Documentație Tehnică: Detectarea și Evaluarea Pieselor Qwirkle

Tema 1 - Computer Vision

Decembrie 2025

1 Prezentare Generală

Sistemul implementat detectează automat piesele dintr-o imagine a jocului Qwirkle, extrage tabla de joc din fundal, identifică formele și culorile pieselor, calculează scorul conform regulilor oficiale Qwirkle, și identifică pătratele bonus pe baza configurației inițiale a jocului.

Întreaga implementare se află într-un singur fișier Python: `solutie.py` (515 linii).

2 Arhitectura Soluției

2.1 Extragerea Tablei de Joc (HSV Masking)

Funcția `extrage_careu()` extrage tabla de joc folosind o abordare bazată pe **mascare HSV** a fundalului, urmată de transformare perspectivă:

1. **Conversie în spațiul HSV**:

```
hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
```

   Spațiul HSV permite separarea mai ușoară a culorilor comparativ cu BGR.

2. **Mascare culoare pentru fundal**: Se creează o mască pentru fundalul maro/masă folosind intervalul HSV [0,50,20] - [30,255,200]:

```
lower = np.array([0, 50, 20])
upper = np.array([30, 255, 200])
mask = cv.inRange(hsv, lower, upper)
mask = cv.bitwise_not(mask)  # Inversare: 1=tabla, 0=fundal
```

   Acest interval captează tonurile de maro/bej ale mesei, iar inversarea măștii izolează tabla de joc.

3. **Operații morfologice**: Pentru eliminarea zgomotului se aplică:

```
kernel = np.ones((5, 5), np.uint8)
mask = cv.morphologyEx(mask, cv.MORPH_OPEN, kernel)
mask = cv.morphologyEx(mask, cv.MORPH_CLOSE, kernel)
```

MORPH_OPEN elimină pixel-ii izolați, iar MORPH_CLOSE umple golurile mici.

4. **Detecție contururi**: Se găsesc contururile externe și se sortează descrescător după arie:

```
contours, _ = cv.findContours(mask, cv.RETR_EXTERNAL,
                              cv.CHAIN_APPROX_SIMPLE)
contours = sorted(contours, key=cv.contourArea, reverse=True)
```

5. **Filtrare geometrică**: Pentru fiecare contur se verifică:
   - Arie minimă: ¿5% din aria imaginii
   - Raport aspect: 0.5 ¡ w/h ¡ 2.0 (aproape pătrat)
   - Soliditate: ¿0.7 (raport între arie și convex hull)

6. **Aproximare 4 colțuri**: Se folosește convex hull și aproximare poligonală:

```
hull = cv.convexHull(cnt)
perimeter = cv.arcLength(hull, True)
approx = cv.approxPolyDP(hull, 0.02 * perimeter, True)
if len(approx) == 4:
    # Am gasit tabla!
```

7. **Transformare perspectivă**: Se ordonează cele 4 puncte (top-left, top-right, bottom-right, bottom-left) și se aplică transformare către dimensiunea țintă:

```
padding = 50  # pixeli extra pe fiecare parte
width = height = 1600 + 2 * padding  # 1700x1700
M = cv.getPerspectiveTransform(rect, destination_of_puzzle)
result = cv.warpPerspective(img, M, (width, height))
```

   Rezultatul este o imagine 1700×1700 pixeli cu tabla normalizată la 1600×1600 și 50 pixeli padding pe fiecare latură.

2.2 Detectarea Formelor (Template Matching cu Rotații)

Funcția `match_cell()` folosește template matching pentru identificarea formelor:

1. **Încărcarea template-urilor**: Funcția `load_templates()` citește imaginile din folderul `templates/` și subfolderele sale:

```
img_hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
templates.append((label, img_hsv))
```

   Eticheta (label) este numele subfolderului (ex: ”cerc”, ”patrat”, ”romb”, ”trifoi”, ”stea”, ”shuri”).

2. **Matching cu 4 rotații**: Pentru invarianță la rotație, fiecare template este testat la 0°, 90°, 180°, 270°:

```
for angle in [0, 90, 180, 270]:
    if angle == 90:
        rotated_tmpl = cv.rotate(tmpl, cv.ROTATE_90_CLOCKWISE)
    # ... (180, 270 similar)
    res_v = cv.matchTemplate(cell_hsv[:,:,2],
```

```
                                rotated_tmpl[:,:,2],
                                cv.TM_CCOEFF_NORMED)
    score = np.max(res_v)
```

Matching-ul se face pe canalul V (brightness) din spațiul HSV, care corespunde unei imagini grayscale.

3. **Sistem de votare top-K**: Se păstrează toate potrivirile peste threshold (0.55) și se votează:

```
votes = {}
for score, label in top_matches[:top_k]:
    if label not in votes:
        votes[label] = 0
    votes[label] += score
best_label = max(votes, key=votes.get)
```

4. **Regula exact match**: Dacă scorul ¿0.95, se returnează direct acea potrivire fără votare.

Pentru fiecare celulă din grilă (100×100 pixeli), se extrage un patch mai mare (150×150, cu margin de 25 pixeli) pentru matching mai robust la margini.

2.3 Detectarea Culorilor (Multi-Point Sampling)

Funcția `detect_color()` implementează o strategie avansată pentru detectarea robustă a culorii:

1. **Sampling din 4 puncte cardinale**: În loc să sample doar din centru (care poate fi umbră sau parte neagră a formei), se prelevează din 4 regiuni:

```
sample_points = [
    (center_y - 15, center_x),      # sus
    (center_y + 15, center_x),      # jos
    (center_y, center_x - 15),      # stanga
    (center_y, center_x + 15),      # dreapta
]
# Pentru fiecare punct, regiune 5x5
region = cell_hsv[max(0, py-2):min(h, py+3),
                  max(0, px-2):min(w, px+3)]
```

2. **Filtrare pixeli luminoși**: Se păstrează doar pixelii cu V¿60 pentru a elimina umbrele:

```
all_pixels = np.array(all_pixels)
bright_pixels = all_pixels[all_pixels[:, 2] > 60]
```

Această tehnică elimină problema detecției eronate cauzate de sampling din zonele întunecate/umbrite.

3. **Calcul median HSV**: Se calculează mediana valorilor H (Hue), S (Saturation), V (Value):

```
h_median = np.median(bright_pixels[:, 0])
s_median = np.median(bright_pixels[:, 1])
v_median = np.median(bright_pixels[:, 2])
```

4. **Clasificare după intervale HSV**:

   - **White (W)**: $S \leq 50$ și $V \geq 60$
   - **Red (R)**: $H \geq 165$ sau $155 \leq H < 165$
   - **Orange (O)**: $H \leq 18$
   - **Yellow (Y)**: $18 < H \leq 35$
   - **Green (G)**: $40 \leq H \leq 80$
   - **Blue (B)**: $90 < H \leq 130$

   Intervalele au fost ajustate empiric pentru a separa corect portocaliul de roșu și albastrul de verde.

Culoarea este detectată din **celula fără margin** (100×100), nu din patch-ul extins, pentru a evita captarea liniilor verzi ale grilei.

2.4 Detectarea Pătratelor Bonus (Pattern Matching)

Funcția `detecteaza_bonus_pattern()` determină locațiile pătratelor bonus pe baza configurației inițiale (`x_00.jpg`):

- Tabla este împărțită în **4 cadrane**:
  - Cadran 1: rânduri 1-8, coloane A-H
  - Cadran 2: rânduri 1-8, coloane I-P
  - Cadran 3: rânduri 9-16, coloane A-H
  - Cadran 4: rânduri 9-16, coloane I-P
- Pentru fiecare cadran se verifică o **poziție cheie**: 2B, 2J, 10B, 10J
- În funcție de ocuparea poziției cheie, se aplică unul din 2 pattern-uri predefinite:

```
if '2B' not in piese_set:
    bonus_2_set.update(['2B', '7G'])
    bonus_1_set.update(['6B', '5C', '4D', '3E', '2F',
                        '7C', '6D', '5E', '4F', '3G'])
else:
    bonus_2_set.update(['7B', '2G'])
    bonus_1_set.update(['2C', '3D', '4E', '5F', '6G',
                        '3B', '4C', '5D', '6E', '7F'])
```

- Returnează 2 set-uri: `bonus_1_set` (pătrate +1) și `bonus_2_set` (pătrate +2)

Această abordare permite detectarea dinamică a bonusurilor fără a fi hard-coded pentru o anumită configurație.

2.5 Calculul Scorului (Reguli Qwirkle)

Funcția `calculeaza_scor_mutare()` implementează regulile oficiale Qwirkle:

1. **Construire board**: Se creează un dicționar cu toate piesele (vechi + noi):

```
board = {}
for coord, shape, color in toate_piesele:
    row, col = coord_to_pos(coord)
    board[(row, col)] = (coord, shape, color)
```

2. **Detectare linii pentru fiecare piesă nouă**: Pentru fiecare piesă nouă, se verifică liniile orizontale și verticale continue:

```
for direction in ['H', 'V']:
    if direction == 'H':
        linie = [(row, c) for c in range(1, 17)
                 if (row, c) in board]
    else:
        linie = [(r, col) for r in range(1, 17)
                 if (r, col) in board]
```

3. **Găsire secvență continuă**: Se identifică secvența continuă care conține piesa curentă (fără goluri).

4. **Scor linie**: Dacă linia are ¿1 piesă și nu a fost deja calculată:

```
puncte_linie = len(secventa)
if len(secventa) == 6:
    puncte_linie += 6  # Bonus Qwirkle
scor_total += puncte_linie
```

5. **Evitare dublă contorizare**: Fiecare linie este marcată cu un identificator unic:

```
linie_id = (direction, tuple(secventa))
if linie_id not in linii_calculate:
    linii_calculate.add(linie_id)
    # ... calcul scor
```

6. **Bonus pătrate speciale**: Se adaugă +1 sau +2 puncte pentru piese noi plasate pe pătrate bonus:

```
if coord in bonus_1_set:
    scor_total += 1
elif coord in bonus_2_set:
    scor_total += 2
```

7. **Caz special (piese singure)**: Dacă nicio linie nu este formată, scorul = numărul de piese noi.

3 Pipeline-ul de Procesare

Bucla principală din `solutie.py` procesează imaginile din folderul de input (implicit `antrenare/`):

1. **Detecție joc nou**: La fiecare fișier `x_00.jpg`, se resetează starea:

```
game_num = int(file.split('_')[0])
if game_num != current_game:
    current_game = game_num
    previous_pieces = set()
    if '_00' in file:
        # Se detecteaza bonusurile, dar NU se salveaza fisier
```

2. **Extragere tablă**: Se obține tabla normalizată 1700×1700:

```
board = extrage_careu(img)
gray_board = cv.cvtColor(board, cv.COLOR_BGR2GRAY)
hsv_board = cv.cvtColor(board, cv.COLOR_BGR2HSV)
```

3. **Iterare prin grilă 16×16**: Pentru fiecare celulă (CELL_SIZE=100, CELL_OFFSET=50):

```
x1 = CELL_OFFSET + col * CELL_SIZE
y1 = CELL_OFFSET + row * CELL_SIZE
x2, y2 = x1 + CELL_SIZE, y1 + CELL_SIZE

# Patch extins pentru matching (150x150)
patch_x1 = max(0, x1 - MARGIN)  # MARGIN=25
# ...
cell_hsv = hsv_board[patch_y1:patch_y2, patch_x1:patch_x2]
cell_hsv = cv.resize(cell_hsv, (PATCH_SIZE, PATCH_SIZE))

# Celula fara margin pentru culoare (100x100)
cell_hsv_no_margin = hsv_board[y1:y2, x1:x2]
```

4. **Verificare conținut întunecat**: O piesă Qwirkle are parte neagră (forma):

```
dark_pixels = np.sum(cell_gray < 100)
dark_ratio = dark_pixels / cell_gray.size
has_valid_dark_content = dark_ratio > 0.40
```

5. **Template matching + detectare culoare**:

```
is_match, score, name = match_cell(cell_hsv, templates, 0.55)
color = detect_color(cell_hsv_no_margin) if is_match else None
```

6. **Filtrări speciale**:

   - Eliminare template-uri cu "sus"/"jos" în nume (false positives)
   - Verificare confuzie cerc-romb pentru piese albe (score ¡0.88)

7. **Tracking piese noi**: Se compară cu imaginea anterioară pe baza (coord, culoare):

```
previous_coords_colors = {(coord, color)
    for coord, _, color in previous_pieces}
if (coord, color) not in previous_coords_colors:
    new_pieces.add((coord, shape_code, color))
```

Acest lucru evită false positives când forma este detectată inconsistent dar piesa e aceeași.

8. **Calcul scor + salvare**:

```
scor = calculeaza_scor_mutare(new_pieces, current_pieces,
                              bonus_1_set, bonus_2_set)
# Salvare in detectate/x_YY.txt
```

Format fișier: fiecare linie = `<coord> <cod_forma><culoare>`, ultima linie = `<scor>`.

Coduri forme: cerc=1, trifoi=2, romb=3, patrat=4, shuri=5, stea=6.

4 Rularea Codului

4.1 Pregătirea Mediului

Instalați OpenCV și NumPy:

```
pip install opencv-python==4.10.0.84 numpy==1.26.4
```

4.2 Structura Folderelor

```
CavaTema1/
  solutie.py
  evalueaza_detectie.py
  README.txt
  documentatie.tex
  antrenare/
    1_00.jpg, 1_01.jpg, ..., 1_20.jpg
    1_01.txt, ..., 1_20.txt (ground truth)
    2_00.jpg, 2_01.jpg, ..., 2_20.jpg
    ... (5 jocuri)
  templates/
    cerc/
    patrat/
    romb/
    trifoi/
    stea/
    shuri/
  detectate/ (creat automat)
    *.txt
    *_detected.jpg (daca DEBUG_MODE=True)
  evaluare/
    fake_test/
    cod_evaluare/
      evalueaza_solutie.py
```

4.3 Executarea Detectării

Rulare pe setul de antrenare:

```
python solutie.py
```

Rulare pe fake_test:

```
python solutie.py evaluare/fake_test evaluare/fake_test/detectate
```

Rezultate în folderul de output specificat:

- `x_YY.txt`: coordonate, forme, culori, scor
- `x_YY_detected.jpg`: imagini debug (dacă `DEBUG_MODE=True`)

4.4 Evaluarea Rezultatelor

Evaluare custom:

```
python evalueaza_detectie.py
```

Evaluare oficială:

```
cd evaluare/cod_evaluare
python evalueaza_solutie.py
```

Scripturile compară fișierele de output cu ground truth.

5 Rezultate

- **Training set (antrenare/)**: 100% acuratețe (8.00/8.00 puncte, 5 jocuri, 100 mutări)
- **Fake test**: 100% acuratețe (8.00/8.00 puncte, 1 joc, 20 mutări)

Abordarea HSV masking + 4-cadran bonus detection + multi-point color sampling s-a dovedit robustă la variații de iluminare, perspectivă și umbre. Sistemul de detectare a bonusurilor bazat pe verificarea ocupării colțurilor permite adaptarea automată la diferite configurații inițiale ale jocului.