Vaja 4: Računalniški vid v robotiki

Vsako od vaj zagovarjate pri asistentu. Za uspešen zagovor morate rešiti vse naloge, razen tistih, ki so označene z ★. Če katera od nalog zahteva skiciranje ali ročno računanje, morate to pokazati na zagovoru. Programerske naloge morate rešiti samostojno in izvorno kodo ter rezultate predstaviti na zagovoru. Če kode, ki rešuje vaše naloge, ne boste poznali v detajle, lahko asistent vaš zagovor zavrne. V primeru zavrnitve zagovora, ga lahko ponovno opravljate z zamudo, a bo zato vaše končno število točk kaznovano s faktorjem **0.75**.

Za rešen obvezen del nalog lahko dobite največ **75** točk, dodatne naloge pa vam lahko prinesejo do **25** dodatnih točk. Za pomanjkljive rešitve ali slabo teoretično znanje vam asistent lahko odšteje točke.

> Pogoste napake in nasveti
> - Zamenjava x in y osi
> - Napačen podatkovni tip: `float`, `uint8`
> - Napačno definicijsko območje podatkov: [0,255], [0,1]
> - Rešitev razvijajte na enostavnih primerih (lahko so sintetični)

Ta naloga povezuje robotiko in računalniško zaznavanje. Naš sistem z robotsko roko bomo nadgradili s kamero, ki opazuje delovno površino robotskega manipulatorja. Na delovni površini boste detektirali objekte glede na barvo in obliko. Z uporabo homografije boste točke na delovni površini preslikali v koordinatni sistem robota, kar vam bo omogočilo odzivanje na trenutno stanje delovne površine.

1. Homografska transformacija

Homografska transformacija (oz. homografija) je transformacija med ravninami v dveh dimenzijah. Uporabimo jo lahko za preslikavo planarnih (t.j. takih, ki ležijo na isti fizični ravnini) regij v slikah v druge koordinatne sisteme. V našem primeru s kamero opazujemo ravno delovno površino robota. To površino hočemo najprej definirati v sliki, jo nato opisati s pravokotno mrežo in končno preslikati v koordinatni sistem robota. Za to bomo uporabili homografijo.

Točke, ki nam določajo ravnino naše delovne površine so definirane z oznakami April (angl. AprilTag). Zunanja oglišča oznak nam definirajo štiri točke na ravnini, ki jih lahko uporabimo za izračun 3×3 homografske matrike.

Naj bo točka $\boldsymbol{p}_c = [x_c, y_c, 1]^T$ točka v sliki, točka $\boldsymbol{p}_w = [x_w, y_w, 1]^T$ pa točka v koordinatnem sistemu naše delovne površine (velikost tega koordinatnega sistema navadno definiramo v pikslih). Matrika

$$\boldsymbol{H}_w^c = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix},$$

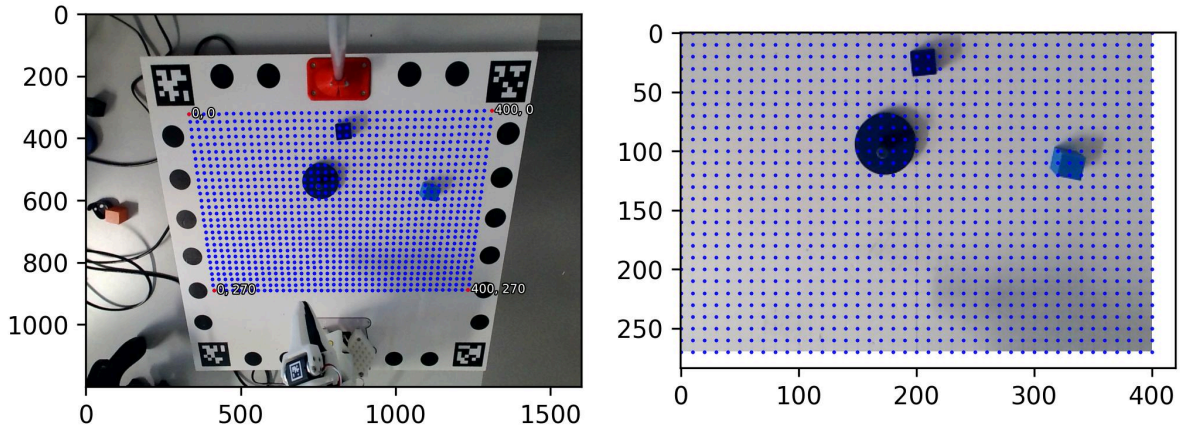
ki predstavlja našo preslikavo preko enačbe

$$\boldsymbol{x}_w = \boldsymbol{H}_w^c x_c$$

preslika našo točko v sliki v koordinatni sistem delovne površine. Ker je koordinatni sistem kamere usmerjen desno in navzdol, je najenostavneje, če izhodišče delovne površine postavimo v levi zgornji vogal, kot je prikazano na Sliki 1. Za določitev vseh vrednosti matrike $\boldsymbol{H}_w^c$ moramo uporabiti vsaj štiri točke (vrednost elementa h_{33} je vedno 1, zato ima homografija 8 prostih parametrov). Preslikavo v nasprotno smer lahko dosežemo z uporabo matrike $\boldsymbol{H}_c^w = (\boldsymbol{H}_w^c)^{-1}$.

a) Ročno določite štiri vogalne točke na eni od slik v priloženem materialu, in štiri točke, ki bodo predstavljale vašo delovno površino (za ta primer vogale izberite na vogalih mreže na sredini delovne površine). Izračunajte homografijo z uporabo funkcije `cv2.findHomography()` in ustvarite novo sliko, ki vsebuje samo vašo delovno površino. Bodite pozorni na vrstni red točk v obeh koordinatnih sistemih, sicer boste dobili neustrezne matrike H. Homografsko matriko lahko uporabite s funkcijo `cv2.warpPerspective()`. Na vaši delovni površini v preslikani sliki določite vse vogale celic. Mreža je visoka 27 celic in široka 40 celic (celice so velike 1cm). Velikost ciljnega koordinatnega sistema lahko določite v metričnih koordinatah, izraženih v milimetrih.

b) Vogale, ki ste jih določili v prejšnji sliki, preslikajte nazaj v začetno sliko. Preslikavo iz delovne površine nazaj v sliko lahko dobite z inverzom matrike $\boldsymbol{H}$ (uporabite funckijo `np.linalg.inv()`). Vaše vhodne točke morajo biti v homogenih koordinatah, rezultat pa morate zato pretvoriti nazaj v kartezične koordinate tako, da ga delite z zadnjo vrednostjo. Izrišite vse vogale celic v obeh slikah.



Slika 1: Prikaz homografije za preslikavo delovne površine robota.

V okviru sledi še nekaj o detekciji oznak AprilTag. To ni obvezen del naloge in služi kot zanimivost.

Oznake AprilTag lahko v opencv detektiramo z naslednjo kodo

```
# naložite sliko
aruco_dict =
cv2.aruco.getPredefinedDictionary(cv2.aruco.DICT_APRILTAG_36h11)
arucoParams = cv2.aruco.DetectorParameters()

(corners, ids, rejected) = cv2.aruco.detectMarkers(im, aruco_dict,
parameters=arucoParams)
cv2.aruco.drawDetectedMarkers(im, corners, ids)
```

Iz detektorja dobimo pozicije oglišč oznake, polega tega pa tudi njeno unikatno identifikacijsko številko, s katero lahko ločimo oznake z različnimi vzorci. Če poznamo tudi kalibracijsko matriko kamere, lahko za posamezno oznako izračunamo tudi 3d pozicijo:

```
tag_size = 0.04 # velikost oznake v metrih
# M in D sta kalibracijska matrika in vektor koeficientov distorzije
for i in range(len(corners)):
  rvec, tvec, markerPoints =
cv2.aruco.estimatePoseSingleMarkers([corners[i]], tag_size, M,
D)
  cv2.drawFrameAxes(im, M, D, rvec, tvec, 0.01)
```

2. Detekcija objektov

V tej nalogi boste implementirali preprosto detekcijo planarnih (ploskih) objektov z uporabo barvne segmentacije, morfoloških operacij ter opisa regij. Lahko uporabite kodo iz prve vaje.

a) Naložite eno od slik iz priloženega materiala. Na njej uporabite funkcijo `get_workspace_corners()` iz datoteke `workspace_utils.py`, da dobite zunanje vogale oznak delovne površine. Te vogale lahko vstavite v funkcijo `calculate_homography_mapping()`, ki vam bo vrnila dve homografski matriki. Prva izvede preslikavo med izvorno sliko in delovno površino, definirano z oznakami April (velikost celotne delovne površine določimo kot 1000x1000px). Druga matrika pa preslika vogale delovne površine v koordinatni sistem robota. Izrišite zaznane vogale oznak v originalni sliki.

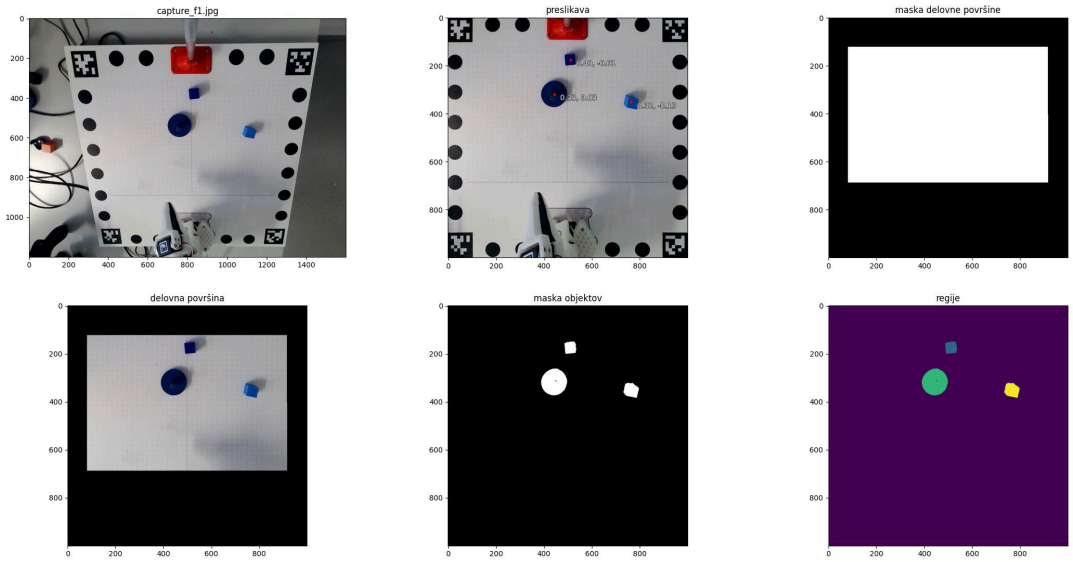
b) Preslikajte izhodiščno sliko v sliko delovne površine s funkcijo `cv2.warpPerspective()` in ustrezno homografsko matriko. Nato ročno določite vogale mreže na poravnani sliki. Z uporabo barvnega prostora HSV boste nato določili regije objektov na delovni površini. Določite ustrezne vrednosti pragov za posamezen kanal in ustvarite masko za mrežo na delovni površini. Uporabite funkcijo `cv2.connectedComponentsWithStats()` in prikažite centroide zaznanih objektov. Maske lahko po potrebi tudi popravite z ustreznimi morfološkimi operacijami. Slika 2 prikazuje vmesne korake postopka.

c) Z uporabo priloženih kamer zajemite dodatne slike, z drugimi predmeti različnih barv in oblik. Za zajem lahko uporabite kodo spodaj. Roke so opremljene z dvema tipoma kamer: roke f1, f4 in f5 uporabljajo kamere s širšim zornim kotom, roke f2, f3 in f7 pa kamere z ožjim zornim kotom. Pred uporabo slik boste morali odstraniti radialno distorzijo. Intrinzične parametre kamere (kalibracijska matrika M in vektor parametrov distorzije D) lahko naložite iz datotek `calibration_data.npz` in `wide_calibration_data.npz` in jih uporabite s funkcijo `cv2.undistort()`. Sliko morate zatem še zavrteti za 180°. Resolucijo slike nastavite na 1600x1200.

```
w, h = 1600, 1200

cap = cv2.VideoCapture(2)
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'))
cap.set(cv2.CAP_PROP_FRAME_WIDTH, w)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, h)

while True:
  _, frame = cap.read()
  cv2.imshow('image', frame)
  cv2.imwrite(f'capture.jpg', frame)
```

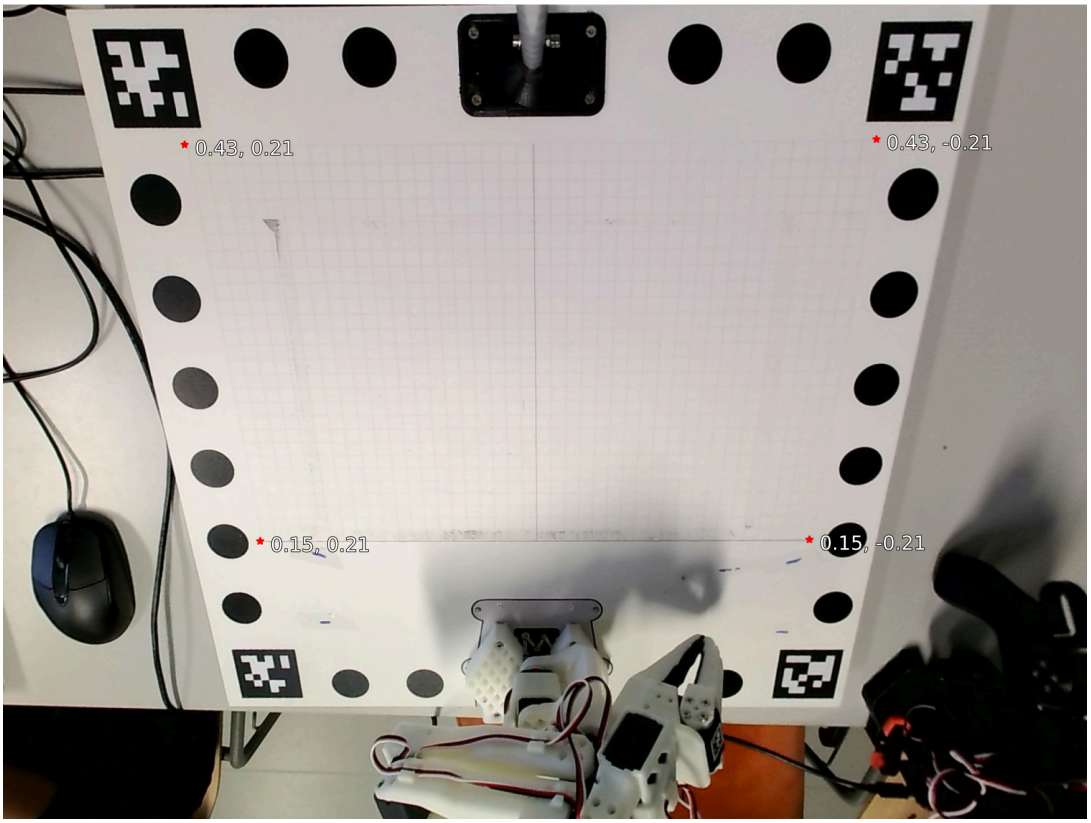


Slika 2: Detekcija modrih objektov na delovni površini.

3. Krmiljenje robotskega manipulatorja

Za preslikavo koordinat na delovni površini v fizične enote, ki jih uporablja robotski manipulator, potrebujemo še eno homografsko matriko. Ta preslika delovno površino v koordinatni sistem, ki je centriran na bazi robotske roke in izražen v metrih. Kot omenjeno v prejšnji podnalogi, lahko ustrezno homografsko matriko pridobite s funkcijo `calculate_homography_mapping()`.

a) Določite nekaj točk na delovni površini in za njih izračunajte koordinate v koordinatnem sistemu robotske roke. Prikažite sliko delovne površine, označite izbrane točke in poleg njih izpišite izračunane koordinate (uporabite funkcijo `plt.text()`). Točke lahko izberete v originalni sliki ali v premapirani sliki delovne površine. V enem primeru boste morali uporabiti obe homografski matriki, v drugem pa samo eno. Če uporabite obe, ju lahko vnaprej zmnožite med sabo. Primer je prikazan na Sliki 3.



Slika 3: Vogali mreže, izraženi v koordinatnem sistemu robota.

b) Sedaj lahko konico robotske roke usmerite tako, da bo kazala na točko na delovni površini. Izberite primerno višino (vsaj 5 cm nad delovno površino), določite točko in izvedite premik roke tako, da se bo nahajala nad izbrano točko. Ocenite oz. izmerite natančnost preslikave.

c) Napišite funkcijo, ki bo zajela sliko s kamere, preslikala delovno površino in detektirala objekt (npr. barvno kocko), ter usmerila robotsko roko, da bo kazala na pozicijo objekta.

★ d) (10 točk) Napišite funkcijo, ki bo krmilila robotsko roko tako, da bo sledila izbranemu objektu. Predpostavite, da je predmet vedno blizu delovne površine. Če objekta v določeni sliki ne zaznate zanesljivo, lahko sledenje začasno prekinete. Objekt lahko definirate z masko na podlagi barve, če želite, pa lahko uporabite tudi kaj naprednejšega (YOLO ali VitTrack).

★ e) (15 točk) Napišite funkcijo, ki bo zaznala vse kocke dveh barv, nato pa izvedla nadzor robotske roke tako, da bo ustrezno prijela in razporedila kocke glede na barvo na levo in desno stran delovne površine.

★ f) (15 točk) Napišite funkcijo, ki zazna barvne kocke na površini in jih zloži v stolp (vsaj štiri). Višino vsake naslednje kocke boste morali nastaviti "na slepo", ker nimate informacije o višini objektov na površini. Kocke lahko vnaprej poravnate z mrežo na delovni površini, lahko pa njihovo rotacijo tudi ocenite iz slike.