Poročilo - Vaja 4

Pri tej vaji smo implementirali algoritem za nalaganje in augmentacijo zbirke slik koc ter njihovih pripadajočuh značilk.

Algoritem izvaja premike, rotacije ter skaliranje v skladu z navodili naloge. Poskrbljeno je tudi za deljivost s 16 ter črne robove ki nastanejo pri augmentaciji, z metodo vrisanega kvadrata.

```
image = image[int(width/2-p/2):int(width/2+p/2),int(height/2-p/2):int(height/2+p/2)]
for d in dice:
    if d[2] < int(width/2-p/2) or d[2] > int(width/2+p/2) or d[3] < int(height/2-p/2) or d[3] > int(height/2+p/2):
        d[2] = 0
        d[3] = 0
    else:
        d[2] -= int(width/2-p/2)
        d[3] -= int(height/2-p/2)


for d in dice:
    if d[2] != 0 and d[3] != 0:
        image = cv2.circle(image, (d[3],d[2]), int(16*scale), (0,0,255), 2)

cv2.imshow("img",image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Augmentacija značilk je izvedena z nekaj ročnimi operacijami nad koordinatami točk, ter tudi z zrcaljenjem operacije nad sliko na polje zančilk.

```
image_center = (int(height/2), int(width/2))
rot_mat = cv2.getRotationMatrix2D(image_center, random.uniform(0.0,360.0), 1.0)
image = cv2.warpAffine(image, rot_mat, image.shape[1::-1], flags=cv2.INTER_LINEAR)

for d in dice:
    tmp_arr = np.zeros((width,height), dtype=np.uint8)
    tmp_arr[d[2],d[3]] = 255
    tmp_arr = cv2.warpAffine(tmp_arr, rot_mat, tmp_arr.shape[1::-1])

    if np.sum(tmp_arr) == 0:
        d[2] = 0
        d[3] = 0
    else:
        d[2] = np.where(tmp_arr != 0)[0][0]
        d[3] = np.where(tmp_arr != 0)[1][0]
```

Najzanimivejša težava ki se je pojavila je bila pri rotaciji in skaliranju slike, saj obe operaciji potrebujeta interpolacijo zaradi spreminjanja dimenzij in orientacije

matrike. Zaradi tega je interpolacija pokvarila matriko značilk, kar pa smo rešili z uporabo kubične interpolacije in spremembe načina označevanja značilk v matriki.

Rezultati: