Image et Interaction - ArUco

Cyril Barrelet & Marc Hartley

December 2024

Figure 1: Marqueurs ArUco

1 Introduction

Un marqueur ArUco (voir Figure 1) est un genre de QR code facilement détectable dans une image, et à l'instar des QR code, ils sont associés à un identifiant unique.
De part leurs propriétés et leur facilité à être généré, ils sont très utilisés dans des applications de réalité augmenté. En effet, en plus d'être facilement localisable, un repère 3D peut être associé à chacun des marqueurs permettant d'y ancrer, par exemple, un objet 3D.

Exercices d'échauffement :

- Si ce n'est pas déjà fait, créez l'arborescence `IG3/FASE`.
  Conseil : Pensez à créer un dossier pour chacun de vos cours sous la racine `IG3`. L'organisation est la clé de la réussite.
- Créez maintenant `IG3/FASE/Image/TP_Intro` et `IG3/FASE/Image/TP_Background-substraction`, et glissez-y vos travaux précédents dans leur dossier respectif.
- Enfin, créez l'arborescence `IG3/FASE/Image/TP_ArUco`. Nous allons travailler dans cet espace de travail (c-à-d, dans le dossier `TP_Aruco`).

Une fois échauffé, nous allons apprendre à lire le flux vidéo de la webcam installée sur votre moniteur. Pensez à enlever le cache de la webcam ou à la sortir de son emplacement si nécessaire.
Créez un fichier `webcam.py` et copiez le code ci-dessous :

```
import cv2

camera = cv2.VideoCapture(0)
if(not(camera.isOpened())):
        print("Impossible d'ouvrir la webcam.")
        exit()
```

```
while(True):
        # Capture frame-by-frame
        ret, frame = camera.read()

        if(not(ret)):
                print("Impossible de recevoir une nouvelle frame. Quitte...")
                break

        # Add text in the top-left corner of the image
        cv2.putText(frame, "Smile, you're filmed!", (10, 50),
    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

        cv2.imshow('frame', frame)
        if(cv2.waitKey(1) == ord('q')):
                break

# When everything done, release the capture
camera.release()
cv2.destroyAllWindows()
```

Ce code ressemble à s'y méprendre à celui utilisé pour lire une vidéo, n'est-ce pas ?

Exercice : Le troisième argument de la fonction `cv2.putText` correspond à des coordonnées en pixels. Translatez le texte dans l'image en jouant avec ces coordonnées (aidez-vous de l'exemple donné à la fin du TP d'introduction).

2 Détection d'ArUco

Comme le montre la Figure 1, les arucos peuvent avoir différentes tailles et être constitués d'un nombre de carreaux différent. Ce site permet de générer des arucos en jouant sur ces différents paramètres. Remarquez que chaque identifiant est associé à un unique aruco.

Étant donné le nombre d'arucos disponibles, il est nécessaire d'indiquer quel type d'arucos nous allons utiliser ; on parle du dictionnaire d'aruco. Nous allons travailler avec des arucos de dimension 6×6, il faudra donc préciser le bon dictionnaire afin de faciliter le travail du détecteur :

```
# We use the 6X6_1000 aruco format
dictionary = cv2.aruco.getPredefinedDictionary(cv2.aruco.DICT_6X6_1000)
# Default parameters to detect arucos
parameters = cv2.aruco.DetectorParameters()
# Detector instance
detector = cv2.aruco.ArucoDetector(dictionary, parameters)
```

La code suivant permet de détecter les arucos et d'accéder à leur position et identifiant :

```
    # Detect arucos within the frame
    markerCorners, markerIds, rejectedCandidates = detector.detectMarkers(frame)

    if len(markerCorners) > 0:
        ids = markerIds.flatten()
```

```
        for (markerCorner, markerID) in zip(markerCorners, ids):

            corners = markerCorner.reshape((4, 2))
            (topLeft, topRight, bottomRight, bottomLeft) = corners
            corners = corners.astype(int)
```

Exercices :

- Créez un nouveau fichier nommé `aruco_detection.py`.
- Copiez-y le code nécessaire à la lecture d'une webcam.
- Instanciez un détecteur d'aruco au début de votre script.
- Détectez les arucos.
- Dessinez les bords de l'aruco en vous aidant de la documentation.
- Dessinez un cercle positionné sur le centre de l'aruco et dont le périmètre est égal à la plus grande des diagonales (pensez à inclure NumPy pour calculer le centre et la taille des diagonales).
- Affichez l'identifiant de l'aruco détecté sur l'un de ses côtés à l'aide de `cv2.putText`.
- Changez la couleur des lignes et des cercles en fonction de l'identifiant de l'aruco.
- Testez avec deux arucos différents et appelez l'enseignant.

3 Masque d'image

Dans certaines situations, il est parfois nécessaire de masquer certaines parties de l'image. Par exemple, on peut masquer le visage d'une personne pour "garantir" l'anonymité de son identité.
Nous allons ici essayer de garantir l'anonymité de notre aruco en le masquant dans l'image.

Mais déjà, c'est quoi un masque ?

Un masque est une image binaire ou multi-canaux utilisée pour sélectionner ou ignorer certaines parties d'une image. Chaque pixel du masque indique si le pixel correspondant dans l'image d'origine doit être conservé ou ignoré.
Le masque étant binaire (constitué de 0 et/ou de 1), il suffira de multiplier l'image d'origine par le masque pour masquer ou garder les pixels correspondants.

Exercice : Voici la structure de code que vous utiliserez :

```python
import cv2
import numpy as np

# Load all the images into a list
images = [cv2.imread(f"random_images/img{i}.jpg") for i in range(10)]

for img in images:
    height, width, channel = img.shape

    # Create a 2D mask
    mask = ..

    radius = 100
    center = (int(width/2), int(height/2))
```

```
    # Draw an ellipse on the mask
    mask = cv2.ellipse(mask, center, (radius, radius), 0, 0, 360, 1, -1)

    # Apply the mask
    masked_img = ...

    cv2.imshow('Image Masquee', masked_img)
    cv2.waitKey()

cv2.destroyAllWindows()
```

- Téléchargez le dossier `random_images` sur Moodle et glissez le dans votre espace de travail.
- Créez un fichier `masque.py` et copiez-y la structure donnée.
- Créez un masque de la taille de l'image courante.
- Appliquez le masque sur l'image courante.
- Changez les coordonnées du centre de façon aléatoire (`np.random.randint`) et appelez l'enseignant.

4 Mini-jeu : RandomGuessr

Nous aimerions créer un jeu révolutionnaire à l'aide de nos connaissances. L'idée serait d'afficher la webcam, puis de proposer au joueur de déplacer des marqueurs sur l'écran pour laisser apparaître une image aléatoirement choisie. Chaque marqueur se verrait associé à une image, de façon à ce que le joueur puisse découvrir différentes images en fonction du marqueur utilisé.

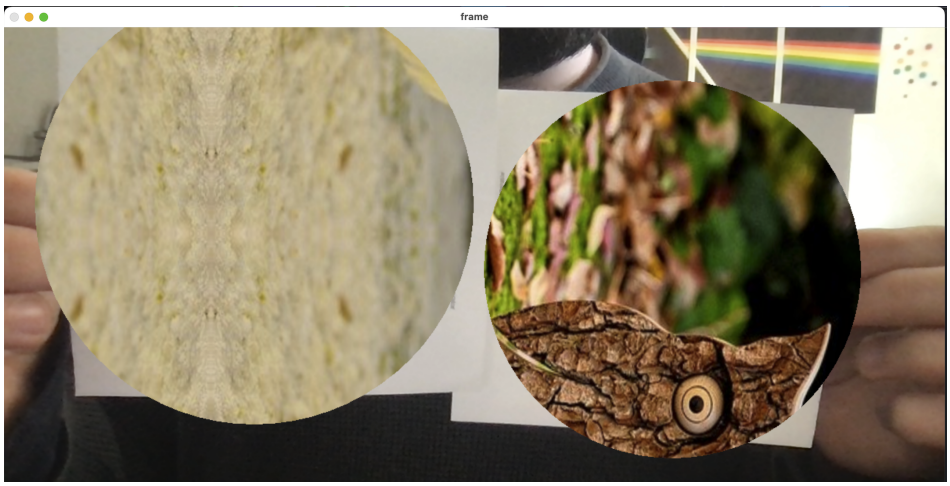

Figure 2: RandomGuessr v0.0.1 alpha preview

TODO :

- Créer un fichier `randomGuessr.py`
- Charger deux images aléatoirement.
- Gérer la webcam de l'utilisateur.
- Adapter la taille des images chargées à la résolution de la webcam.
- Détecter les arucos.
- "Masquer" les arucos.

- "Sélectionner" l'image de fond correspondante à l'identifiant du marqueur.

Bonus TODO :
Le jeu est trop facile, le joueur reconnaît trop facilement les images. Pour pimenter un peu le jeu, nous aimerions ajouter une rotation aléatoire à l'image de fond.

```
# Petit bout de code magique
def rotateImage(image, angle):
    h, w, c = image.shape
    center = h/2, w/2
    rot_mat = cv2.getRotationMatrix2D(center, 180 * (math.pi/2 - angle) / math.pi,1.0)
    return cv2.warpAffine(image, rot_mat, (w, h),flags=cv2.INTER_LINEAR,
    borderMode=cv2.BORDER_REFLECT)
```