Haladóprogramozás beadandó

Projekt: Képmódosító script Python nyelven

Készítette: Nagy Kristóf – S1LGWD

GITHUB: https://github.com/SecretPadawan2/Halado-beadando

Projekt célja:

A program célja, hogy a felhasználó egyszerre több képet tudjon automatikusan feldolgozni előre megírt képmódosító algoritmusokkal.
A megoldás egy parancssorból futtatható Python-script, amely alkalmas tömeges (batch) képfeldolgozásra.
Ez hasznos lehet például webre feltöltés előtti optimalizálásnál, képgaléria előkészítésnél, vagy egy fényképsorozat átméretezésénél.
Egy AI-funkció amely egy igazolványkép szerkesztő arcdetektálás funkcióval és alapvető szerkesztési utasítások.

Rendszerkövetelmények

A program bármilyen gépen fut, amin:

- **Python 3.8+**
- **PIP csomagkezelő**
- Windows / Linux / macOS

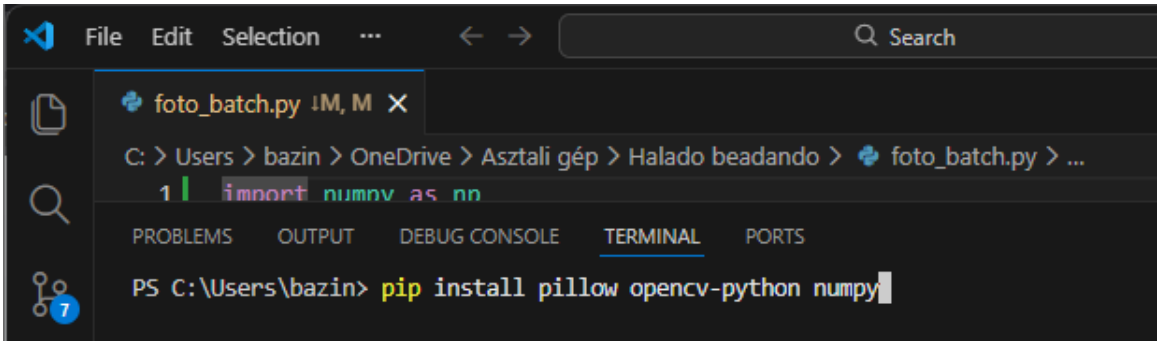
Ajánlott minimális gép:

- CPU: Intel i3 / AMD Ryzen 3 vagy jobb
- RAM: legalább 4 GB
- HDD/SSD: pár száz MB hely a feldolgozott képek miatt
- OpenCV használatához legalább SSE2 támogatás (minden modern CPU-ban van)

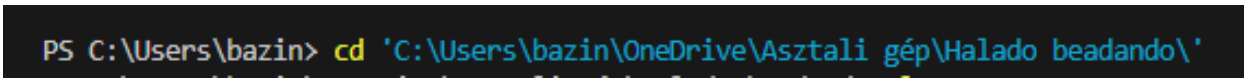
Szükséges telepítés

A következő Python csomagokat kell a Visual Studio Code terminálon keresztül telepíteni:

pip install pillow opencv-python numpy



Nyissuk meg a mappát ahova mentettük a fájlunkat használjuk a cd és ls parancsokat a könnyebb navigációért.(Használjuk a TAB billentyűt)



Mappa szerkezet

foto_batch.py → Maga a képfeldolgozó program
proba → Egy Mappa amibe rakhatod a képeket.
alap-{időbélyeg} → Készképek mappája(Ez később jelenik meg)cd

Főbb kód részek magyarázattal:

1. **Importok**

```
import argparse
import numpy as np
import cv2
from pathlib import Path
from PIL import Image, ImageOps, ImageDraw, ImageFont, ImageFilter
```

- A szükséges könyvtárakat tölti be: képfeldolgozás (Pillow, OpenCV), parancssor feldolgozás (argparse), fájlkezelés.

2. **Támogatott fájlok**

```
SUPPORTED_EXTS = {".jpg", ".jpeg", ".png", ".bmp", ".tif", ".tiff", ".webp"}
```

- A script csak ezeket a képformátumokat dolgozza fel.

3. **iter images()**

```
def iter_images(input_path: Path):···
```

- Bejárja a megadott mappát **rekurzívan**, és **visszaadja az összes képfájlt**, ami támogatott formátum.

4. **auto_orient()**

```
def auto_orient(img: Image.Image) -> Image.Image:···
```

- Automatikusan helyes tájolásba forgatja a képet EXIF metaadat alapján.

5. **detect_face_and_crop()**

```
def detect_face_and_crop(img: Image.Image) -> Image.Image:···
```

- Ez a legöszetettebb rész, **automatikus portréközépre vágás**..
- **Lépések:**

1. Pillow képből OpenCV mátrixot csinál.
2. Szürkeárnyalatúvá alakítja.
3. OpenCV Haar-cascade arcdetektálással **keresi az arcot**.
4. Ha talál:
    a. kiválasztja a legnagyobb arcot,
    b. köré rajzol egy nagyobb (50%-kal kibővített) kivágási négyzetet,
    c. levágja a képet.
5. Ha nem talál:
    a. nem módosítja a képet.

6. make_idphoto()

```
def make_idphoto(img: Image.Image, target_w: int, target_h: int) -> Image.Image:···
```

- Ez egy tipikus igazolványkép-gyártó funkció.
- A képet a megadott méretre (pl. 413×531) szabja ki úgy, hogy:
    - középre igazítja,
    - kivágja a felesleget,
    - átméretezi jó minőségű LANCZOS módszerrel,
    - fekete-fehérré + vissza RGB-vé alakítja.

7. Különböző egyszerű képmódosító funkciók:

- apply_grayscale()
    - Fekete-fehérre konvertálja a képet.
- apply_rotate()
    - A képet X fokkal elforgatja.

```
def apply_grayscale(img: Image.Image) -> Image.Image:
    return img.convert("L").convert("RGB")

def apply_rotate(img: Image.Image, angle: int) -> Image.Image:
    return img.rotate(angle, expand=True)
```

8. main() – a főprogram

- parancssori opciók feldolgozása
  - --face
  - --idphoto 413x531
  - --grayscale
  - --rotate 90
  - --blur 5
  - --sharp
  - --watermark "hello"
- Ha a felhasználó **nem adta meg a --face kapcsolót**, akkor **interaktívan rákérdez**, hogy szeretne-e arc detektálást.

```python
def main():
    parser = argparse.ArgumentParser(description="Moduláris batch képfeldolgozó")

    parser.add_argument("input", type=str, help="Bemeneti mappa vagy kép")

    # Funkciók:
    parser.add_argument("--face", action="store_true", help="Arc detektálás automatikusan")
    parser.add_argument("--idphoto", type=str, help="Igazolványkép méret, pl. 413x531")
    parser.add_argument("--grayscale", action="store_true", help="Fekete-fehér")
    parser.add_argument("--rotate", type=int, help="Elforgatás fokban")
    parser.add_argument("--blur", type=int, help="Gauss blur pixelekben")
    parser.add_argument("--sharp", action="store_true", help="Élesítés")
    parser.add_argument("--watermark", type=str, help="Vízjel szöveg")

    args = parser.parse_args()
```

9. Kimeneti mappa létrehozása

- Összegyűjti az aktivált funkciók nevét + a pontos idővel menti el a mappát.

10. Hibakezelés

- A program futás közben:
  - felismeri, ha egy fájl nem nyitható meg,
  - hiba esetén nem áll le, hanem hibát ír a fel nem dolgozott kép(ek)re és folytatja a feldolgozást.

11 AI-használat dokumentálása

A program fejlesztése során igénybe vettem AI-alapú eszközök támogatását, elsősorban ötletek és szerkezeti javaslatok formájában. A kód tényleges implementálását, ellenőrzését és a funkciók szakmai megértését ugyanakkor önállóan végeztem el.

Kódszerkezet tervezése

- a moduláris függvények elnevezése, elrendezése
- folyamatlogika (pipeline) megszervezés

Képfeldolgozó lépések optimalizálása

- arc körülvágás paddinggel
- igazolványkép-arányok kezelése

Hibaellenőrzés, kivételkezelés kialakítása

Program használat:

Alap futtatási forma:

> python **foto_batch.py** <input_mappa_vagy_kép> [opciók]

Példák a bemenetre:

- mappa: python foto_batch.py fotos/
- egyetlen kép: python app.py kep.jpg

Funkciók és opciók részletesen

1. Arc detektálás (--face) parancs

- felismeri a legnagyobb arcot
- körbevágja (50% paddal)

Ha megadod, automatikusan bekapcsol az arcfelismerés:

```
python foto_batch.py input --face
```

Ha kihagyod akkor a program rákérdez az arcfelismerésre.

2. Igazolványkép készítés (--idphoto SZÉLESSÉGxMAGASSÁG)

- középre illesztés
- átméretezés
- fekete-fehér (L → RGB) konverzió

```
python foto_batch.py input --idphoto 450x500
```

3. Fekete-fehér (--grayscale)

```
python foto_batch.py input --grayscale
```

4. **Forgatás (--rotate)**

- lehetséges értékek: 90, 180, 270

```
python foto_batch.py input --rotate 90
```

5. **Gaussian Blur/elmosás (--blur)**

- enyhe elmosás: --blur 3
- erős elmosás: --blur 10

```
python foto_batch.py input --blur 8
```

6. **Élesítés (--sharp)**

```
python foto_batch.py input --sharp
```

7. **Vízjel szöveg (--watermark ”szöveg”)**

```
python foto_batch.py input -watermark ”GAMF”
```

Dinamikus mappanév generálás

A program a kimeneti mappát *automatikusan* nevezi el annak alapján, hogy milyen opciókat használtál.

A név formátuma:

<funkciók_összefűzve>-YYYY-MM-DD-HHMM

PÉLDA:

- Parancs:
  ```
  python app.py kepek --rotate 90 --blur 5
  ```
- Mappa:
  ```
  rotate90-blur5-2025-03-10-1530/
  ```

Hivatkozások

https://www.bonza.dog/2024/12/pug-dog-health-issues-comprehensive-guide-to-common-problems/

https://filmtett.ro/uploads/Filmkepek/mr_bean.jpg

https://www.reddit.com/r/MarvelStudiosSpoilers/comments/185d1t6/marvel_studios_reportedly_wants_thor_5_to_have_a/

https://omsprecision.com/services/bone-grafting/

https://i0.wp.com/wisdomwithinct.com/wp-content/uploads/2019/12/4959E3DE-5B2C-4CE6-B6D4-7B1B4ED255B1.jpeg?resize=768%2C672&ssl=1

https://cdn.britannica.com/35/233235-050-8DED07E3/Pug-dog.jpg

https://akns-images.eonline.com/eol_images/Entire_Site/20171018/rs_634x1024-171118132453-1024-2017-miss-world-miss-china.jpg?fit=around%7C634:1024&output-quality=90&crop=634:1024;center,top

www.chatgpt.com

https://gemini.google.com/app?hl=hu