

## Vaja 11: Vektor premika

Pripravila: Gašper Podobnik & Tomaž Vrtovec

### Navodila

Algoritem *bločnega ujemanja* predstavlja enega najosnovnejših pristopov za določanje vektorjev premika na področju obdelave videa.

1. Dan je testni video `simple-video.avi`, ki je sestavljen iz  $K=153$  zaporednih sivinskih slik velikosti  $X \times Y = 160 \times 100$  slikovnih elementov in zapisan v nezgoščeni obliki s frekvenco 25 Hz znotraj zalogovnika AVI.

Napišite funkcijo, ki naloži točno določeno sliko iz videa:

```
def loadFrame(iVideo, iK):  
    """  
    Funkcija za nalaganje slike iz videa  
    """  
  
    return oFrame
```

kjer vhodni argument `iVideo` predstavlja video posnetek, ki ga predhodno naložite s funkcijo `VideoCapture` iz knjižnice `cv2` (<https://pypi.org/project/opencv-python/>), `iK` pa zaporedno številko slike, ki jo želimo naložiti.

Naložite dani video, iz njega izluščite slike pri  $k=30$  in  $k=31$  ter slike prikažite. Pomagajte si z metodama `.set()` in `.read()` objekta `cv2.VideoCapture`.

2. Napišite funkcijo za določanje polja vektorjev premika z algoritmom bločnega ujemanja:

```
def blockMatching(iF1, iF2, iSize, iSearchSize):  
    """  
    Funkcija za določanje polja vektorjev premika z bločnim  
    ujemanjem  
    """  
  
    return oMF, oCP
```

kjer vhodni argument `iF1` predstavlja sliko videa pri času  $t_1$ , `iF2` sliko videa pri času  $t_2 > t_1$ , `iSize` =  $[B_x, B_y]$  vektor dimenzije  $B_x \times B_y$  (v slikovnih elementih) posameznega bloka slike, `iSearchSize` =  $2P + 1$  pa velikost (v slikovnih elementih) kvadratnega področja iskanja vektorjev premika. Izhodni argument `oMF` predstavlja polje vektorjev premika, `oCP` pa koordinate središča posameznega bloka slike (v slikovnih elementih). Oba izhodna argumenta sta tridimenzionalni matriki, kjer število vrstic predstavlja število blokov vzdolž vrstic slike, število stolpcev število blokov vzdolž stolpcev slike, število plasti pa  $x$  in  $y$  komponenti vektorjev premika oz. koordinati središč blokov.

Bločno ujemanje temelji na razdelitvi slike na  $M$  blokov  $\mathcal{B}_m$ ;  $m = 1, 2, \dots, M$ , za katere je potrebno določiti pripadajoče vektorje premika  $\mathbf{d}_m = [d_{m,x}, d_{m,y}]^T$  med sliko  $I_k(\mathbf{x})$  ob času  $k$  in sliko  $I_{k-1}(\mathbf{x})$  ob času  $k-1$  tako, da je napaka napovedovanja  $\varepsilon(\mathbf{d}_m)$  čim manjša:

$$\mathbf{d}_m^* = \arg \min_{\mathbf{d}_m \in \mathcal{P}} \varepsilon(\mathbf{d}_m); \quad \varepsilon(\mathbf{d}_m) = \frac{1}{|\mathcal{B}_m|} \sum_{\mathbf{x} \in \mathcal{B}_m} |I_k(\mathbf{x}) - I_{k-1}(\mathbf{x} - \mathbf{d}_m)|,$$

kjer  $\mathbf{x} = [x, y]^T$  predstavlja vektor prostorskih koordinat slikovnega elementa slike,  $\tilde{I}_k(\mathbf{x}) = I_{k-1}(\mathbf{x} - \mathbf{d}_m)$  napoved slike  $I_k(\mathbf{x})$  s kompenzacijo premika,  $\mathbf{d}_m^*$  pa pripadajoči optimalni vektor premika za blok  $\mathcal{B}_m$ , ki sestoji iz  $|\mathcal{B}_m|$  slikovnih elementov. Kandidate za optimalni vektor premika  $\mathbf{d}_m^*$  se izbira na področju iskanja  $\mathcal{P}$  ob predpostavkah, da je velikost  $\mathcal{P}$  enaka  $(2P + 1) \times (2P + 1)$  ter da velja  $P = 2^a - 1$ ;  $a \in \mathbb{N}$ , in sicer se  $\mathbf{d}_m^*$  določi s t.i. logaritemskim iskanjem v treh korakih:

$$P_i = \frac{P+1}{2^i}; \quad \mathcal{P}_i = \left\{ \mathbf{x} : (0, 0); (0, \pm P_i); (\pm P_i, 0); (\pm P_i, \pm P_i) \right\}; \quad i = 1, 2, 3.$$

Določite polje vektorjev premika za izluščeni sliki videa pri velikosti bloka  $8 \times 8$  slikovnih elementov in velikosti področja iskanja vektorjev premika 15 slikovnih elementov.

3. Napišite funkcijo za prikaz polja vektorjev premika:

```
def displayMotionField(iMF, iCP, iTitle, iImage=None):
    """
    Funkcija za prikaz polja vektorjev premika
    """

    return fig
```

kjer vhodni argument `iMF` predstavlja polje vektorjev premika, `iCP` koordinate središča posameznega bloka (v slikovnih elementih), `iTitle` naslov prikaznega okna, `iImage` pa je parameter, ki predstavlja sliko iz videa. Če uporabnik ne poda vrednosti za ta argument, se izriše samo polje vektorjev, v nasprotnem primeru pa se polje vektorjev izriše superponirano na podano sliko iz videa. Za prikaz vektorjev si pomagajte s funkcijo `quiver()` iz knjižnice `matplotlib`. Izhodni argument `fig` naj predstavlja objekt prikaznega okna (`fig = plt.figure()`).

Prikažite polje vektorjev premika, ki ste jih pridobili pot točko 2, ter jih superponirajte na pripadajočo sliko videa.

## Vprašanja

Odgovore na sledeča vprašanja zapišite v poročilo, v katerega vstavite zahtevane izrise in programske kodo.

1. Izračunajte napoved slike pri času  $t_2 > t_1$  na podlagi slike pri času  $t_1$  ter pripadajočo sliko razlik. Priložite programsko kodo in izrise obeh slik.
2. Sestavite video polja vektorjev premika za vsak par zaporednih slik danega videa in ga shranite v datoteko tipa gif, tako da bo frekvenca novega videa enaka frekvenci originalnega video posnetka. Za pridobitev slik videa si pomagajte s Pythonovo funkcijo `fig2img()`:

```
from PIL import Image
import io
def fig2img(fig):
    """Convert a Matplotlib figure to a PIL Image and return it"""
    buf = io.BytesIO()
    fig.savefig(buf)
    buf.seek(0)
    img = Image.open(buf)
    return img
```

ki celotno sliko prikaznega okna `fig` pretvori v spremenljivko tipa `PIL.Image`. Zapis videa v gif naredite tako, da si posamezne slike tipa `PIL.Image` shranite v seznam (npr. z imenom `frames`), nato pa z metodo `.save()` shranite gif na želeno lokacijo. Pomagate si lahko s spodnjim ukazom in dokumentacijo na spletni strani: <https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html#saving>.

```
frames[0].save(
    "<ime-posnetka>.gif",
    duration= # potrebno nastaviti,
    loop=0,
    save_all=True,
    optimize=False,
    append_images=frames[1:],
)
```

Priložite programsko kodo, v priponki pa tudi dobljeni video.

3. Preizkusite delovanje algoritma bločnega ujemanja na realnem videu `real-video.avi`, ki je sestavljen iz  $K = 138$  zaporednih sivinskih slik velikosti  $X \times Y = 256 \times 144$  slikovnih elementov in zapisan v nezgoščeni obliki s frekvenco 25 Hz znotraj zalgovnika AVI. Priložite programsko kodo in izrise slik (polje vektorjev premika ter polje vektorjev premika, superponirano na sliko) za poljubno izbrani primer slik iz tega videa.