TP5 - Création de panoramiques

QUESTION 1 : PRENEZ CONNAISSANCE DU SQUELETTE FOURNI.

Nous changeons la code:

```
Python
def feature_detector(type, gray, nb):
    if gray is not None :
        match type :
            case "GFTT":
                # TODO
                print("not implemented yet")
                sys.exit(1)
            case "ORB":
                orb = cv.ORB_create(nb)
                kp = orb.detect(gray, None)
            case _:
                sift = cv.SIFT_create(nb)
                kp=sift.detect(gray, None)
    else:
        kp =    None
    return kp
```

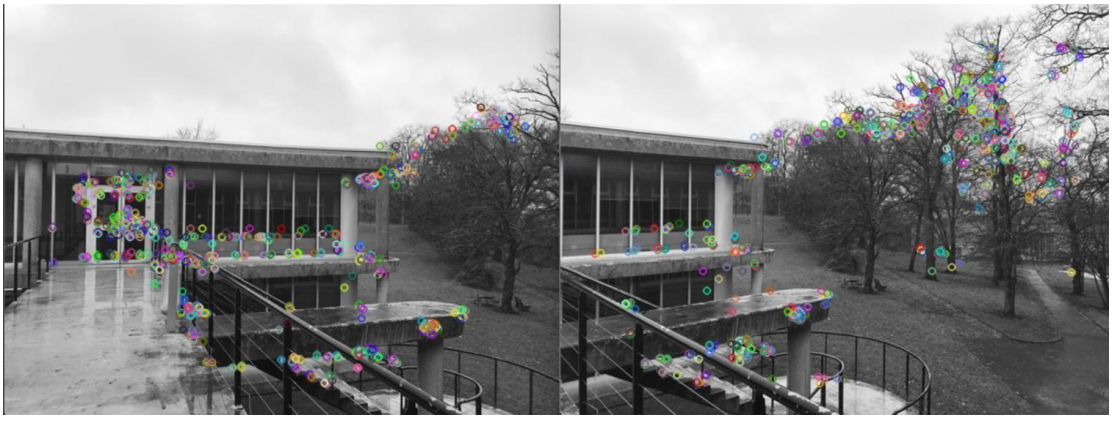
Donc Nous pouvons choisi le détector. Nous avons constaté que les inspections d'eipd étaient réparties sur l'ensemble de l'image, tandis que le détecteur ORB détectait plus de détails.

En SIFT (n=250):

n=750:

En ORB (n=250):

n=750:



QUESTION 2 : DESCRIPTION DE POINTS D'INTÉRÊT- *feature detector*

```
Python
def feature_extractor(type, img, kp):
        desc = None
        if type == "SIFT":
```

```
        sift = cv.SIFT_create()
        kp, desc = sift.compute(img, kp)
    elif type == "ORB":
        orb = cv.ORB_create()
        kp, desc = orb.compute(img, kp)
    elif type == "GFTT":
        # GFTT does not have a built-in descriptor, you might want to use another
method or create a custom one
        # Here, we just return the keypoints without descriptors
        pass
    return desc
```

QUESTION 3 : FAIRE LA MISE EN CORRESPONDANCE

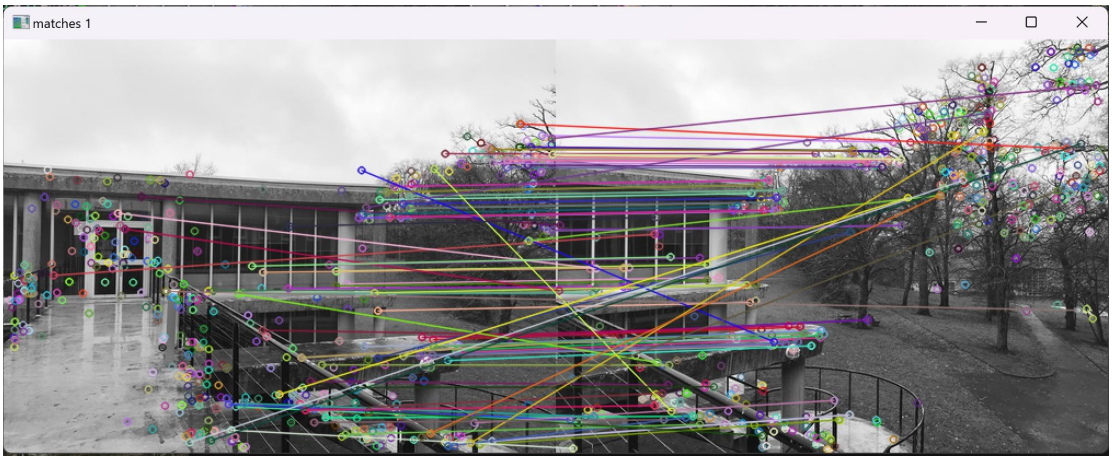
Nous écrivons le code pour faire correspondre. Dans les matchs de première catégorie, seuls deux points sont les plus proches voisins l'un de l'autre. Cette approche est trop simple. Nous essayons à nouveau la méthode KNN pour faire correspondre. Il établit une liste contenant le voisin le plus proche et le deuxième voisin le plus proche. Si la distance du voisin le plus proche est inférieure à 0,75 fois la distance du voisin le plus proche, c'est une bonne correspondance, ajoutez-la à la liste des bons.

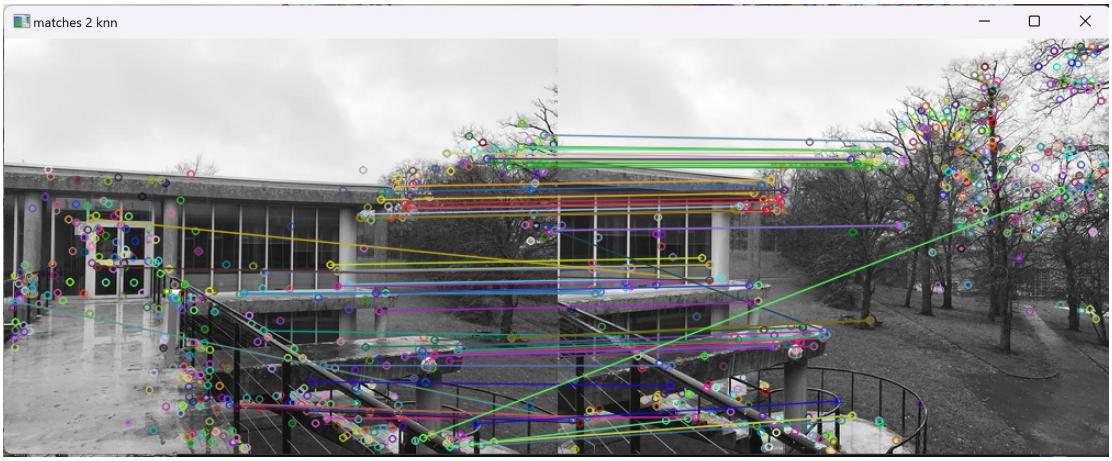
```
Python
    # match
    bf = cv.BFMatcher(cv.NORM_L2, crossCheck=True)
    # Match descriptors.
    matches = bf.match(desc1,desc2)
    # Sort them in the order of their distance.
    matches = sorted(matches, key = lambda x:x.distance)
    # Draw first 10 matches.
    res =
cv.drawMatches(img1,kp1,img2,kp2,matches[:100],None,flags=cv.DrawMatchesFlags_N
OT_DRAW_SINGLE_POINTS)
    display_image(res, "matches 1")

    # KNN match
    bf = cv.BFMatcher()
    matches = bf.knnMatch(desc1,desc2, k=2)
    good = []
    for m,n in matches:
        if m.distance < 0.75*n.distance:
            good.append([m])
    res = cv.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
    display_image(res, "matches 2 knn")
```

QUESTION 4 : AFFICHER LE RÉSULTAT DE LA MISE EN CORRESPONDANCE

Nous pouvons constater que la première méthode produit plus de faux matchs que la deuxième. KNN est donc la meilleure méthode d'appariement.





QUESTION 5 : CALCULER À PARTIR DES POINTS EN CORRESPONDANCE L'HOMOGRAPHIE PERMETTANT DE TRANSFORMER LA 2E IMAGE DANS LE POINT DE VUE DE LA 1RE.

```
Python
def calculate_homography(kp1, kp2, matches):
    dst_pts = np.float32([kp1[m[0].queryIdx].pt for m in matches]).reshape(-1, 1, 2)
    src_pts = np.float32([kp2[m[0].trainIdx].pt for m in matches]).reshape(-1, 1, 2)

    H, _ = cv.findHomography(src_pts, dst_pts, cv.RANSAC)
    return H
```

[[5.89090392e-01 2.33377731e-02 1.82369328e+02]
 [-1.53679175e-01 8.61422815e-01 3.16062418e+01]
 [-8.27409856e-04 3.41271971e-07 1.00000000e+00]]

QUESTION 6 : COLLER LES DEUX IMAGES ENSEMBLE POUR FAIRE UN PANORAMIQUE.

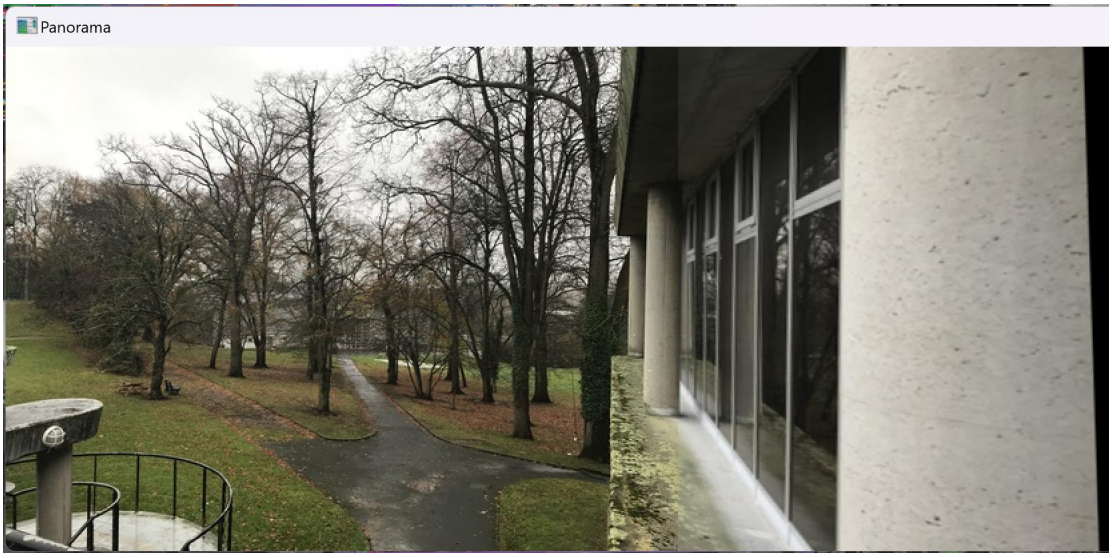
Les épissures fonctionnent bien.

```
Python
def warp_and_stitch(img1, img2, H):
    result = cv.warpPerspective(img2, H, (img2.shape[1] + img1.shape[1], img2.shape[0]))
    result[0:img1.shape[0], 0:img1.shape[1]] = img1

    return result
```

QUESTION 7 : CHANGEMENT D'IMAGES ?

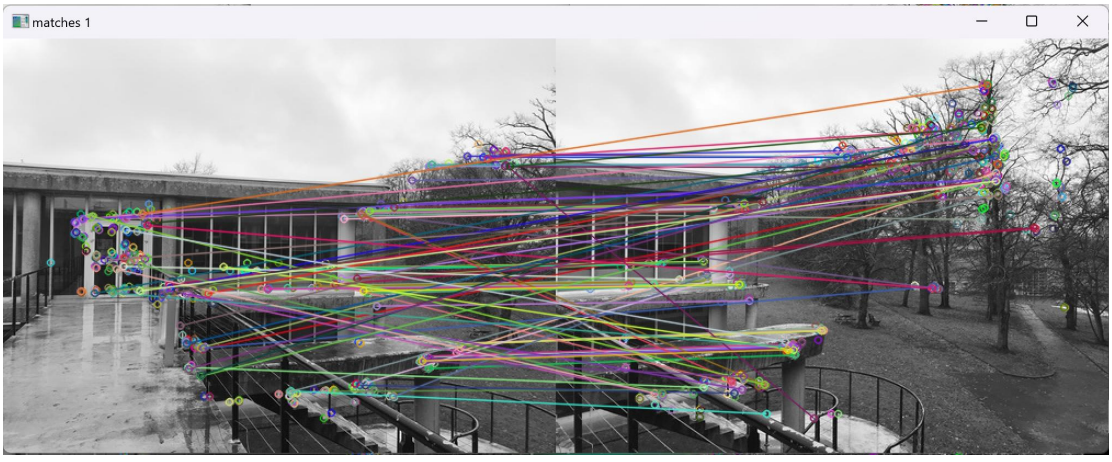
Après avoir changé l'image, l'effet de patchwork est encore très bon. La logique du code est très raisonnable.

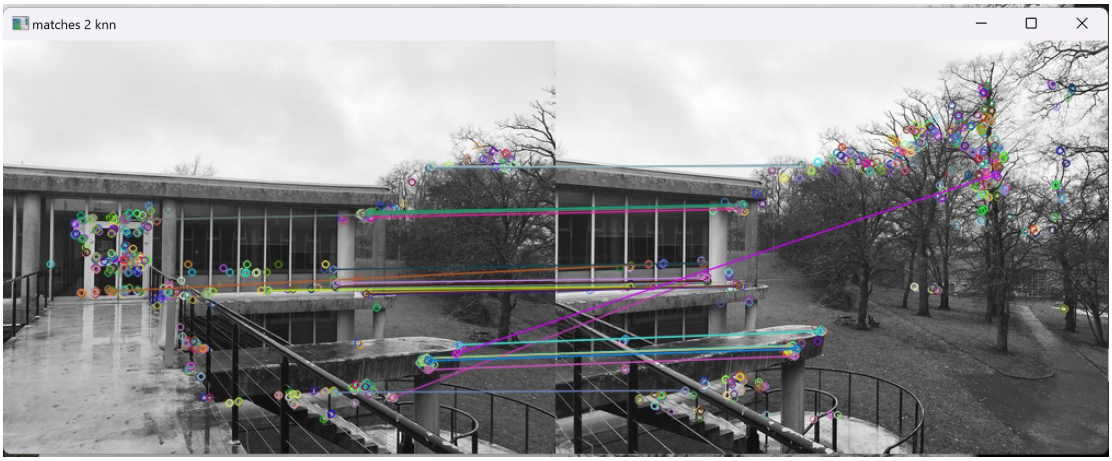


QUESTION 8 : CHANGEMENT DE DÉTECTEUR ET DESCRIPTEUR.

L'utilisation d'ORB pour la reconnaissance des points d'intérêt est un peu moins efficace que celle de SIFT. La méthode de correspondance simple est beaucoup moins bonne que la méthode KNN.

ORB






Panorama