Nohai Alexandru
Grupa 334

Documentație: Concepte și Aplicații în Vederea Artificială - Mathable

1. Încărcarea și afișarea imaginilor

Pasul 1: Încărcarea unei imagini

- Se încarcă o imagine dintr-un fișier folosind funcția OpenCV cv2.imread(). Imaginea este stocată într-o variabilă (de exemplu, image).

Pasul 2: Afișarea imaginii

- Imaginea încărcată este redimensionată (în acest caz, la 20% din dimensiunea originală) pentru a fi ușor vizualizată.
- Se utilizează funcția cv2.imshow() pentru a afișa imaginea într-o fereastră.

```
def show_image(title, image):
    image_resized = cv2.resize(image, (0, 0), fx=0.2, fy=0.2)
    cv2.imshow(title, image_resized)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

---

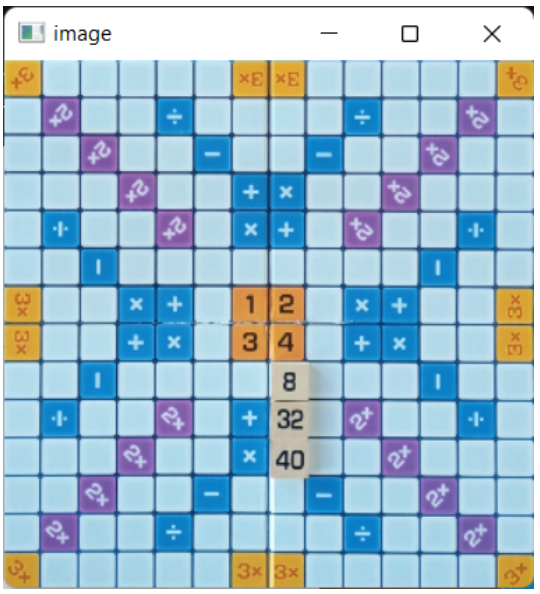
2. Detectarea tablei de joc și aplicarea unei transformări în perspectivă

Pasul 1: Detectarea pătratului în imagine

- Imaginea este convertită în spațiul de culoare HSV (cv2.cvtColor()).
- Se aplică o mască pe baza unui interval de culori pentru a izola zonele care reprezintă pătratul.
- Se utilizează cv2.findContours() pentru a detecta contururile din masca obținută.

Pasul 2: Aplicarea transformării în perspectivă

- Se identifică cel mai mare contur (care reprezintă pătratul dorit).
- Se aplică o transformare în perspectivă folosind funcția cv2.getPerspectiveTransform().
- Imaginea este decupată pentru a izola doar pătratul și este redimensionată pentru a avea dimensiuni fixe.




```
def extrage_careu(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    lower_range = np.array([40, 40, 40])
    upper_range = np.array([80, 255, 255])
    mask = cv2.inRange(hsv, lower_range, upper_range)
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    largest_contour = max(contours, key=cv2.contourArea)
    rect = cv2.minAreaRect(largest_contour)
    box = cv2.boxPoints(rect)
    box = np.int0(box)

    width, height = 200, 200
    pts1 = np.float32(box)
    pts2 = np.float32([[0, 0], [0, height], [width, height], [width, 0]])
    M = cv2.getPerspectiveTransform(pts1, pts2)
    dst = cv2.warpPerspective(image, M, (width, height))

    return dst
```

---

3. Aflarea poziției piesei adaugate pe tablă

Pasul 1: Diferenta absoluta dintre poza curenta si poza anterioara

- Se foloseste cv.absdiff intre canalele v (value) extrase dupa conversia imaginilor la HSV ale celor doua poze
- Se aplica filtre de GaussianBlur, treshold, erode si dilate pentru a curata mastile si pentru a avea acuratete cat mai ridicata

---

4. Determinarea centrului unei regiuni albe în imagine

Pasul 1: Găsirea centrului regiunii albe

- Se aplică o mascare pe imagine pentru a evidenția regiunile albe (de exemplu, utilizând un prag).
- Se detectează contururile regiunilor albe și se calculează centrul regiunii celei mai mari folosind momentele conturului.

```
def find_largest_region_center(mask):
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    largest_contour = max(contours, key=cv2.contourArea)
    moments = cv2.moments(largest_contour)
    cX = int(moments["m10"] / moments["m00"])
    cY = int(moments["m01"] / moments["m00"])
    return (cX, cY)
```

---

5. Determinarea celulei de grilă în care se află centrul regiunii albe

Pasul 1: Împărțirea imaginii în grilă

- Imaginea este împărțită într-o grilă de 14x14 celule, iar pentru fiecare punct, se determină celula corespunzătoare în funcție de coordonatele sale.
- Se calculează rândul și coloana pe baza poziției punctului în imagine.

```
def find_grid_cell(center):
    cell_size = 2030 // 14
    center_x, center_y = center
    row = center_y // cell_size
    col = center_x // cell_size
    column_letter = chr(ord('A') + col)
    return f"{row + 1}{column_letter}"
```

.

6. Identificarea numerelor de pe piese

Pasul 1:Determinarea numărului de cifre

Funcția `determina_numar_cifre` folosește detectarea contururilor pentru a determina câte cifre sunt vizibile pe piesă

- **Detecția contururilor**: Se folosesc contururi mari pentru a identifica cifrele.
- **Filtrarea contururilor**: Se elimină zgomotul, păstrând doar contururile relevante.
- **Numărul de cifre**: Dacă sunt două contururi valabile, piesa conține două cifre; altfel, o singură cifră.

Pasul 2: Template Matching

- După stabilirea numărului de cifre, se aplică tehnica de `template matching` pentru a identifica fiecare cifră:
- **Aplicarea template matching**: Se compară imaginea piesei cu template-urile pentru a găsi cea mai mare corelație.
- **Determinarea cifrei**: Se returnează cifra corespunzătoare pe baza celei mai mari corelații.