Pálcika számolás feladat – H86L47

1. Program célja

Ez a program arra lett kitalálva, hogy képeken megkeresse és megszámlálja a pálcikákat, akkor is, ha a pálcikák keresztezik egymást, vagy párhuzamosan futnak.

2. Tesztelési képek

A program az images mappában lévő képekkel működik. Van benne pár tesztkép:

- palcika1.jpg
- palcika2.jpg
- palcika3.jpg
- palcika4.jpg

3. Program felépítése

3.1 Fájlrendszer

- main.py # A fő program
- line_detector.py # Vonalak detektálása
- image_processor.py # A képfeldolgozásért felelős
- constants.py # Itt vannak a beállítások
- images # A bemeneti képek
    - palcika1.jpg
    - palcika2.jpg
    - palcika3.jpg
    - palcika4.jpg
- output #Kimeneti képek mappája

3.2 Osztályok és függvények

Osztály:

- **LineDetector osztály (line_detector.py):** Ez az osztály kezeli a vonalakat matek szempontjából. Minden metódusa statikus, szóval nem kell példányosítani.

A fontosabb metódusok:

- line_length(line):
    - Bemenet: Egy vonal koordinátái (x1, y1, x2, y2)
    - Kimenet: A vonal hossza pixelben

    - Működés: Euklideszi távolságot számol
    - Használat: Vonalak szűrésénél, összehasonlításnál
- get_line_angle(line):
    - Bemenet: Egy vonal koordinátái
    - Kimenet: A vonal szöge fokban (0-180°)
    - Működés: Arkusz tangenst számol, aztán fokba konvertálja
    - Használat: Párhuzamosság, kereszteződések vizsgálatánál
- find_intersection(line1, line2):
    - Bemenet: Két vonal koordinátái
    - Kimenet: A metszéspont koordinátái, vagy None ha nincs
    - Működés:
        - Megnézi, mekkora a szögek különbsége
        - Megold egy lineáris egyenletrendszert
        - Ellenőrzi, hogy a metszéspont a vonalszakaszon van-e
    - Használat: Kereszteződések keresésénél
- are_lines_parallel_and_close(line1, line2, max_angle_diff=15, max_distance=60):
    - Bemenet: Két vonal és opcionális paraméterek
    - Kimenet: True ha a vonalak párhuzamosak és közel vannak egymáshoz, False egyébként
    - Működés:
        - Kiszámolja a szögek különbségét
        - Megnézi a középpontok távolságát
        - Kiszámolja a vonalak közötti legkisebb távolságot
        - Irányvektorokkal is ellenőrzi
    - Használat: Párhuzamos vonalak csoportosításánál
- merge_lines(lines, min_distance=60, min_angle_diff=15):
    - Bemenet: Vonalak listája és opcionális paraméterek
    - Kimenet: A vonalak összevonva

- Működés:
    - Rendezi a vonalakat a hosszúságuk szerint
    - Csoportosítja a párhuzamos vonalakat
    - Összeköti a legtávolabbi pontokat
- Használat: Ha a vonalak szaggatottak, ezeket össze lehet fogni

Osztály:

- **ImageProcessor osztály (image_processor.py):** Ez az osztály felelős a képfeldolgozási dolgokért. Itt is minden metódus statikus.

A fontosabb metódusok:

1. preprocess_image(image):
    - Bemenet: Egy kép (BGR színtérben)
    - Kimenet: Egy előfeldolgozott bináris kép és egy éldetektált kép
    - Működés:
        - Szürkeárnyalatossá konvertálja a képet
        - Gauss-szal elmossa (hogy ne legyen olyan zajos)
        - Adaptív küszöbölést alkalmaz (így bináris lesz a kép)
        - Morfológiai nyitást csinál (ez is a zaj ellen jó)
        - Canny él-detektálást alkalmaz (hogy megtalálja a vonalakat)
    - Használat: A nyers képet előkészíti a vonaldetektáláshoz
2. detect_lines(edges):
    - Bemenet: Egy éldetektált bináris kép
    - Kimenet: A megtalált vonalak listája
    - Működés:
        - Probabilisztikus Hough transzformációt alkalmaz
        - Paraméterekkel szűri a vonalakat
    - Használat: A vonalak kezdeti megtalálásához
3. generate_distinct_colors(n):
    - Bemenet: A szükséges színek száma

- Kimenet: Egy lista BGR színekkel
- Működés:
  - HSV színtérben generál egyenletes eloszlású színeket
  - Ezeket BGR-be konvertálja
- Használat: A vonalcsoportok vizualizációjához (mindegyik más színű lesz)

constants.py

Ebben a fájlban vannak a program beállításai, pl. a legkisebb vonalhossz, a párhuzamos vonalak távolsága, meg a Hough transzformáció beállításai. Ezekkel lehet finomhangolni a programot.

main.py

Ez a fő program, ami összeköti a dolgokat, és elindítja a képfeldolgozást.

4. Lépésről lépésre a megvalósítás

4.1 Kép előfeldolgozás

1. Szürkeárnyalatossá alakítás
2. Gauss-féle elmosás (zajcsökkentés)
3. Adaptív küszöbölés
4. Morfológiai műveletek
5. Canny él-detektálás

4.2 Vonalak detektálása

1. Hough transzformáció
2. Paraméterek (a constants.py-ban található):
   - rho = 1
   - theta = pi/180
   - threshold = 80
   - minLineLength = 150
   - maxLineGap = 20

4.3 Vonalak feldolgozása

1. Szűrés a vonalak hossza alapján (MIN_LINE_LENGTH)
2. Párhuzamos vonalak keresése:

- Szögek különbsége (max 15° - MAX_ANGLE_DIFF)
- Távolság (MAX_PARALLEL_DISTANCE)

3. Kereszteződések detektálása:
    - Szögkülönbség (MIN_ANGLE_DIFF)
    - Metszéspont számítása, ellenőrzése

4.4 Vonalak csoportosítása

1. Nem kereszteződő vonalak külön csoportba
2. Kereszteződő vonalak kezelése:
    - Közeli kereszteződések összevonása
    - Csoportosítás a szögek alapján
    - Összekapcsolódó vonalak összegyűjtése

4.5 Eredmények megmutatása

1. Generálunk egyedi színeket a vonalcsoportoknak
2. Kirajzoljuk a vonalakat a megfelelő színnel
3. A kereszteződéseket bekarikázzuk pirossal
4. Kiírjuk, hány vonalcsoport van (a konzolra)
5. Létrehozunk egy ablakot:
    - Lehessen átméretezni
    - Mindig felül legyen
    - Nyomj egy gombot, hogy bezáródjon

5. Hogyan kell használni a programot?

5.1 Indítás

1. A parancssorban menj a projekt mappájába.
2. Írd be, hogy python main.py
3. Válaszd ki, melyik képet akarod feldolgozni: a program kiírja, melyik szám melyik kép.

5.2 Eredmények

1. Az eredményablak automatikusan megjelenik.
2. Átméretezhető, mozgatható.

5.3 A program által létrehozott fájlok

Ezek az output mappába kerülnek:

- *_binary.jpg: A binárissá alakított kép
- *_edges.jpg: Ahol az élek vannak
- *_result.jpg: A végeredmény, bejelölve mindent

6. Beállítások módosítása

A constants.py fájlban lehet a program beállításait módosítani.

6.1 Vonal detektálás

- MIN_LINE_LENGTH: A legkisebb vonalhossz (alapból: 200)
- MIN_ANGLE_DIFF: A legkisebb szög, aminek a vonalaknak kell lennie, hogy keresztezzék egymást (alapból: 30)
- MAX_PARALLEL_DISTANCE: A maximális távolság, amilyen messze lehetnek a párhuzamos vonalak egymástól (alapból: 50)
- MIN_LENGTH_RATIO: Minimális hosszarány (alapból: 0.5)
- MAX_ANGLE_DIFF: A maximális szögkülönbség a párhuzamos vonalak vizsgálatánál. (alapból: 15)

6.2 Hough transzformáció

- threshold: Akkumulátor küszöbérték (alapból: 80)
- minLineLength: Minimális vonalhossz (alapból: 150)
- maxLineGap: Maximális vonalrés (alapból: 20)

7. A kód magyarázata (példák)

7.1 Példa: preprocess_image függvény (image_processor.py)

```python
class ImageProcessor:  4 usages  odrykrisztina

    """ Képfeldolgozási műveletek végrehajtása """
    @staticmethod  1 usage  odrykrisztina
    def preprocess_image(image):
        """
        Args:
            image: BGR színtérben lévő bemeneti kép

        Returns:
            tuple: (binary, edges)
                - binary: Binarizált kép
                - edges: Éldetektált kép
        """

        # Szürkeárnyalatos konverzió
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        # Gauss-féle elmosás a zaj csökkentésére
        # 5x5-ös kernel méret, 0 szigma érték (automatikus számítás)
        blurred = cv2.GaussianBlur(gray, ksize: (5, 5), sigmaX: 0)

        # Adaptív küszöbölés a binarizáláshoz
        # - 255: maximális érték
        # - ADAPTIVE_THRESH_GAUSSIAN_C: Gaussian-alapú adaptív küszöbölés
        # - THRESH_BINARY_INV: Invertált bináris kép
        # - 11: A környezet mérete (11x11 pixel)
        # - 2: Konstans kivonás a számított küszöbértékből
        binary = cv2.adaptiveThreshold(blurred, maxValue: 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                       cv2.THRESH_BINARY_INV, blockSize: 11, C: 2)

        # Morfológiai nyitás a zaj további csökkentésére
        # 3x3-as kernel az apró zajok eltávolításához
        kernel = np.ones((3, 3), np.uint8)
        binary = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)

        # Canny él-detektálás
        # - 30: alsó küszöbérték
        # - 150: felső küszöbérték
        edges = cv2.Canny(binary, 30, 150)

        return binary, edges
```

Ez a függvény csinálja a kép előkészítését:

1. **Szürkeárnyalatos konverzió:** A színes képet szürkeárnyalatossá alakítja.
2. **Gauss-féle elmosás:** Elmossa a képet, hogy kevésbé legyen zajos.
3. **Adaptív küszöbölés:** A képet binárissá alakítja (fekete-fehér).

4. **Morfológiai nyitás:** Eltünteti a kisebb zajokat.
5. **Canny él-detektálás:** Megkeresi a képen az éleket (vonalakat).

7.2 Példa: find_intersection függvény (line_detector.py)

```
""" Két vonal metszéspontjának meghatározása """
@staticmethod  1 usage  odrykrisztina
def find_intersection(line1, line2):
    """
    Args:
        line1: Első vonal koordinátái [[x1, y1, x2, y2]] formátumban
        line2: Második vonal koordinátái [[x1, y1, x2, y2]] formátumban

    Returns:
        tuple vagy None: (x, y) metszéspont koordinátái, vagy None ha nincs metszéspont
    """
    x1, y1, x2, y2 = line1[0]
    x3, y3, x4, y4 = line2[0]

    # Szögek ellenőrzése
    angle1 = LineDetector.get_line_angle(line1)
    angle2 = LineDetector.get_line_angle(line2)
    angle_diff = abs(angle1 - angle2)

    # 90 foknál nagyobb szögkülönbség esetén a kiegészítő szöget vesszük
    if angle_diff > 90:
        angle_diff = 180 - angle_diff

    # Ha a szögkülönbség kisebb mint a minimum, nincs érvényes kereszteződés
    if angle_diff < MIN_ANGLE_DIFF:
        return None

    # Metszéspont számítása lineáris egyenletrendszer megoldásával
    # A vonalak egyenletei: ax + by = c formában
    a1 = y2 - y1
    b1 = x1 - x2
    c1 = a1 * x1 + b1 * y1
    a2 = y4 - y3
    b2 = x3 - x4
    c2 = a2 * x3 + b2 * y3
    det = a1 * b2 - a2 * b1

    # Ha a determináns 0, a vonalak párhuzamosak
    if det == 0:
        return None

    # Metszéspont koordinátáinak számítása
    x = (b2 * c1 - b1 * c2) / det
    y = (a1 * c2 - a2 * c1) / det
```

```
    # Ellenőrizzük, hogy a metszéspont a vonalszakaszokon van-e
    if (min(x1, x2) <= x <= max(x1, x2) and
            min(y1, y2) <= y <= max(y1, y2) and
            min(x3, x4) <= x <= max(x3, x4) and
            min(y3, y4) <= y <= max(y3, y4)):
        return int(x), int(y)

    return None
```

Ez a függvény megkeresi két vonal metszéspontját.

1. **Bemenet:** Két vonal koordinátái.
2. **Determináns:** Ha a determináns nulla, a vonalak párhuzamosak.
3. **Metszéspont:** Kiszámolja a metszéspontot.
4. **Ellenőrzés:** Megnézi, hogy a metszéspont a vonalszakaszon van-e.
5. **Kimenet:** A metszéspont koordinátái, vagy None, ha nincs metszéspont.