POKERCV

Gwenaël SERPETTE

Université de Bretagne Sud

Vision par Ordinateur

DESCRIPTION DU PROJET

Définitions

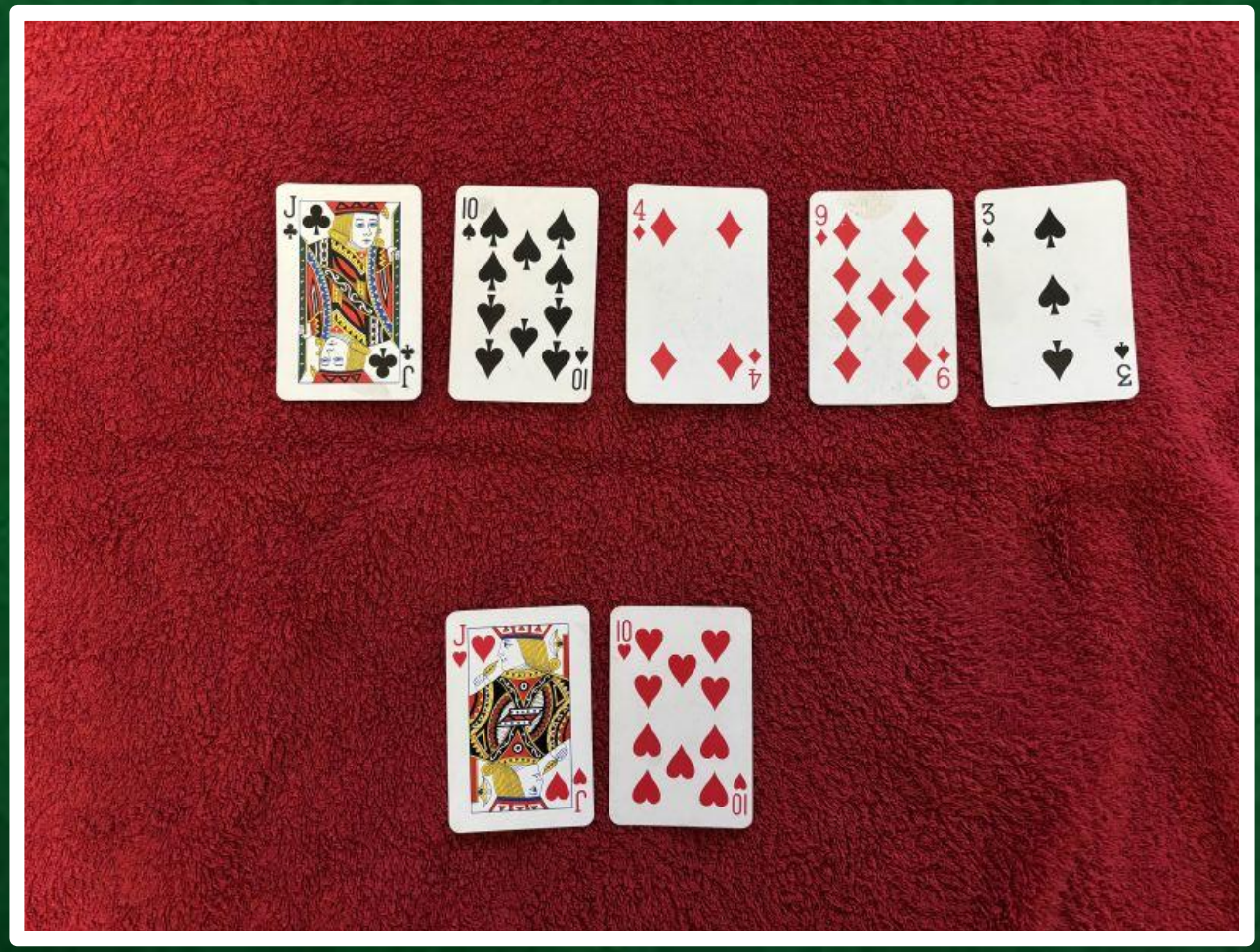
- Rang
- Couleur

Contraintes

- Fond unis
- Superposition des cartes
- Qualité suffisante
- Limité à un joueur
- Jeu de carte limité

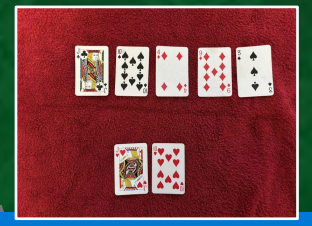
TABLE INVALIDE

TABLE VALIDE



ETAPES

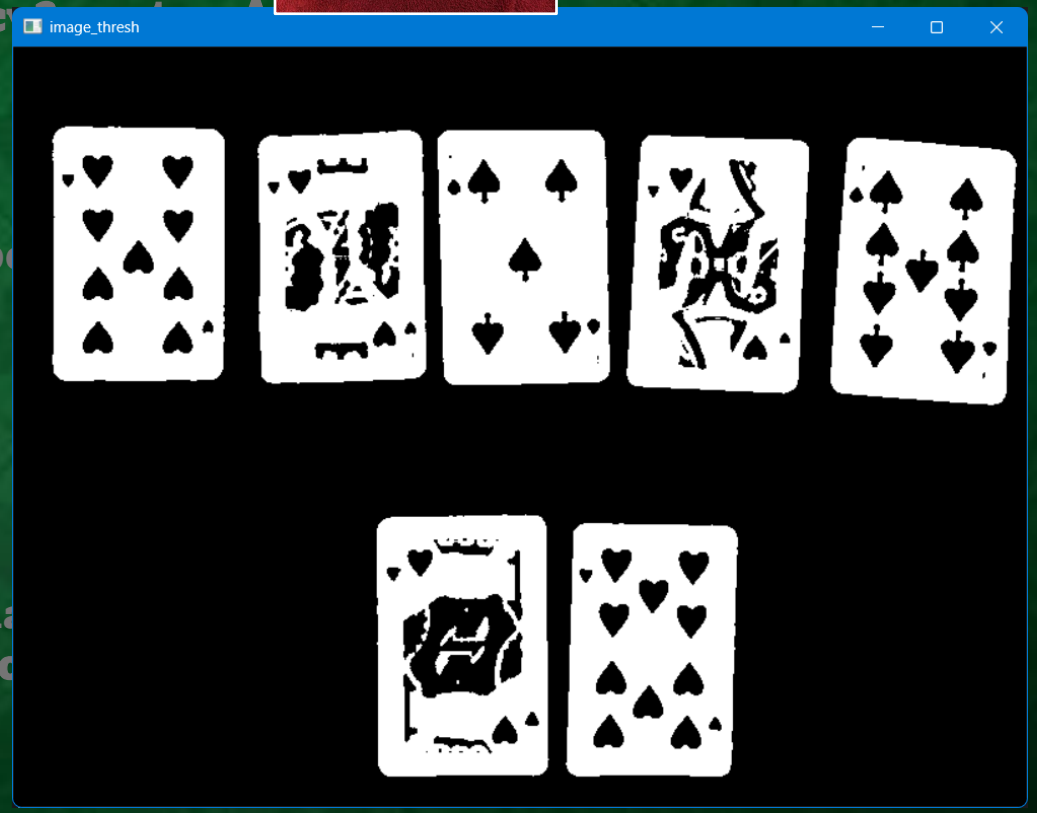
1. Chargement de l'image de la table (cv2.imread)
2. Seuillage binaire de l'image (cv2.threshold)



3. Détection des contours aves cv2.findContours et filtrage avec c

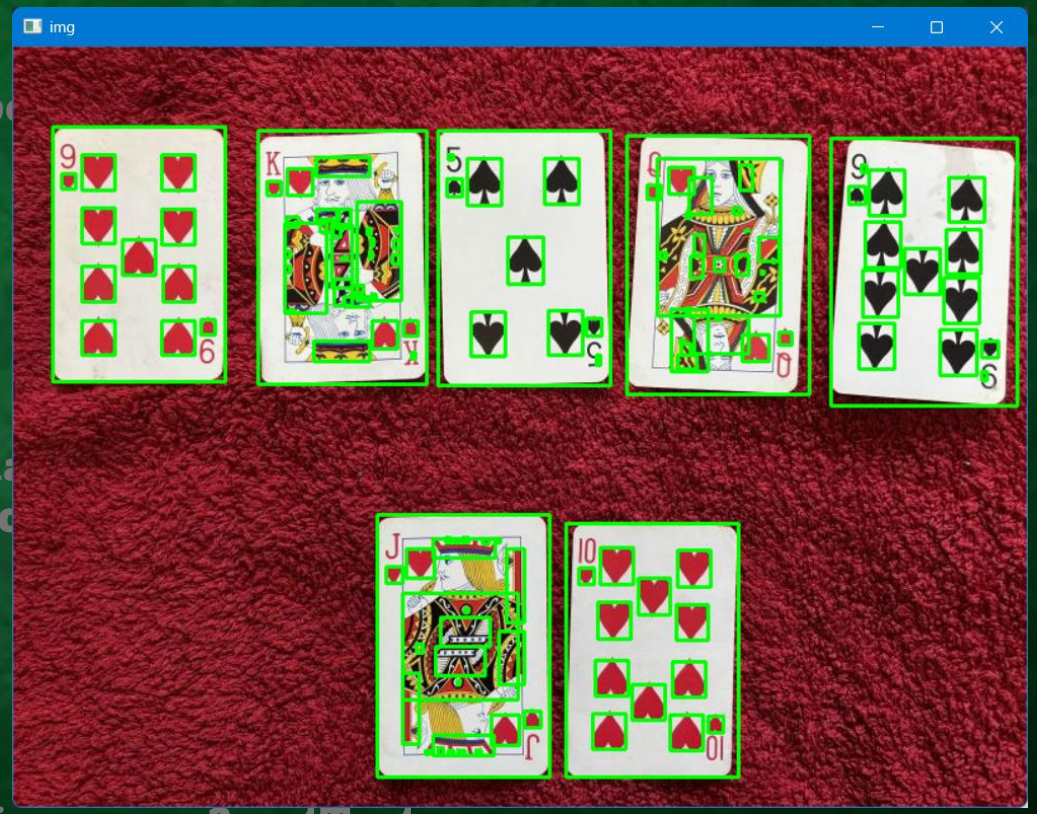
4. Pour chaque contour de carte :
    - 4.1. Perspective de la carte (cv2.warpPerspective)
    - 4.2.ROI sur le coin supérieur gauche
    - 4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste p
    - 2 plus grande aires (cv2.contourArea)
    - 4.4. Pour chaque image :
        - 4.4.1. Chargement des modèles pour la comparaison
        - 4.4.2. Redimensionne l'image et le modèle (cv2.resize)
            +
            Seuillage binaire inverse (cv2.threshold)
        - 4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque ima
        - 4.4.4. Compte le nombre de pixel blanc avec cv2.countN
        - 4.4.5. On garde le nom du modèle qui en a le moins.

    - 4.5. Moyenne des positions des pixels de la carte



ETAPES

1. Chargement de l'image de la table (cv2.imread)
2. Seuillage binaire de l'image (cv2.threshold)
3. Détection des contours aves cv2.findContours et filtrage avec cv2.contourArea
4. Pour chaque contour de carte :
    - 4.1. Perspective de la carte (cv2.warpPerspective)
    - 4.2.ROI sur le coin supérieur gauche
    - 4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste p
    - 2 plus grande aires (cv2.contourArea)
    - 4.4. Pour chaque image :
        - 4.4.1. Chargement des modèles pour la comparaison
        - 4.4.2. Redimensionne l'image et le modèle (cv2.resize)
            +
            Seuillage binaire inverse (cv2.threshold)
        - 4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque ima
        - 4.4.4. Compte le nombre de pixel blanc avec cv2.countN
        - 4.4.5. On garde le nom du modèle qui en a le moins.
    - 4.5. Moyenne des positions des pixels de la carte
5. Comparer toutes les mains du poker
6. Dessine les contours de cartes avec cv2.rectangle + écrit la main avec cv2.putText



ETAPES

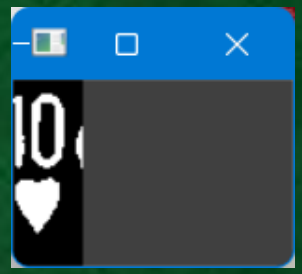
1. Chargement de l'image de la table (cv2.imread)
2. Seuillage binaire de l'image (cv2.threshold)
3. Détection des contours aves cv2.findContours et filtrage avec cv2.contourArea
4. Pour chaque contour de carte :
    - 4.1. Perspective de la carte (cv2.warpPerspective)
    - 4.2.ROI sur le coin supérieur gauche
    - 4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste p
    - 2 plus grande aires (cv2.contourArea)
    - 4.4. Pour chaque image :
        - 4.4.1. Chargement des modèles pour la comparaison
        - 4.4.2. Redimensionne l'image et le modèle (cv2.resize)
            +
            Seuillage binaire inverse (cv2.threshold)
        - 4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque ima
        - 4.4.4. Compte le nombre de pixel blanc avec cv2.countN
        - 4.4.5. On garde le nom du modèle qui en a le moins.
    - 4.5. Moyenne des positions des pixels de la carte
5. Comparer toutes les mains du poker
6. Dessine les contours de cartes avec cv2.rectangle + écrit la main avec cv2.putText

ETAPES

1. Chargement de l'image de la table (cv2.imread)
2. Seuillage binaire de l'image (cv2.threshold)
3. Détection des contours aves cv2.findContours et filtrage avec cv2.contourArea
4. Pour chaque contour de carte :
    - 4.1. Perspective de la carte (cv2.warpPerspective)
    - 4.2.ROI sur le coin supérieur gauche
    - 4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste pour garder que les 2 plus grande aires (cv2.contourArea)
    - 4.4. Pour chaque image :
        - 4.4.1. Chargement des modèles pour la comparaison
        - 4.4.2. Redimensionne l'image et le modèle (cv2.resize)
            +
            Seuillage binaire inverse (cv2.threshold)
        - 4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque image du modèle.
        - 4.4.4. Compte le nombre de pixel blanc avec cv2.countNonZero
        - 4.4.5. On garde le nom du modèle qui en a le moins.
    - 4.5. Moyenne des positions des pixels de la carte
5. Comparer toutes les mains du poker
6. Dessine les contours de cartes avec cv2.rectangle + écrit la main avec cv2.putText

ETAPES

1. Chargement de l'image de la table (cv2.imread)
2. Seuillage binaire de l'image (cv2.threshold)
3. Détection des contours aves cv2.findContours et filtrage avec cv2.contourArea
4. Pour chaque contour de carte :
    - 4.1. Perspective de la carte (cv2.warpPerspective)
    - **4.2.ROI sur le coin supérieur gauche**
    - 4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste pour garder que les 2 plus grande aires (cv2.contourArea)
    - 4.4. Pour chaque image :
        - 4.4.1. Chargement des modèles pour la comparaison
        - 4.4.2. Redimensionne l'image et le modèle (cv2.resize)
            +
            Seuillage binaire inverse (cv2.threshold)
        - 4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque image d
        - 4.4.4. Compte le nombre de pixel blanc avec cv2.countNonZe
        - 4.4.5. On garde le nom du modèle qui en a le moins.
    - 4.5. Moyenne des positions des pixels de la carte
5. Comparer toutes les mains du poker
6. Dessine les contours de cartes avec cv2.rectangle + écrit la main ave



ETAPES

1. Chargement de l'image de la table (cv2.imread)
2. Seuillage binaire de l'image (cv2.threshold)
3. Détection des contours aves cv2.findContours et filtrage avec cv2.contourArea
4. Pour chaque contour de carte :
    - 4.1. Perspective de la carte (cv2.warpPerspective)
    - 4.2.ROI sur le coin supérieur gauche
    - 4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste pour garder que les 2 plus grande aires (cv2.contourArea)
    - 4.4. Pour chaque image :
        - 4.4.1. Chargement des modèles pour la comparaison
        - 4.4.2. Redimensionne l'image et le modèle (cv2.resize)
          +
          Seuillage binaire inverse (cv2.threshold)
        - 4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaq… …èle
        - 4.4.4. Compte le nombre de pixel blanc avec cv2.co…
        - 4.4.5. On garde le nom du modèle qui en a le moins…
    - 4.5. Moyenne des positions des pixels de la carte
5. Comparer toutes les mains du poker
6. Dessine les contours de cartes avec cv2.rectangle + écrit l… …putText

ETAPES

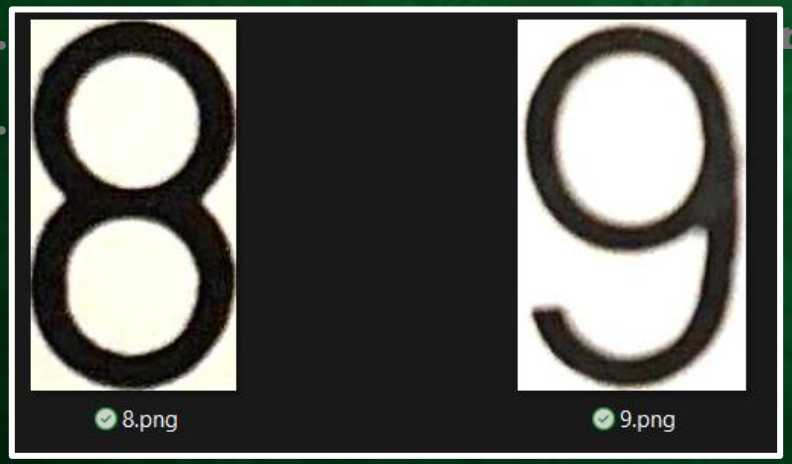
3. Détection des contours aves cv2.findContours et filtrage avec cv2.contourArea

4. Pour chaque contour de carte :
    4.1. Perspective de la carte (cv2.warpPerspective)
    4.2.ROI sur le coin supérieur gauche
    4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste pour garder que les 2 plus grande aires (cv2.contourArea)
    4.4. Pour chaque image :
        4.4.1. Chargement des modèles pour la comparaison
        4.4.2. Redimensionne l'image et le modèle (cv2.resize)
            +
            Seuillage binaire inverse (cv2.threshold)
        4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque image du modèle.
        4.4.4. Compte le nombre de pixel blanc avec cv2.countNonZero
        4.4.5. On garde le nom du modèle qui en a le moins.

    4.5. Moyenne des positions des pixels de la carte

5. r

6. cv2.re



4.2.ROI sur le coin supérieur gauche

4.3. Refaire un cv2.findContours sur le ROI + Tri de la liste pour garder que les 2 plus grande aires (cv2.contourArea)

4.4. Pour chaque image :

4.4.1. Chargement des modèles pour la comparaison

4.4.2. Redimensionne l'image et le modèle (cv2.resize)

+

Seuillage binaire inverse (cv2.threshold)

4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque image du modèle.

4.4.4. Compte le nombre de pixel blanc avec cv2.countNonZero

4.4.5. On garde le nom du modèle qui en a le moins.

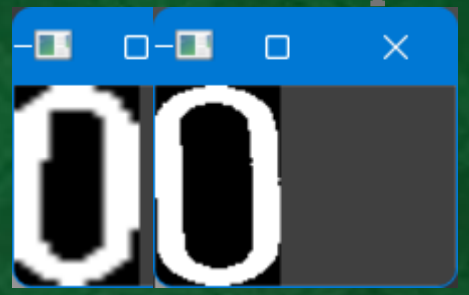
4.5. Moyenne des positions des pixels de la carte

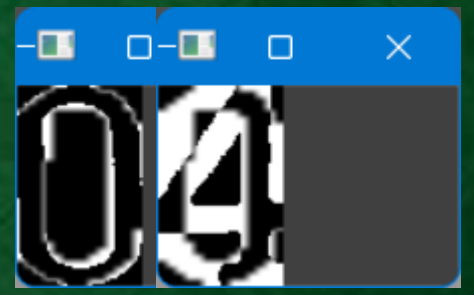
5. Comparer toutes les mains du poker

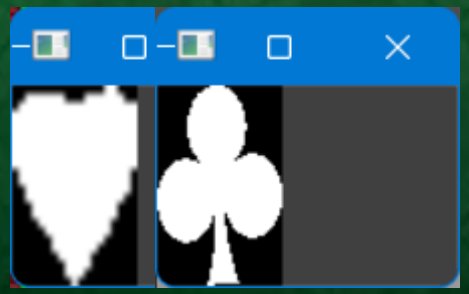
6. Dessine les contours de cartes avec cv2.rectangle + écrit la main avec cv2.putText

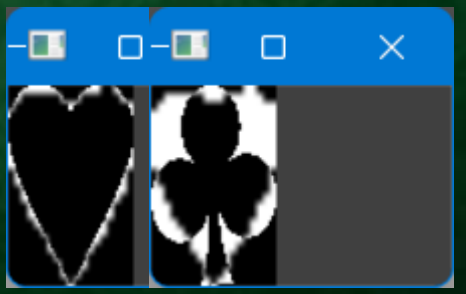
ETAPES

XOR	0	1
0	0	1
1	1	0









4.4.1. Chargement des modèles pour la comparaison
4.4.2. Redimensionne l'image et le modèle (cv2.resize)
+
Seuillage binaire inverse (cv2.threshold)
4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque image du modèle.
4.4.4. Compte le nombre de pixel blanc avec cv2.countNonZero
4.4.5. On garde le nom du modèle qui en a le moins.

ETAPES

4.5. Moyenne des positions des pixels de la carte

5. Comparer toutes les mains du poker

6. Dessine les contours de cartes avec cv2.rectangle + écrit la main avec cv2.putText

Pseudo code

```
card_center = mean( card_pixels )
if ( card_center < image_center / 2 ) :
    card_table.add( Card )
else :
    card_player.add( Card )
```

Seuillage binaire inverse (cv2.threshold)

4.4.3. XOR (cv2.bitwise_xor) entre l'image et chaque image du modèle.

4.4.4. Compte le nombre de pixel blanc avec cv2.countNonZero

4.4.5. On garde le nom du modèle qui en a le moins.

ETAPES

4.5. Moyenne des positions des pixels de la carte

5. Comparer toutes les mains du poker

6. Dessine les contours de cartes avec cv2.rectangle + écrit la main avec cv2.putText

PROBLEMATIQUES RENCONTREES

cv2.matchTemplate

Compare la carte

Compare le ROI

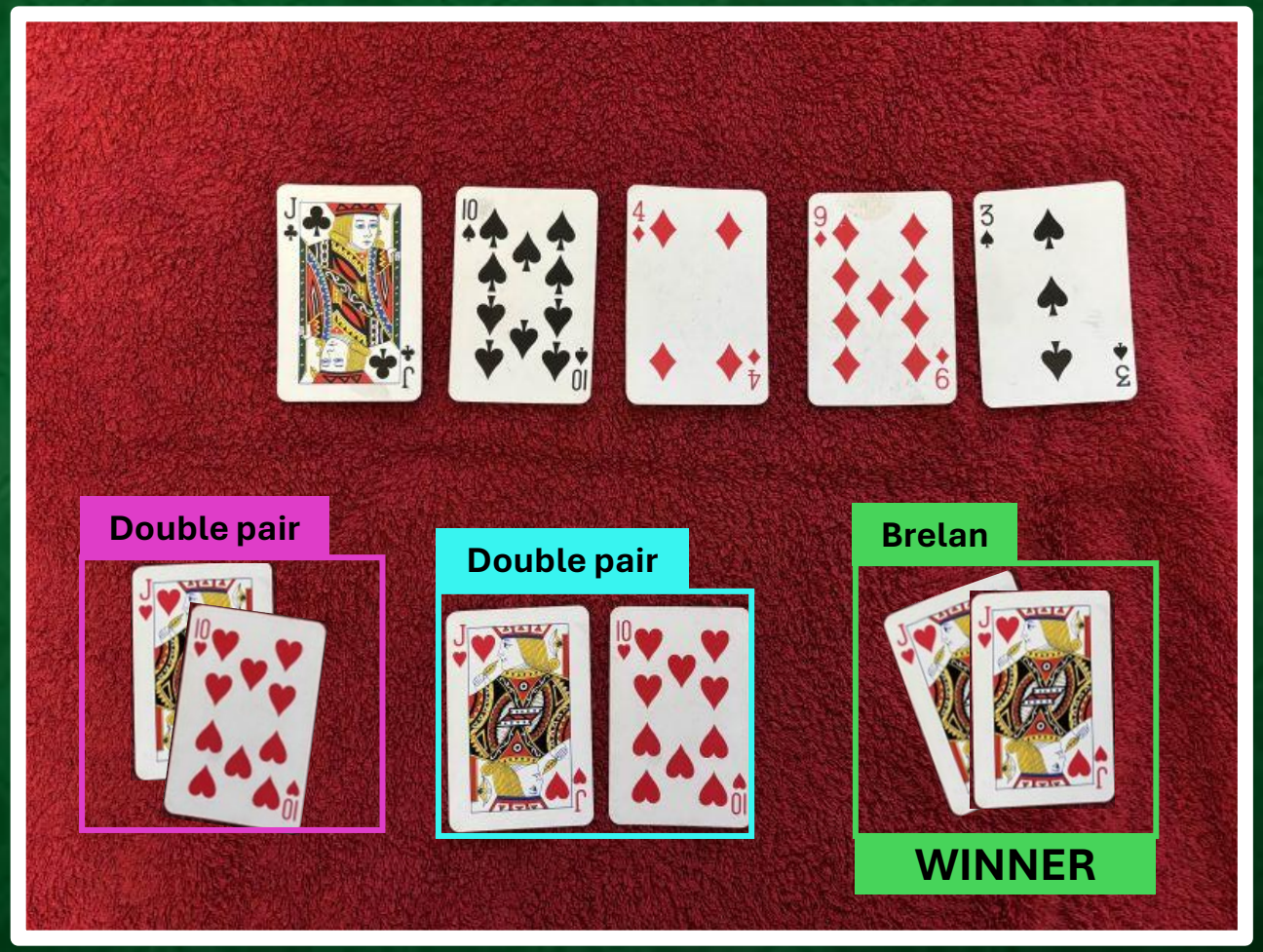
Compte les contours

PROBLEMATIQUES RENCONTREES

9 de pique de différent jeu de carte

FUTURS TRAVAUX

- **Superposition des cartes**
- **Augmenter nombre de joueur**
- **Déterminer un gagnant**
- **Adapter à la vidéo**



FUTURS TRAVAUX

- **Adapter à la vidéo**

Finale Winamax Poker Tour 2025

MERCI

Gwenaël SERPETTE

Université de Bretagne Sud

Vision par Ordinateur