UNIVERSITÉ DU QUÉBEC

RAPPORT PRÉSENTÉ À

FRANÇOIS MEUNIER

PROJET DE SESSION – PHASE 2

PAR

JÉRÉMY VEILLETTE

DYLAN SICARD-SMITH

MIRIAM DAVYDOV

29 avril 2024

Table des matières

Description de la problématique

Notre équipe de trois personnes avons travaillé sur le développement d'un programme visant à détecter les drapeaux de différents pays. Les drapeaux présentent une diversité de formes géométriques, de couleurs, et parfois des symboles uniques ou similaires. Cette combinaison en fait un choix idéal pour le traitement d'images. L'objectif principal est de fournir au programme des images de drapeaux permettant ainsi au système de deviner à quel pays chaque drapeau appartient en appliquant divers traitements.

Méthodologie de résolution

Collecte des données : Tout d'abord, nous avons établi une banque de critères/données pour chaque drapeau afin que le système puisse identifier le pays d'un drapeau après avoir traité une image fournie. Ces données incluent entre autres les couleurs et les formes contenu dans l'image.

Prétraitement des images : Les images de drapeaux sont soumises à un prétraitement pour redimensionner si nécessaire et améliorer le contraste et la répartition des valeurs de luminosité grâce à ces méthodes :

convertir_niveaux_de_gris()

1. On convertit en niveau de gris avec cv2.cvtColor() et le paramètre cv2.COLOR_BGR2GRAY.

convertir_niveaux_de_gris_ameliore()

1. Appel convertir_niveaux_de_gris()

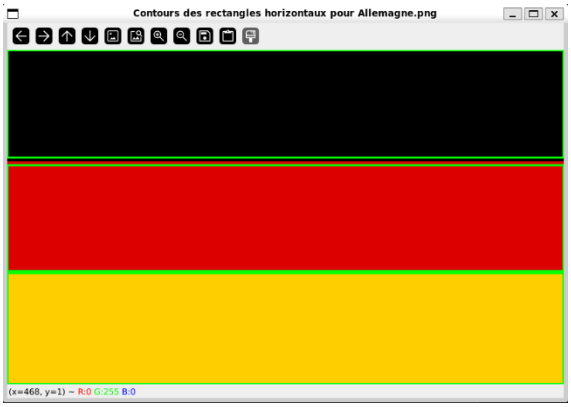
2. On normalise les valeurs de l'image en niveaux de gris pour les ramener dans la plage de 0 à 255 en utilisant la fonction cv2.normalize().

3. On applique un filtre de netteté à l'image normalisée avec cv2.filter2D().

4. On égalise l'histogramme de l'image affinée pour améliorer le contraste et la répartition des valeurs de luminosité avec cv2.equalizeHist().

Extraction et interprétation des caractéristiques : Les caractéristiques distinctives des drapeaux sont ensuite extraites à partir des images prétraitées. Cela peut inclure l'extraction de formes géométriques, l'analyse des couleurs dominantes, et la détection de symboles spécifiques. Ces caractéristiques sont ensuite validés avec la banque de critères afin d'identifier le pays du drapeau de l'image fournie. La section suivante expliquera le fonctionnement de notre programme pour différents types de drapeaux :

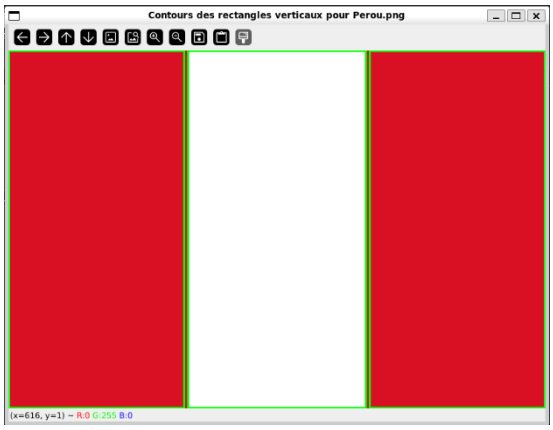
Drapeau avec Rectangles



Segmentation :

- Extraction des couleurs dans l'image.
- La conversion de l'image en niveaux de gris (convertir_niveaux_de_gris_agressif() ou convertir_niveaux_de_gris_ameliore() en fonction du niveau spécifié).
- L'application d'un seuillage adaptatif pour segmenter l'image en zones d'intérêt et d'arrière-plan (cv2.adaptiveThreshold()).
- La détection des contours dans l'image seuillée (cv2.findContours()).

Représentation/Description :

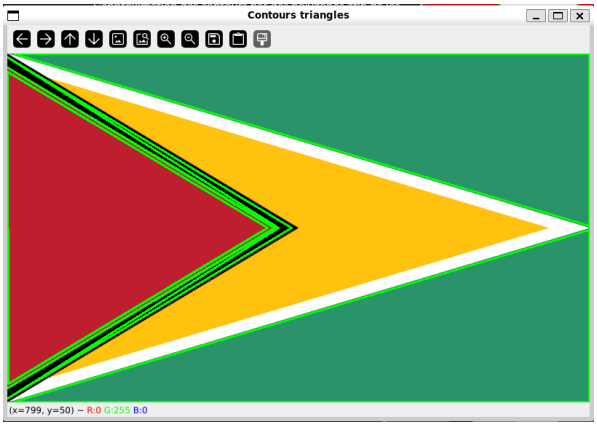


- Le calcul du périmètre des contours détectés (cv2.arcLength()).
- L'approximation des contours par des polygones afin de les représenter de manière plus simple (cv2.approxPolyDP()).
- L'identification des rectangles parmi les approximations de contours en vérifiant le nombre de côtés (4 ou 5 dans certains cas).

Reconnaissance/Interprétation :

- Les couleurs extraites correspondent aux couleurs attendues.
- La validation des rectangles en fonction de certains critères (nombre de rectangles attendus, orientation verticale ou horizontale).

Drapeau avec Triangles



Segmentation :

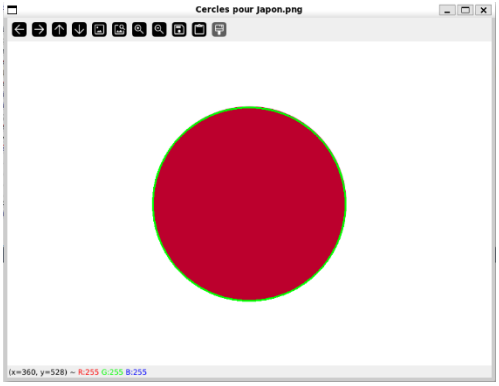
- Extraction des couleurs dans l'image.
- Conversion de l'image en niveaux de gris améliorés avec convertir_niveaux_de_gris_ameliore().
- Seuillage adaptatif pour segmenter l'image en zones d'intérêt et d'arrière-plan (cv2.adaptiveThreshold()).
- Détection des contours dans l'image seuillée (cv2.findContours()).

Représentation/Description :

- Approximation des contours par des polygones pour les représenter de manière simplifiée (cv2.approxPolyDP()).
- Calcul du périmètre des contours détectés (cv2.arcLength()).

Reconnaissance/Interprétation :

- Les couleurs extraites correspondent aux couleurs attendues.
- Validation des triangles en vérifiant le nombre de côtés (3 pour un triangle).
- Validation des triangles en fonction de certains critères spécifiques à la forme des triangles (angles internes de 180 degrés pour un triangle régulier, conditions spécifiques pour les triangles en forme de chevron).



Drapeau avec Cercles

Segmentation :

- Extraction des couleurs dans l'image.
- Conversion de l'image en niveaux de gris avec convertir_niveaux_de_gris().
- Utilisation de l'algorithme de détection de contours Canny pour détecter les bords dans l'image en niveaux de gris cv2.Canny().

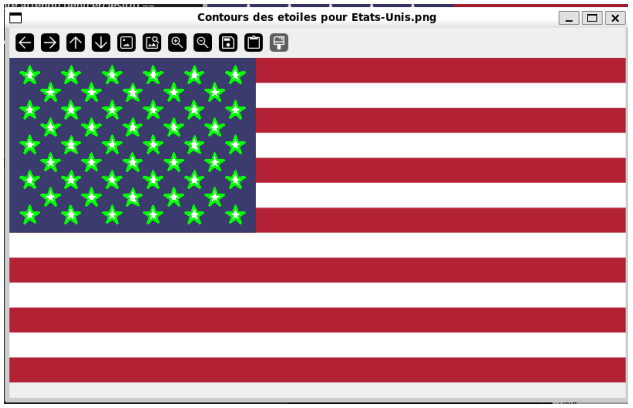
Représentation/Description :

- Détection des cercles à l'aide de la transformée de Hough pour les cercles (cv2.HoughCircles()). Cette fonction renvoie les paramètres des cercles détectés.

Reconnaissance/Interprétation :

- Les couleurs extraites correspondent aux couleurs attendues.
- Validation des cercles en vérifiant si des cercles ont été détectés et si le nombre de cercles détectés correspond au nombre attendu.

Drapeau avec Étoiles



Segmentation :

- Extraction des couleurs dans l'image.
- Conversion de l'image en niveaux de gris avec convertir_niveaux_de_gris().
- La segmentation est réalisée avec l'application d'un seuil adaptatif (adaptiveThreshold) pour binariser l'image, ce qui sépare les étoiles du fond.
- Ensuite, les contours des étoiles sont trouvés avec la fonction findContours.

Représentation/Description :

- La fonction approxPolyDP est utilisée pour approximer les contours des étoiles avec un petit nombre de côtés, permettant ainsi de représenter chaque étoile par un polygone.
- L'identification des étoiles parmi les approximations de contours en vérifiant le nombre de côtés (10).

Reconnaissance/Interprétation :

- Les couleurs extraites correspondent aux couleurs attendues.
- Validation des étoiles en vérifiant que le nombre trouvé correspond à celui attendu avec marge d'erreur (plus ou moins de 1 pour une valeur total > 0).

Drapeau avec Trapèze

Segmentation :

- Extraction des couleurs dans l'image.
- Conversion de l'image en niveaux de gris avec convertir_niveaux_de_gris_ameliore().
- La segmentation est réalisée avec l'application d'un seuil adaptatif (adaptiveThreshold) pour binariser l'image, ce qui sépare les trapèzes du fond.
- Ensuite, les contours des trapèzes sont trouvés avec la fonction findContours.

Représentation/Description :

- La fonction approxPolyDP est utilisée pour approximer les contours des trapezes avec un petit nombre de côtés, permettant ainsi de représenter chaque trapeze par un polygone.
- L'identification des trapèzes parmi les approximations de contours en vérifiant le nombre de côtés (4).
- Vérifie que les approximations correspondent aux propriétés d'un trapèze (convexité et rapport d'aspect).

Reconnaissance/Interprétation :

- Les couleurs extraites correspondent aux couleurs attendues.
- Si le nombre de trapèzes détectés correspond au nombre attendu, ils sont dessinés sur l'image à l'aide de la fonction dessiner_contours.
- Enfin, le résultat de la validation est retourné pour interprétation.

Analyse des résultats

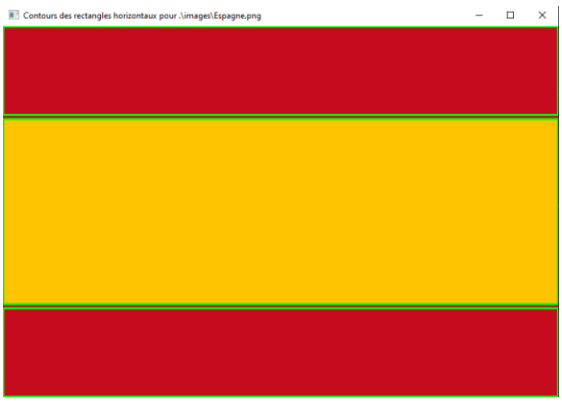
Drapeau avec Rectangles

Lorsque nous exécutons le programme avec une image du drapeau de l'Allemagne, le programme identifie bien le pays :

```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Allemagne.png
.\images\Allemagne.png
Allemagne - Valide
```

Lorsque nous exécutons le programme avec une image du drapeau de l'Allemagne, le programme identifie bien le pays :

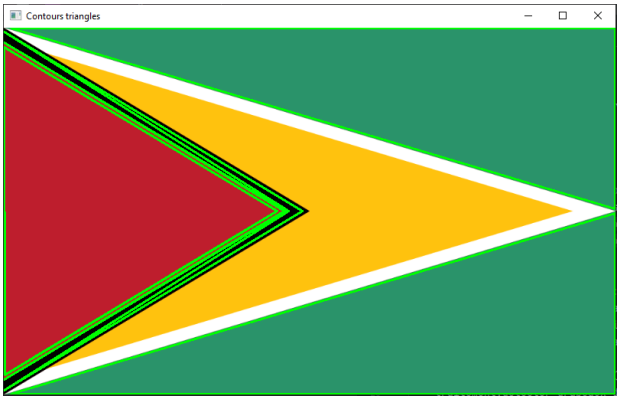
```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Espagne.png
.\images\Espagne.png
Espagne - Valide
```



Drapeau avec Triangles

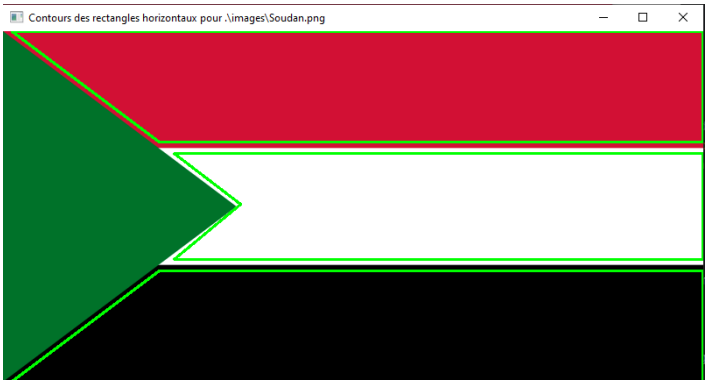
Lorsque nous exécutons le programme avec une image du drapeau de Guyana, le programme identifie bien le pays :

```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Guyana.png
.\images\Guyana.png
Guyana - Valide
```



Lorsque nous exécutons le programme avec une image du drapeau du Soudan, le programme identifie bien le pays :

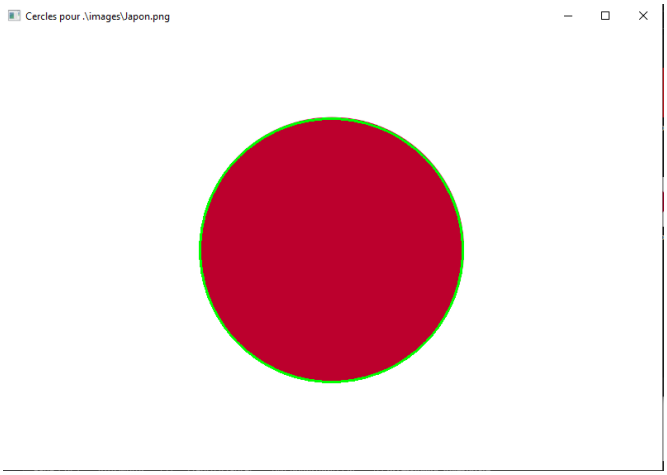
```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Soudan.png
.\images\Soudan.png
Soudan - Valide
```



Drapeau avec Cercles

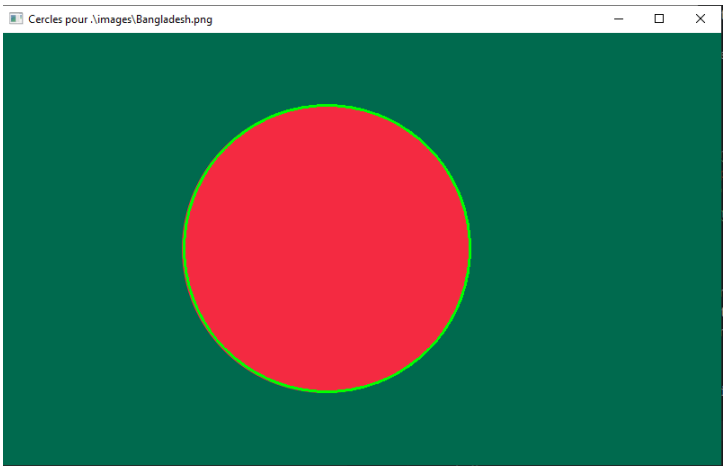
Lorsque nous exécutons le programme avec une image du drapeau du Japon, le programme identifie bien le pays :

```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Japon.png
.\images\Japon.png
Japon - Valide
```



Lorsque nous exécutons le programme avec une image du drapeau du Bangladesh, le programme identifie bien le pays :

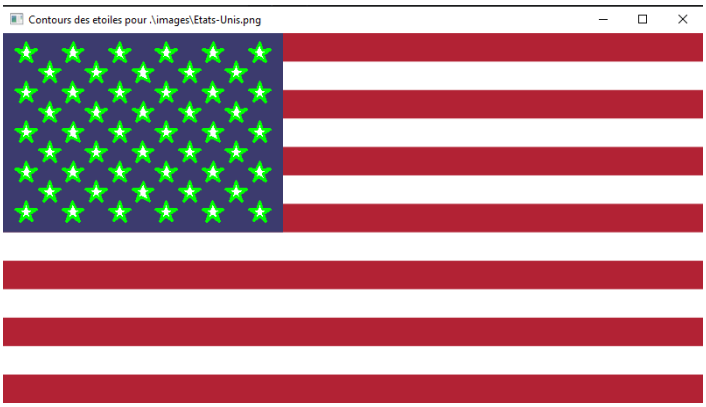
```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Bangladesh.png
.\images\Bangladesh.png
Bangladesh - Valide
```



Drapeau avec Étoiles

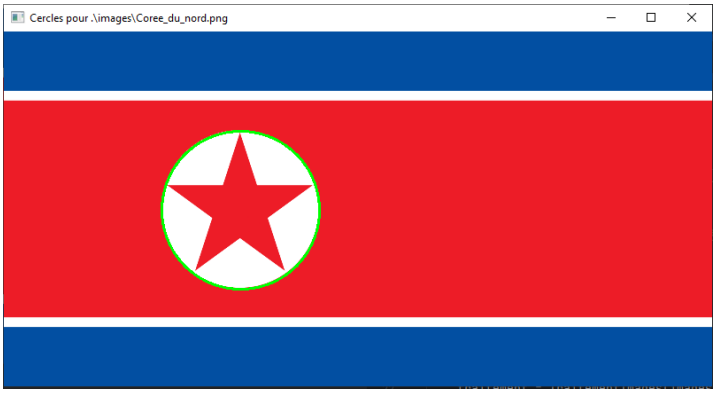
Lorsque nous exécutons le programme avec une image du drapeau des États-Unis, le programme identifie bien le pays :

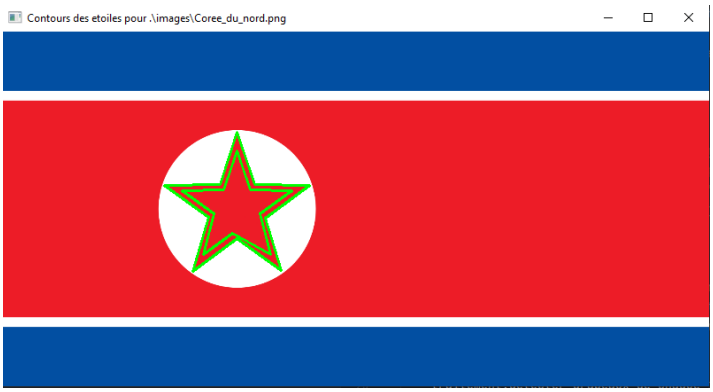
```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Etats-Unis.png
.\images\Etats-Unis.png
États-Unis - Valide
```



Lorsque nous exécutons le programme avec une image du drapeau de la Corée du Nord, le programme identifie bien le pays :

```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Coree_du_nord.png
.\images\Coree_du_nord.png
Coree du Nord - Valide
```

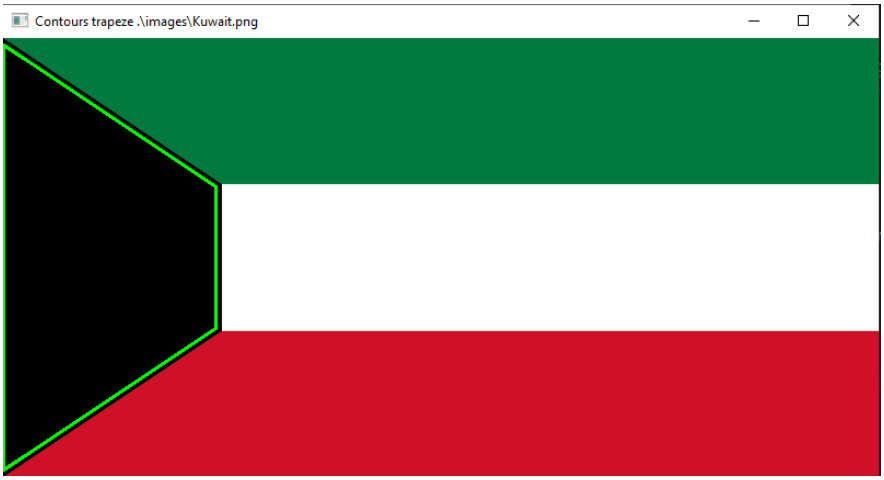


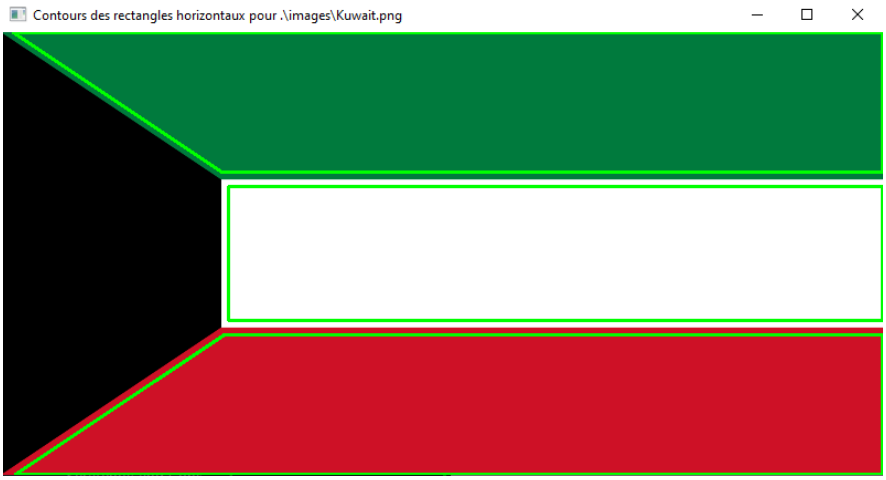


Drapeau avec Trapèze

Lorsque nous exécutons le programme avec une image du drapeau du Kuwait, le programme identifie bien le pays :

```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Kuwait.png
.\images\Kuwait.png
Kuwait - Valide
```





Problèmes/Améliorations

Problème 1 : Détection du Liban

Bien que notre programme identifie bien le drapeau du Liban, la méthode utilisée n'est pas idéale. Tout d'abord, la détection des couleurs est faite de la même façon que tous les autres drapeaux. Le problème vient lors de la détection du cèdre, qui se fait à partir de la surface de l'image de couleur verte. Si la surface minimale est valide, le drapeau est identifié comme le drapeau du Liban.

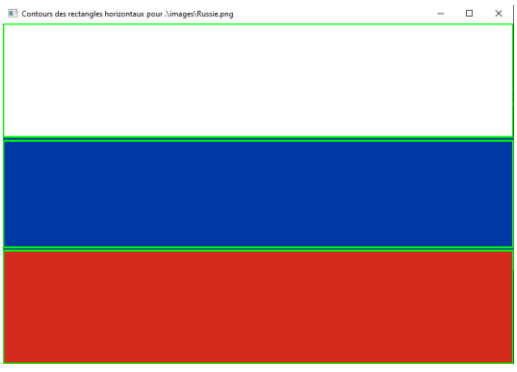
```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Liban.png
.\images\Liban.png
Liban - Valide
```

Comme solution, il serait préférable d'entrainer notre programme avec des images de cèdres, afin que l'on puisse détecter le cèdre à l'aide des données fournies.

Problème 2 : Ordre des couleurs

Dans son état actuel, notre programme ne prend pas en compte l'ordre des couleurs, ce qui veut dire qu'il n'y aura pas de problème à identifier le drapeau de la Russie :

```
PS C:\Users\TegMiles\SIF1033\Traitement-image-SIF1033> py main.py .\images\Russie.png
.\images\Russie.png
Russie - Valide
```



Cependant, cela causera des problèmes lors de la détection du drapeau des Pays-Bas, qui sera aussi identifié comme le drapeau de la Russie, puisqu'il est également composé de 3 rectangles de couleurs blanc, bleu et rouge :

La solution est simplement d'établir un ordre de couleurs dans la banque de critères (par exemple, de gauche à droite et de haut en bas) afin que le programme puisse interpréter l'ordre des couleurs des drapeaux.