Nom :.....**Prénom :**.....

Matricule :.....**Groupe :**.....

Exercice 1 : *(10 pts +0.5 : réponse correcte ou* ***-0.25 réponse incorrecte****):*

Indiquez **la réponse correcte** parmi les propositions :

1. Le but principal de la vision par ordinateur est :
A. Améliorer la qualité des images
B. Permettre à la machine d'interpréter les images
C. Compresser les images
D. Stocker les images

2. L'inconvénient principal du filtre moyenneur est :
A. Il est lent
B. Il supprime les contours
C. Il augmente le bruit
D. Il change les couleurs

3. Un filtre de Sobel sert à :
A. Lisser l'image
B. Détecter les contours
C. Réduire le bruit
D. Compresser l'image

4. Un filtre passe-bas sert principalement à :
A. Détecter les contours
B. Réduire le bruit
C. Augmenter la résolution
D. Extraire des points clés

5. Le rôle d'un gradient dans une image est de :
A. Décrire la couleur
B. Mesurer les variations d'intensité
C. Compresser l'image
D. Supprimer le bruit

6. La segmentation d'image consiste à :
A. Lisser l'image
B. Découper l'image en régions
C. Compresser l'image
D. Détecter les coins

7. La matrice intrinsèque contient :
A. La rotation
B. La translation
C. La focale et le centre optique
D. La profondeur

8. Les paramètres extrinsèques décrivent :
A. La taille de l'image
B. La position et l'orientation de la caméra
C. La distorsion
D. Le capteur

9. Le calibrage utilise généralement :
A. Une image aléatoire
B. Un motif connu
C. Une image floue
D. Une image binaire

10. La disparité correspond à :
A. Une différence de couleur
B. Une différence de position d'un point
C. Une erreur de calibrage
D. Une profondeur

11. Plus la disparité est grande, l'objet est :
A. Plus loin
B. Plus proche
C. Plus flou
D. Invisible

12. NeRF signifie :
A. Neural Rendering Function
B. Neural Radiance Fields
C. Network Ray Function
D. Neural Ray Filter

13. NeRF est basé sur :
A. Des règles géométriques
B. Un réseau de neurones
C. Un filtre linéaire
D. Une transformée de Fourier

14. L'entrée principale d'un NeRF est :
A. Une image dont la dimension est connue
B. Une coordonnée spatiale et une direction
C. Une carte de profondeur
D. Une disparité

15. La sortie d'un NeRF est :
A. Une image binaire
B. Une couleur et une densité
C. Une profondeur
D. Une texture

16. NeRF utilise le principe de :
A. Projection orthographique
B. Ray marching
C. Seuillage
D. Segmentation

17. Un filtre en vision par ordinateur sert principalement à :
A. Modifier la géométrie de l'image
B. Transformer l'image pour extraire ou améliorer l'information
C. Compresser l'image
D. Changer le format du fichier

18. Le rendu dans NeRF consiste à :
A. Additionner les couleurs le long d'un rayon
B. Détecter les contours
C. Corriger la distorsion
D. Segmenter la scène

19. Le principal inconvénient de NeRF est :
A. La faible qualité
B. Le temps de calcul élevé
C. Le manque de précision
D. Le besoin d'une stéréo

20. Un filtre passe-haut permet :
A. De lisser l'image
B. De supprimer les contours
C. De mettre en évidence les détails
D. De réduire la résolution

Exercice 2 *(4 pts)* **:**

On considère une caméra calibrée définie par :

Matrice intrinsèque $K = \begin{pmatrix} 800 & 0 & 320 \\ 0 & 800 & 240 \\ 0 & 0 & 1 \end{pmatrix}$, $R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $t = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

1. Expliquez en détail le rôle de chaque paramètre de la matrice. (0.75)
   Fx=800 : focale de la caméra exprimée **en pixels selon l'axe x**.
   fy=800 : focale de la caméra exprimée **en pixels selon l'axe y**.
   cx=320 : coordonnée **x en pixel** du centre de l'image.
   cy=240 : coordonnée **y en pixel** du centre de l'image.

2. Précisez le rôle de *R* et *t*. (0.75)
   R est la matrice de rotation (extrinsèque) qui décrit **l'orientation de la caméra par rapport au repère monde**.
   t est le vecteur de translation qui **décrit la position de l'origine du repère caméra dans le repère monde**.

   Dans cette exemple le repère caméra est **confondu** avec le repère monde (pas de rotation, pas de translation

Soit un point 3D exprimé dans le repère monde $P_w = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$

3. Déterminer la matrice de projection : $P = [R|t]$. (0. 5)

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Calculer les coordonnées $\boldsymbol{P_c}$ du point $\boldsymbol{P_w}$ dans le repère caméra (1pt).

$$P_c = PP_w = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 5 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$$

5. Calculer les coordonnées image $\boldsymbol{P_{im}}$ du point $\boldsymbol{P_c}$ en pixel. (1pt)

$$P_{im\_hom} = \begin{pmatrix} 800 & 0 & 320 \\ 0 & 800 & 240 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix} = \begin{pmatrix} 2400 \\ 2800 \\ 5 \end{pmatrix} \qquad P_{im\_hom} = \begin{pmatrix} 2400/5 \\ 2800/5 \\ 5/5 \end{pmatrix} = \begin{pmatrix} 480 \\ 650 \\ 1 \end{pmatrix}$$

Ou

$$P_{im\_hom} = \begin{pmatrix} 800 & 0 & 320 \\ 0 & 800 & 240 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1/5 \\ 2/5 \\ 5/5 \end{pmatrix} = \begin{pmatrix} 480 \\ 650 \\ 1 \end{pmatrix}$$

$$==> P_{im} = (u, v) = (480, 650) \quad \boldsymbol{en\ pixels}$$

Exercice 3 *(6 pts)* **:**

I. Le programme suivant calcule une image en niveaux de gris en utilisant la moyenne des trois composantes de couleur, mais l'image n'est pas affichée correctement. Identifiez chaque erreur et proposez une correction possible :(2pts)

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread("usthb.jpg",cv2.IMREAD_GRAYSCALE)
5  h,w,c  = img.shape
6
7  B = img[:,:,0]
8  G = img[:,:,1]
9  R = img[:,:,2]
10
11 imgGray = (B + G +R)/3
12
13 cv2.imshow("image2",imgGray)
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
```

1) img = cv2.imread("usthb.jpg", cv2.IMREAD_COLOR) (0.5pt)

2) La somme des composantes B+G+R peut dépasser la valeur maximale représentable sur 8 bits (255), ce qui provoque un débordement. De plus, cette opération génère une image de type flottant, qui doit être entre 0 et 1. Plusieurs solutions ont été étudiées en TP, par exemple : (1,5 pts)

- (B.astype(np.float32) + G.astype(np.float32) + R.astype(np.float32))/(3*255)
- ((B.astype(np.float32) + G.astype(np.float32) + R.astype(np.float32))/3) astype(np.uint8)
- 0.114 * B + 0.587 * G + 0.299 * R
- …..

II. On veut détecter un **objet bleu** dans une image en utilisant **la couleur**. Le programme suivant **fonctionne partiellement**, mais la détection n'est **pas robuste**.

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread("usthb.jpg")
5
6  lower_blue = np.array([150, 0, 0])
7  upper_blue = np.array([255, 100, 100])
8
9  mask = cv2.inRange(img, lower_blue, upper_blue)
10
11 contours, _ = cv2.findContours(
12     mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
13 )
14
15 for cnt in contours:
16     if cv2.contourArea(cnt) > 500:  # ignorer petits objets
17         x, y, w, h = cv2.boundingRect(cnt)
18         cv2.rectangle(img, (x, y), (x+w, y+h), (0,255,0), 2)
19
20 cv2.imshow("Objet vert detecte", img)
21 cv2.waitKey(0)
22 cv2.destroyAllWindows()
```

1. Que représente la variable mask ? (0.25)
   Le mask est une image binaire qui contient uniquement les pixels dont la couleur est comprise dans l'intervalle du bleu défini en BGR.
   Les pixels blancs correspondent aux zones bleues détectées, les noirs au reste de l'image.

2. Quel est l'espace de couleur utilisé ? Est-il approprié ? Pourquoi ? (0.5)
   L'espace de couleur utilisé est BGR.
   Il n'est pas approprié, car : la couleur et la luminosité ne sont pas séparées, un simple changement de lumière peut fausser la détection.

   L'espace HSV est généralement plus adapté pour la détection par couleur.

3. Quel est le problème principal de ce programme ? pourquoi ? (0.5)
   Le problème principal de cette technique est qu'elle est très sensible à l'éclairage : les valeurs des pixels changent lorsque la lumière varie, ce qui peut empêcher de détecter correctement les objets bleus, surtout dans le cas où la détection se fait directement sur les valeurs BGR d'un pixel.

4. Que se passe-t-il si l'arrière-plan contient une couleur bleue ? (0.25)
   Les zones bleues de l'arrière-plan seront également détectées, ce qui provoque la détection des objets non désirés.

5. Citer deux améliorations (techniques) simples pouvant améliorer l'image ou le résultat (autres que les problèmes déjà mentionnés dans les questions précédentes). (0.5)

   Appliquer un flou gaussien ou moyenneur pour réduire le bruit avant la détection.

   Utiliser des opérations morphologiques (ouverture/fermeture) sur l'image binaire pour éliminer les petits artefacts.

Donner le programme amélioré qui réponds aux questions précédentes (2 pts) :

Le programme complet a déjà été traité lors des séances de TP (mais sans boucle et dépend des améliorations proposées par l'étudiant):

```python
import cv2
import numpy as np

lo=np.array([95, 80, 60])
hi=np.array([115, 255, 255])
def detect_inrange(image, surfaceMin,surfaceMax):
    points=[]
    image=cv2.blur(image, (5, 5))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    mask=cv2.inRange(image, lo, hi)
    mask=cv2.morphologyEx(mask,cv2.MORPH_OPEN, None, iterations=2)
    elements=cv2.findContours(
        mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
    elements=sorted(elements, key=lambda x:cv2.contourArea(x), reverse=True)
    for element in elements:
        if cv2.contourArea(element)>surfaceMin and cv2.contourArea(
            element)<surfaceMax:
            ((x, y), rayon)=cv2.minEnclosingCircle(element)
            points.append(np.array([int(x), int(y),int(rayon),int(
                cv2.contourArea(element))]))
    return image, mask, points
VideoCap=cv2.VideoCapture(0)

import time
while(True):
    start = time.time()
    ret, frame=VideoCap.read()
    cv2.flip(frame,1,frame)
    image,mask,points = detect_inrange(frame,1000,3000)
    cv2.circle(frame, (100, 100), 20, (0, 255, 0), 5)
    print(image[100,100])
    if (len(points)>0):
        cv2.circle(frame, (points[0][0], points[0][1]), points[0][2], (0, 0, 255), 2)
        cv2.putText(frame,str(points[0][3]),(points[0][0], points[0][1]),
        cv2.FONT_HERSHEY_COMPLEX,1,(255,0,0),2,cv2.LINE_AA)
    if mask is not None :
        cv2.imshow("mask",mask)
    cv2.imshow('image', frame)
    if cv2.waitKey(10)&0xFF==ord('q'):
        break
    t = time.time()-start
    fps = 1/t
    print("fps :",fps)
VideoCap.release()
cv2.destroyAllWindows()
```