Določanje poze objekta z algoritmom POSIT

Poročilo prejektne naloge

Robotski vid 2018/19, Fakulteta za elektrotehniko, Univerza v Ljubljani

Marsel Rus

***Povzetek*—Predstavljen je algoritem POSIT in izvedba v programskem jeziku Python. Algoritem se uporablja za določevanje poze objekta v prostoru, potrebuje pa le eno sliko objekta in njegov model, za razliko od nekaterih ostalih izvedb, kjer se primerjajo korespondenčne točke na objektu. Gre za reševanje sistema linearnih enačb, postopek pa je je močno poenostavljen, kar prispeva k hitrosti algoritma. Izvedba v Pythonu ni optimizirana, je grob prevod podanega programa v članku [1]. Izvedbi programa v Pythonu podamo nekaj oslonilnih točk ter sliko objekta, na njej pa ročno izberemo točke, ki se ujemajo z oslonilnimi. Algoritem vrne matriko rotacij ter vektor translacij objekta v prostoru.**

I. Uvod

Računanje pozicije in orientacije objekta v prostoru ima več pomembnih aplikacij - kalibracija, kartografija, sledenje objektu, prepoznavanje objektov... V okviru projektne naloge je bilo potrebno opraviti izračun lege objekta v prostoru s pomočjo algoritma POSIT (*Pose from Ortography and Scaling with Iterations*). Algoritem izračuna rotacije okoli posameznih osi, ter translacijo objekta od koordinatnega izhodišča. Posebnost algoritma je, da za določevanje poze objekta potrebujemo zgolj eno sliko. V članku je podana izvedba algoritma v programu Mathematica, za namene te naloge pa je bilo potrebno osnovno izvedbo prevesti v programski jezik Pyhton.

II. Metodologija

Za začetek je bilo potrebno pridobiti model objekta. To je bilo storjeno s slikanjem stranice objekta s fotoaparatom. Slika je bila skalirana na širino 1000 slikovnih elementov, nato pa so bile določene koordinate oglišč s pomočjo programskega jezika Python. Iz koordinat na sliki je bila določena začetna lega objekta.

$$\mathbf{M}_{obj}=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 325 \\ 325 & 0 & 325 \\ 0 & 280 & 0 \\ 0 & 280 & 325 \\ 325 & 280 & 325 \end{bmatrix}$$

Zapis zajema šest koordinat, ki predstavljajo sprednja oglišča kvadra. Le-ta so vidna v večini orientacij.

Nato je bilo zajetih več slik istega objekta, tokrat pod različnimi koti. Vse slike so bile nato skalirane na širino 1000 slikovnih elementov. V tej obliki so se nato uporabile za izračun lege objekta.

Za izbor slike iz datoteke je bila uporabljena knjižnica **Tkinter**. S pomočjo prikaznega okna odpremo želeno sliko. Na sliki nato označimo točke oglišč, ki jih poznamo. Določevanje točk je bilo opravljeno s pomočjo funkcije **Canvas**, prav tako iz knjižnice**Tkinter**.

Koordinate označenih točk na sliki program zapiše v obliki **.csv**. V nadaljevanju program izpiše koordinate iz datoteke, kjer so podane koordinate točk objekta in datoteke, kjer so podane koordinate oslonilnih točk v obliki niza (*array*). Nato se izvede zanka, ki opravi izračun lege objekta v prostoru.

III. Programiranje

POSIT je skupek dveh algoritmov. Prvi algoritem je osnovni POS (*Pose from Ortography and Scaling*), ki določi perspektivno projekcijo s skalirano ortografsko projekcijo in z razrešitvijo linearnega sistema enačb poišče rotacijsko matriko in translacijski vektor objekta. Drugi algoritem je sam POSIT - kar predstavlja iteracijo prvega algoritma, z namenom pridobitve boljših rezultatov lege objekta v prostoru.

V navodilih je bil podan predlog uporabe postopka SIFT (*ang. Scale Invariant Feature Transform*) v kombinaciji s postopkom RANSAC (*RANdom SAmple Concesus*), ki bi služila kot povezovanje točk na sliki s točkami na modelu. Zaradi časovnih omejitev sem metodo prilagodil. Podane so 3D koordinate nekaj točk na objektu. Za programiranje je bilo uporabljeno okolje Visual Studio 2019.

Koordinate so izbrane tako, da so vidne v večini orientacij objekta. Ob zagonu nam program daje možnost izbire slike, na kateri je opazovani objekt. Na sliki nato označimo toliko točk, kot jih je podanih v datoteki, kjer so le-te definirane. Paziti je potrebno tudi na vrstni red, ki je od leve proti desni, od zgoraj navzdol. Objekt, na katerem se je preverjalo delovanje algoritma pa je z namenom preprost - kvader, ki ima dolžino vseh štirih stranic enako. Ko so na kvadru določene vse točke, se izvajanje programa nadaljuje v algoritem POSIT.

Algoritem je bil preveden iz Mathematice v Python. Delovanje se je sproti pregledovalo, končni rezultat glede na podane vrednosti je v obeh primerih enak. Algoritem vrne matriko rotacij in vektor translacij. Vrstice v rotacijski matriki predstavljajo enotske vektorje **i**,**j** in **k** in je oblike:

$$\mathbf{R}=\begin{bmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{bmatrix}$$

Matrika rotacij služi transformaciji koordinat objekta v kamerin koordinatni sistem. Za izračun rotacije potrebujemo določiti le **i** in **j** v objektnem koordinatnem sistemu. Enotski vektor **k** nato definiramo kot vektorski produkt med **i** in **j**. Algoritem kot vhodne vrednosti vzame koordinate objekta iz 3D (podane vrednosti), matriko koordinat točk na sliki (točke, ki jih poklikamo na sliki), psevdoinverz matrike koordinat objekta ter goriščno razdaljo v pikslih.

V članku je bila podana kocka, znane pa so bile koordinate oglišč. Za preverjanje delovanja algoritma so bile podane še koordinate oglišč na sliki iste kocke, rotirane v prostoru. Ob izdelavi programa v jeziku Python so bile vzete iste točke, tako začetne poze kot po rotaciji.

Ko je algoritem deloval, je bil izbran objekt preproste oblike. Objekt je bil slikan s fotoaparatom. Za delovanje algoritma je bila pomembna tudi goriščna razdalja leče fotoaparata, ki se je glede na velikost senzorja na fotoaparatu preračunala v slikovne elemente:

$$f_{px} = (\frac{f_{mm}}{w_{zaznavalo}} w_{px}) \tag{1}$$

Pri tem je f_{px} goriščna razdalja v slikovnih elementih, f_{mm} goriščna razdalja v milimetrih, $w_{zaznavalo}$ in w_{px} pa sta širina

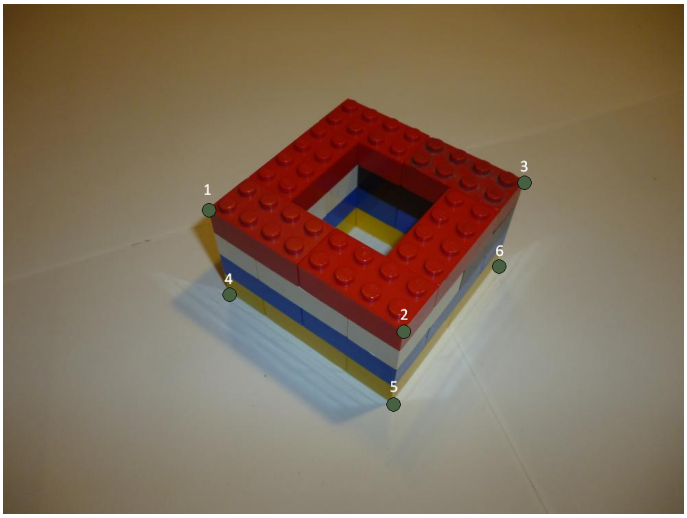
zaznavala v fotoaparatu ter širina izhodne slike v slikovnih elementih.

Podrobnosti delovanja algoritma in izhodišča za računanje rotacijske matrike ter vektorja translacije so na daljši način razložena v članku [1].

IV. Komentar rezultatov

Kot je bilo že omenjeno v predhodnih poglavjih, so bili rezultati programa preverjeni sproti, glede na rezultate izvedbe v Mathematici. Algoritem ne potrebuje zahtevnih računskih operacij. Za rotacijo je potrebno izračunati le enotske vektorje **i** in **j**, za vektor translacije pa le z koordinato Z_0.

Ob zagonu program vpraša, katero sliko želimo analizirati. Izberemo sliko, ki smo jo predhodno pravilno skalirali. Na sliki nato izberemo točke, ki jih imamo podane v modelu. Za primer izračuna so bila to oglišča kvadra, vrstni red izbire lokacije oglišč pa je prikazan na spodnji sliki.



Algoritem v vseh preverjanjih skonvergira po nekaj iteracijah. Spodaj je prikazan izhod algoritma, ki vrne rotacijsko matriko ter vektor translacij. Sproti se izpisuje še napaka med točkami modela in točkami na sliki v posamezni iteraciji.

```
545.0
849.0
246.0
298.0
92.0
131.0
41.0
56.0
17.0
25.0
8.0
11.0
2.0
2.0
0.0
[[ 0.46176976 -0.54005848  0.70363736]
 [-0.68016827  0.28971879  0.67337519]
 [-0.56751894 -0.7895361  -0.23354726]] [ 693.60643432  358.57460573 1477.76201151]
Press any key to continue . . .
```

Za analizo delovanja algoritma je bilo izvedenih več izračunov pozicije objekta, pri različnih orientacijah.

Glede na to, da točke na sliki določamo ročno, je prisotna napaka zaradi nenatančnosti določevanja oglišč objekta. Poleg tega kamera ni bila kalibrirana, zaradi česar se pojavi manjša napaka kot posledica distorzije slike.

V. Povzetek

Opravljena je bila izvedba algoritma POSIT v programskem jeziku Python, v okolju Visual Studio 2019. Za delovanje programa so bile uporabljene knjižnice **Numpy**, **PIL**, **Tkinter**, **Csv** in druge. Za sprotno preverjanje rezultatov je bila uporabljena še knjižnica **Matplotlib**. Opravljeno je bilo zajemanje objekta s fotoaparatom Panasonic DMC-S1, obdelava slik na enako dimenzijo, zapis koordinat v tekstovne datoteke in izračun lege objekta v prostoru. Lega objekta je popisana z rotacijsko matriko dimenzij 3x3 ter vektorjem translacij dimenzije 1x3.

Literatura

[1] Dementhon, D., Davis, L., “Model-Based Object Pose in 25 lines of Code,"*International Journal of Computer Vision*, pp. 123-141, 1995.
[2] Stack Overflow, https://stackoverflow.com 2019
[3] doc. dr. Žiga Špiclin, "Predavanja pri predmetu robotski vid", *Fakulteta za elektrotehniko*, 2019