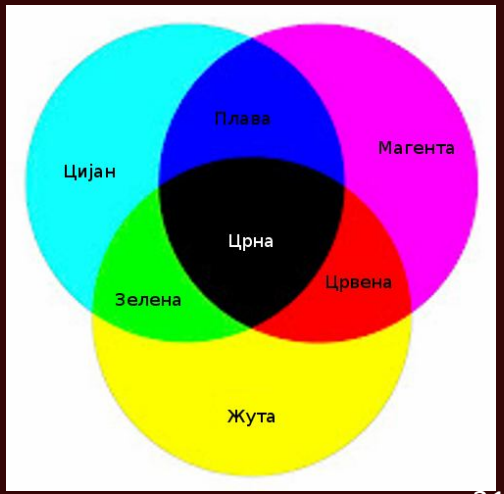
Огледало (2)

```
tile(
    img.rotate( 45, expand=True),
    img.rotate( 56, expand=True),
    img.rotate( 67, expand=True),
    img.rotate( 79, expand=True),
    img.rotate( 90, expand=True)
).show()
```

Канали

- Црно-бела (енгл. *grayscale*) слика има један канал
- Слика у боји има
  - 3 канала – *RGB*
    - адитивни модел боја
    - користи се за екран рачунара
  - Слика у боји има 4 канала – *CMYK*
    - суптрактивни модел боја
    - користи се за штампање

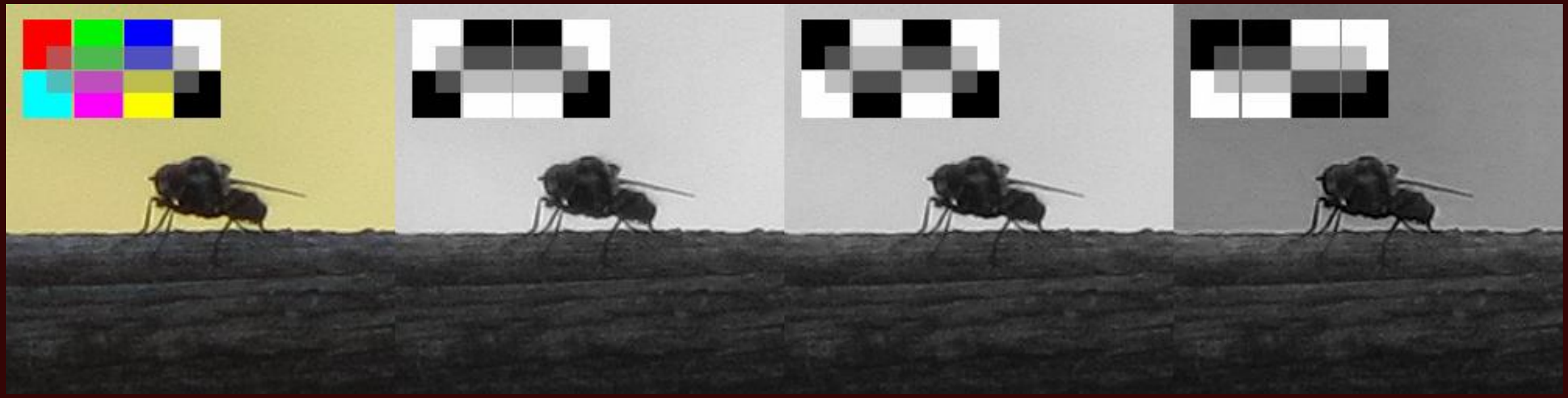




Канали (2)

- Растерска слика се састоји од матрица тачака
  - за сваки канал по матрица интензитета
  - или једна матрица чији су елементи вредности свих канала
- Слика може се конвертује у други формат:
  - img.convert(...)
- Или да се споји од више канала:
  - Image.merge(format, (...))

Канали (3)

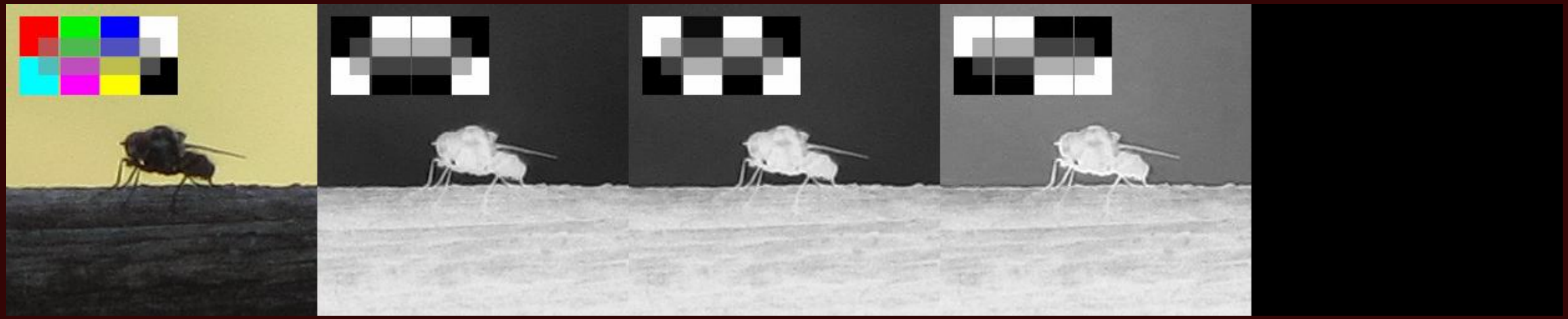


```
r, g, b = img.split()
tile( img, r, g, b ).show()
```

Канали (4)

```
tile(
    img,
    Image.merge( "RGB", (r, black, black)),
    Image.merge( "RGB", (black, g, black)),
    Image.merge( "RGB", (black, black, b)),
    Image.merge( "RGB", (r,g,b) )
).show()
```

Канали (5)

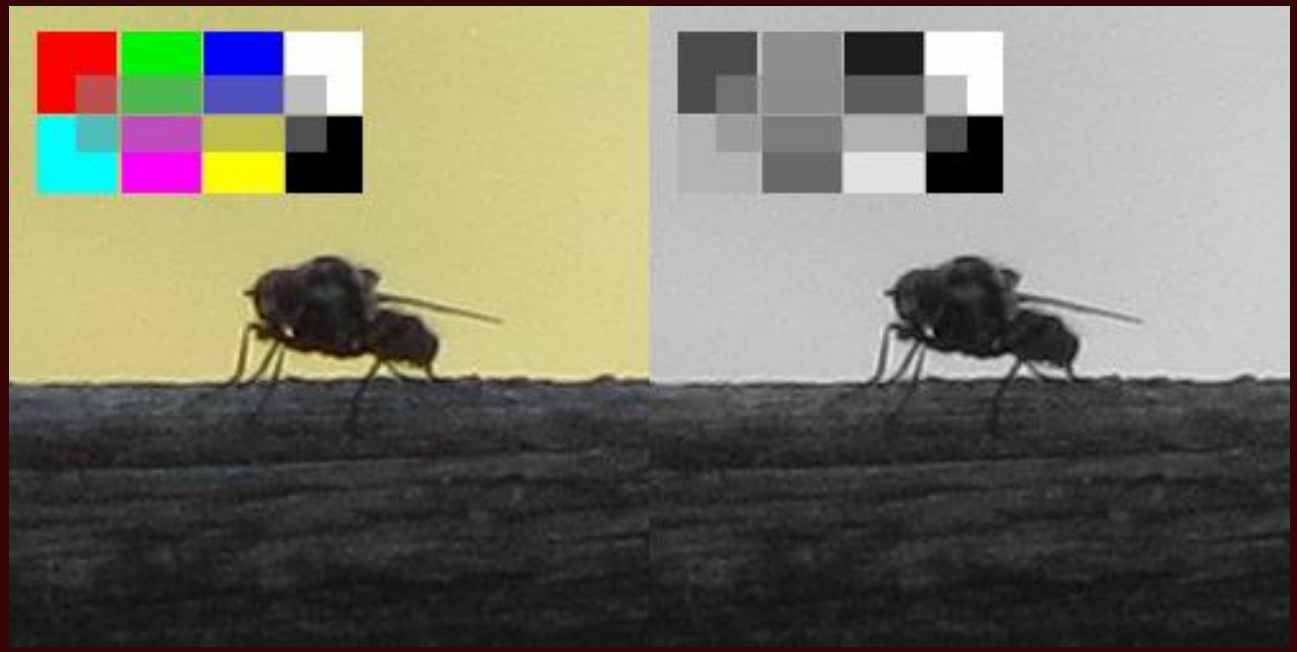


```
c, m, y, k = img.convert("CMYK").split()
tile( img, c, m, y, k ).show()
```

Канали (6)

```
tile(
    img,
    Image.merge( "CMYK", (c, black, black, black)),
    Image.merge( "CMYK", (black, m, black, black)),
    Image.merge( "CMYK", (black, black, y, black)),
    Image.merge( "CMYK", (black, black, black, k)),
    Image.merge( "CMYK", (c,m,y,k) )
).show()
```

Канали (7)



```
l = img.convert("L")
tile( img, l ).show()
```

Локалне трансформације

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

- img.point(lambda x: f(x))
  - вредност сваког *x* канала сваке тачке 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
```

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()
```

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/