Pretvorba slike u ptičju perspektivu pomoću genetičkog algoritma

Miholek, Branka

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:283007

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-06-27**

Repository / Repozitorij:

Repository of School of Applied Mathematics and Computer Science

Sveučilište Josipa Jurja Strossmayera u Osijeku

Odjel za matematiku

Sveučilišni diplomski studij matematike
smjer: Matematika i računarstvo

Pretvorba slike u ptičju perspektivu pomoću genetičkog algoritma

Diplomski rad

Osijek, 2022

Sveučilište Josipa Jurja Strossmayera u Osijeku

Odjel za matematiku

Sveučilišni diplomski studij matematike
smjer: Matematika i računarstvo

Pretvorba slike u ptičju perspektivu pomoću genetičkog algoritma

Diplomski rad

Mentor:

izv. prof. dr. sc. Domagoj Matijević

Komentor:

dr. sc. Mateja Đumić

Ana Petrinec, mag. phys. (Orqa)

Kandidat:

Branka Miholek

Osijek, 2022

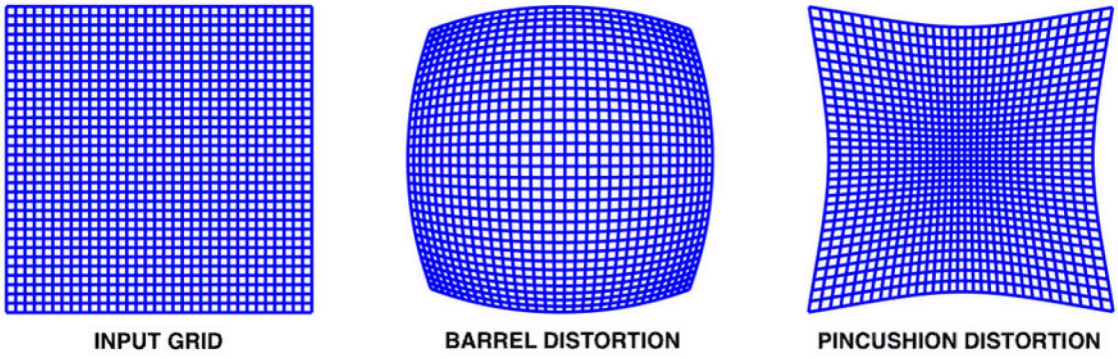
Sadržaj

1 | Uvod

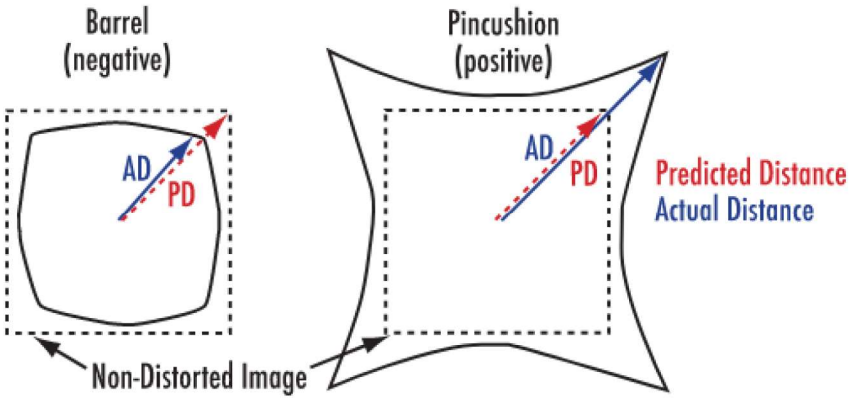
Pretvorba slike u ptičju perspektivu proces je kojim se dobiva povišeni pogled iz oštrog kuta na objekte. Promatrač tada ima dojam da je ptica u letu koja gleda u objekte s velike visine. Ptičja perspektiva koristi se u video igricama, fotografiji, slikarstvu, filmskoj industriji, automobilskoj industriji, ..., i dio je područja računalni vid.
Već u početnim danima računala došlo se do zaključka da su računala odlična u rješavanju zadataka koji su ljudima teški, ali često nisu u zadacima koje ljudi odrađuju lagano, poput prepoznavanja lica [7]. Računalni vid bavi se povezivanjem računala i slika, a cilj je da se računalo koristi za razne zadatke u kojima je potrebno razumjeti i analizirati sliku kako bi se iz nje moglo nešto doznati ili nešto s njom napraviti. Pretvorba u ptičju perspektivu dio je računalnog vida jer se koristeći razne podatke na slici određuje njena transformacija u ptičju perspektivu.
Ovaj rad kreće s poglavljem o radijalnoj distorziji jer je korekcija radijalne distorzije obično prvi korak pri radu sa slikama koje će se prebacivati u ptičju perspektivu. U drugom poglavlju je opisana matrica homografije s teorijske strane, a zatim i postupak njene primjene na sliku. Treće poglavlje daje pregled heuristika i metaheuristika s naglaskom na evolucijske algoritme odnosno genetički algoritam koji je korišten za optimizaciju homografije koja prebacuje slike u ptičju perspektivu. Posljednje poglavlje opisuje način kako je genetički algoritam korišten, parametre koji su ispitani i izabrani u algoritmu, te na kraju rezultate koji su dobiveni.
Projekt ovog diplomskog rada odrađen je u suradnji s Orqa d.o.o. Orqa je IT tvrtka, osnovana 2018. godine, koja se bavi istraživanjem i razvojem tehnologija iz područja FPV-a (*First Person View*), RR-a (*Remote Reality*), proizvodnjom raznih elektroničkih komponenata i uređaja, i dr. Najpoznatiji Orqin proizvod su videonaočale za pilote FPV dronova.

2 | Radijalna distorzija

Distorzije su optička odstupanja koja nastaju pri dizajniranju leće zbog ogiba (difrakcije), loma (refrakcije) i valne prirode svijetlosti[10]. Distorzije spadaju u monokromatska odstupanja i mogu se prepoznati po linijama koje na slici izgledaju deformirano ili zaobljeno, a zapravo su ravne. Za razliku od nekih drugih optičkih odstupanja, distorzije sadržaj slike mijenjaju geometrijski pa zato distorziju možemo izračunati i maknuti sa slike [11], a isto tako ju i dodati slici. Iako distorzije utječu na sadržaj slike, tj. na točke i linije slike, ne utječu na kvalitetu slike. U okruženjima s vrlo jakom distorzijom, neke informacije sa slike ipak mogu biti izgubljene, npr. previše sadržaja bude mapirano u jedan piksel [11].
Najčešći tipovi distorzije su radijalna i tangentna. Kod tangentne distorzije slika je nagnuta u stranu pa su udaljenosti objekata do kamere na slici krive, tj. objekti koji su jednako udaljeni izgledaju kao da nisu.
Kod radijalne distorzije linije koje trebaju biti ravne su zaobljene. Radijalna distorzija dijeli se na 2 tipa - *barrel* i *pincushion*, koji se mogu vidjeti na slici 2.1. Kod *barrel* ili negativne distorzije točke u vidnom polju čine se preblizu centru slike, dok se kod *pincushion* ili pozitivne distorzije točke čine previše udaljene od centra slike. Njihova razlika bolje se može vidjeti na slici 2.2. Kod negativne distorzije je predviđena udaljenost točke do centra veća nego stvarna udaljenost, tj. točka je na slici bliže centru. Kod pozitivne distorzije je suprotno, tj. predviđena udaljenost točke do centra je manja nego stvarna pa je točka udaljenija od centra.
Distorzija je veća što je žarišna duljina kraća, odnosno FOV (*Field of View*), tj. vidno polje kamere, veće. Kod kamera koje se koriste za ptičju perspektivu, FOV je uglavnom veći pa je prisutna velika radijalna distorzija. Npr. prikaz okoline automobila u ptičjoj perspektivi željeli bismo napraviti koristeći što manje kamera. Te kamere zato moraju imati veći FOV jer trebaju obuhvatiti cijeli prostor oko automobila.



Slika 2.1: Tipovi radijalne distorzije. Slika preuzeta iz [12]



Slika 2.2: Razlika između negativne i pozitivne radijalne distorzije s obzirom na stvarnu udaljenost i predviđenu udaljenost. Slika preuzeta iz [11]

2.1 Ispravljanje radijalne distorzije

Uklanjanje radijalne distorzije započinje određivanjem potrebnih parametara kamere, a to su ekstrinzični parametri (vektor rotacije i vektor translacije kamere), intrinzična matrica te koeficijenti distorzije. Pomoću ekstrinzičnih i intrinzičnih parametara određuje se mapiranje trodimenzionalne točke iz stvarnog svijeta scene u dvodimenzionalnu točku na slici.
Mapiranje kreće iz koordinatnog sustava stvarnog svijeta u koordinatni sustav kamere. U oba koordinatna sustava točke su trodimenzionalne. Ako je na sceni neka od točaka određena kao ishodište, svaku drugu točku u sceni možemo definirati pomoću udaljenosti do ishodišta s obzirom na osi x, y i z. Tako za neku proizvoljnu točku T dobivamo koordinate (X_w, Y_w, Z_w). Zatim je potrebno pronaći vezu s kamerom. Za to koristimo ekstrinzične parametre. Lokaciju kamere označavamo s vektorom translacije $\mathbf{t} = [t_x, t_y, t_z]^T$. Kamera na toj lokaciji može biti orijentirana na različite načine, a njenu orijentaciju označavamo s matricom rotacije $\mathbf{R} \in \mathbb{R}^{3\times3}$. Matrica $\mathbf{R}$ može se dobiti iz kuteva s obzirom na svaku od tri osi. Tada je točka T

definirana s koordinatama (X_c, Y_c, Z_c) u koordinatnom sustavu kamere :

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t}.$$

Zatim je potrebno tu točku mapirati na sliku u dvodimenzionalnom obliku. Za to se koristi intrinzična matrica **K**.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

gdje su f_x i f_y žarišne duljine izražene u pikselima, a (c_x, c_y) je optički centar. Točku (u, v) u koordinatnom sustavu slike možemo dobiti iz točke (X_c, Y_c, Z_c) na sljedeći način:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = K \cdot \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Zatim se u i v dobiju prebacivanjem točke (u', v', w') u nehomogene koordinate (homogene koordinate objašnjene su u poglavlju 3) :

$$u = \frac{u'}{w'}$$

$$v = \frac{v'}{w'}.$$

Za računanje ovih parametara koristi se uzorak šahovske ploče, kao na slici 5.1, i postupak objašnjen u [9]. Da bi se ovaj postupak mogao koristiti, potrebno je uslikati uzorak s barem dvije različite orijentacije. Može se koristiti bilo koji planarni uzorak [9], ali se najčešće koristi uzorak šahovske ploče jer je lagano uočljiv na slici te se, zbog kontrasta crne i bijele boje polja, lakše detektiraju točke kuteva. OpenCV ima funkciju *calibrateCamera*() koja tim postupkom računa **R**, **t**, **K** i koeficijente distorzije koristeći detektirane točke uzorka. Potrebna su nam dva koeficijenta distorzije - k_1 i k_2. Ako s (x, y) označimo pravu (nedistortiranu) točku, a s (x', y') distortiranu točku, onda vrijedi ([9]):

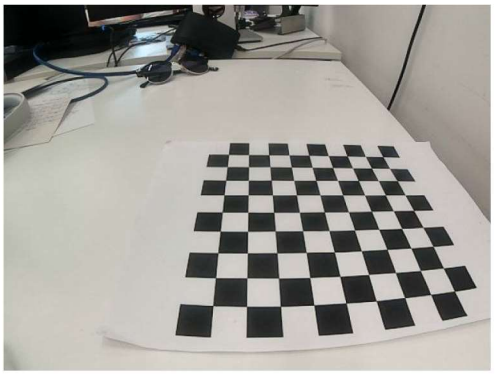
$$x' = x + x \cdot (k_1(x^2 + y^2) + k_2(x^2 + y^2)^2)$$

$$y' = y + y \cdot (k_1(x^2 + y^2) + k_2(x^2 + y^2)^2).$$

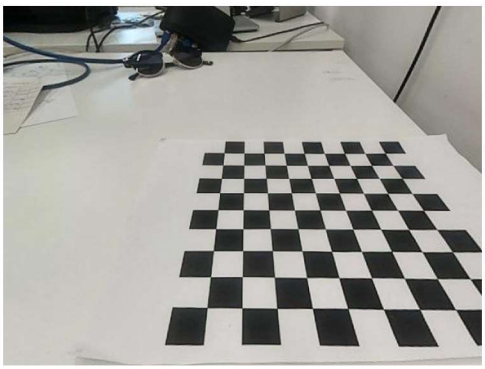
Nakon što se ove vrijednosti izračunaju, može se koristiti funkcija iz OpenCV-a *initUndistortRectifyMap*(). Ona računa polja map_x i map_y s koordinatama za mapiranje. Formule koje ova funkcija koristi mogu se naći na stranici službene dokumentacije OpenCV-a. Polja map_x i map_y se zatim mogu proslijediti funkciji *remap*() koja mapiranje radi preko formule :

$$dst(x, y) = src(map_x(x, y), map_y(x, y)),$$

gdje je *dst* nova slika, a *src* originalna. Ovakvo mapiranje je inverzno *forward* mapiranju kod kojega se mapira iz *src* u *dst*. Primjer korištenja *forward* mapiranja može se vidjeti u poglavlju 3.3.2. Na taj način dobivamo novu, nedistortiranu sliku koja je iste veličine kao i originalna slika.

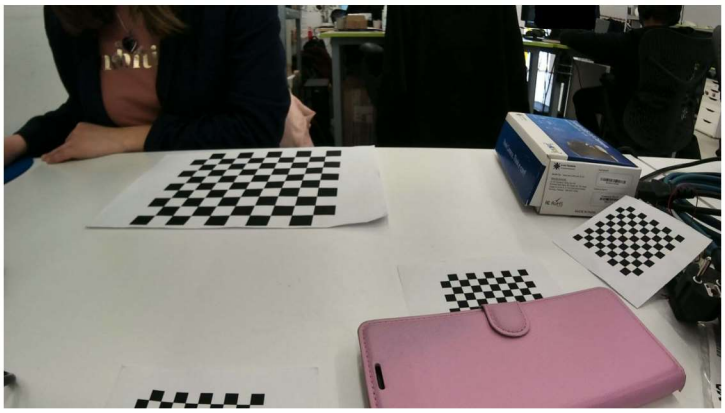


(a) Slika s negativnom radijalnom distorzijom.

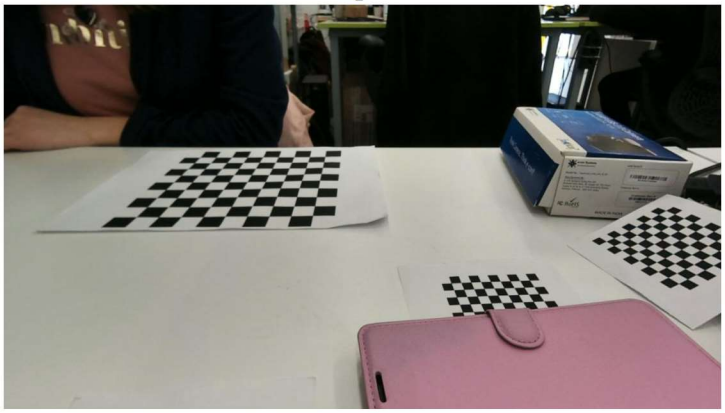


(b) Slika nakon uklanjanja radijalne distorzije.

Slika 2.3: Prvi primjer uklanjanja radijalne distorzije.



(a) Slika s negativnom radijalnom distorzijom. Distorzija se može vidjeti, osim na uzorku, na rubu stola pri vrhu slike te na mobitelu pri dnu slike.



(b) Slika nakon uklanjanja radijalne distorzije.

Slika 2.4: Drugi primjer uklanjanja radijalne distorzije.

3 | Homografija

3.1 Projektivna geometrija i homogene koordinate

Projektivna geometrija je disciplina geometrije koja proučava projektivna svojstva, odnosno svojstva geometrijskih tijela koja ostaju sačuvana pri svakoj projektivnoj transformaciji [13]. Euklidska geometrija dobra je za proučavanje svojstava općeg fizičkog prostora. No, kod snimanja i obrađivanja slika Euklidska geometrija nije dovoljna jer u slikama duljine, kutovi i paralelizam nisu očuvani [8]. Iz tog razloga se u računalnom vidu koristi projektivna geometrija. Euklidska geometrija zapravo je podskup projektivne geometrije [8].
Ako dimenziju prostora označimo s n, prostor kojeg projektivna geometrija proučava označava se s $\mathbb{P}^n$. Npr. projektivna ravnina $\mathbb{P}^2$ analogna je dvodimenzionalnoj Euklidskoj ravnini $\mathbb{R}^2$.
Sustav koordinata koji se koristi u projektivnoj geometriji naziva se homogenim koordinatama. Sve točke iz $\mathbb{P}^2$ su u homogenom koordinatnom sustavu i imaju dimenziju 3.

3.2 Prikaz točaka u homogenim koordinatama

Neka je $\bar{x} = [\mathrm{x}, \mathrm{y}]^T \in \mathbb{R}^2$ točka u Euklidskom prostoru. Pravac koji prolazi kroz $\bar{x}$ može se zapisati kao :

$$p_1 \quad ... \quad ax + by + c = 0.$$

Ako ga pomnožimo s nekim skalarom $\lambda \neq 0$ dobiva se sljedeći pravac :

$$p_2 \quad ... \quad \lambda ax + \lambda by + \lambda c = 0.$$

Očito su pravci p_1 i p_2 jednaki.
Sada možemo označiti $\mathbf{x} = [\lambda \mathrm{x}\ , \lambda \mathrm{y}, \lambda]^T$ i pravac prikazati skalarnim produktom :

$$\mathbf{x}^T(a, b, c)^T = (a, b, c)\mathbf{x} = 0.$$

$\mathbf{x} \in \mathbb{P}^2$ je 3-vektor koji predstavlja homogene koordinate točke koja u nehomogenim koordinatama izgleda kao $[\frac{x}{\lambda}, \frac{y}{\lambda}]^T$. Dakle, točku $\bar{x}$ možemo zapisati u homogenim koordinatama kao $[\mathrm{x},\mathrm{y},1]^T \in \mathbb{P}^2$.
Kao što je vidljivo u primjeru, u projektivnoj geometriji nema razlike u reprezentaciji točke i pravca što se zove princip dualnosti [8].

Definicija 1. *Projektivna transformacija je invertibilno mapiranje* $h : \mathbb{P}^2 \to \mathbb{P}^2$ *takvo da su 3 točke* x_1, x_2 *i* x_3 *smještene na nekom pravcu p ako i samo ako su* $h(x_1)$, $h(x_2)$ *i* $h(x_3)$ *isto smještene na pravcu p.*

Projektivna transformacija još se naziva i homografija. Iz definicije 1 slijedi da ovakva transformacija ravne linije na slici održava ravnima.
Inverz projektivne transformacije je isto jedna projektivna transformacija, kao i kompozicija dvije projektivne transformacije.

Teorem 1 (vidjeti [4, *Theorem* 2.10])**.** *Mapiranje* $h : \mathbb{P}^2 \to \mathbb{P}^2$ *je projektivna transformacija ako i samo ako postoji regularna matrica* $H \in R^{3\times 3}$ *takva da za svaku točku* $\boldsymbol{x} \in \mathbb{P}^2$ *vrijedi* $h(\boldsymbol{x}) = H\boldsymbol{x}$.

Teorem 1 osigurava da svaka projektivna transformacija vrati rezultat u obliku homogenih koordinata.
Mapiranje točke $\mathbf{x} \in \mathbb{P}^2$, $\mathbf{x} = [\mathrm{x}, \mathrm{y}, 1]^T$ je :

$$\mathbf{x}' = H \cdot \mathbf{x}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Matrica homografije H može se pomnožiti proizvoljnim nenul skalarom bez da se projektivna transformacija promijeni [4].

3.3 Izračun homografije

Kada se računa homografija koja će jednu sliku prilagoditi drugoj, koriste se točke koje se podudaraju na obje slike, tj. zajedničke točke. Ako su nam dana dva skupa podudarajućih točaka, jedan za svaku od slika, potrebno je odrediti projektivnu transformaciju koja će svaku točku iz prvog skupa odvesti na mjesto pripadajuće točke iz drugog skupa. Svaku sliku promatramo kao projektivnu ravninu. U ovome problemu bitno je pitanje koliko podudarajućih točaka je potrebno da bi se izračunala homografija.
Već je pokazano da je kod homogenih koordinata jedino bitan omjer elemenata pa tako i homografiju možemo nazvati homogenom matricom. Slijedi da, pošto u matrici ima 8 neovisnih omjera između 9 elemenata, H ima 8 stupnjeva slobode.
Neka $\mathrm{I}_1 \subset \mathbb{P}^2$ predstavlja prvu sliku, a $\mathrm{I}_2 \subset \mathbb{P}^2$ drugu i neka su $\mathrm{x}_1 \in \mathrm{I}_1$ i $\mathrm{x}_2 \in \mathrm{I}_2$ podudarajuće točke. Točke se sastoje od tri koordinate, no treća koordinata je proizvoljna pa točke imaju 2 stupnja slobode. To znači da je za izračun projektivne transformacije između dvije slike potrebno 8 / 2 = 4 točke. Za izračun homografije mogu se koristiti i podudarajuće linije na slikama kojih je isto potrebno minimalno 4.

Ako imamo točno 4 podudarajuće točke, matrica H može se egzaktno izračunati. No, ako ih imamo više, teško ju je odrediti tako da se sve poklapa. Razlog tome je što uglavnom uvijek postoje manje greške pri mjerenju točaka te se može dogoditi da točke koje su određene kao podudarajuće uopće to nisu. Zato se čitav

skup točaka uglavnom neće u potpunosti poklapati niti s jednom projektivnom transformacijom. Problem izračuna homografije tada se svodi na aproksimiranje pri čemu se odabire homografija koja je najbolja od pronađenih.
Iz podudarajućih točaka homografija se može računati DLT metodom, a o tom postupku može se pročitati više u [4].

3.3.1 RANSAC

RANSAC, tj. Random Sample Consensus, kreirali su Fischler i Bolles 1981. godine. To je iterativna metoda koja služi za određivanje parametara modela iz podataka koji se mogu podijeliti na *inliere* i *outliere*. U početku imamo skup podataka iz kojih trebamo odrediti model. *Inlieri* su podaci iz skupa koji se uklapaju u model, a *outlieri* su oni koji se ne uklapaju u model. Kao što je već ranije objašnjeno, ako imamo više od 4 podudarajuće točke, uglavnom nećemo moći pronaći model u kojeg će se uklapati sve točke, odnosno imat ćemo podjelu na *inliere* i *outliere*. Cilj je pronaći model koji će imati što više *inliera*.
Za razliku od nekih drugih metoda koje u početku uzimaju više podataka iz skupa, dobivaju inicijalno rješenje pa zatim pokušavaju eliminirati neuklapajuće podatke, RANSAC kreće od najmanjeg mogućeg inicijalnog podskupa skupa podataka, a zatim ga povećava dodavajući podatke koji su konzistentni u odnosu na model, ako takvi postoje [3].

Algorithm 1 RANSAC

Ulaz:
P - skup podataka, |P| $\geq$ n
n - minimalan broj podataka potreban za određivanje modela
ϵ - tolerancija greške koja određuje uklapa li se podatak u model
t - minimalan broj *inliera* modela da bi model bio prihvaćen
N - maksimalan broj iteracija

Izlaz:
M_{best} - dobiveni model

1: **procedura** RANSAC(P, n, ϵ, t, N)
2: i $\leftarrow$ 0
3: **sve dok** i<N **činiti**
4: nasumično odabrati podskup $S_i \subseteq P$, $|S_i| = n$
5: koristeći S_i odrediti model M_i
6: odrediti podskup $S_i^* \subseteq P$ tako da on sadrži sve točke iz P koje se uklapaju u model M_i s dopuštenom tolerancijom greške ϵ
7: **ako** $|S_i^*| \geq t$ **tada**
8: izračunati novi model M_i^*
9: $M_{best} \leftarrow M_i^*$
10: $t \leftarrow |S_i^*|$
11: i $\leftarrow$ i + 1
12: **kraj**
13: **vrati** M_{best}

U slučaju homografije skup P sadrži točke, a model je homografija. Tolerancija greške ϵ koristi se na sljedeći način - neka imamo točku (x, y) i podudarajuću točku (x', y') te model h :

$$h(x, y) = (x'', y'').$$

Ako vrijedi

$$x'' \in \left\langle x' - \epsilon, x' + \epsilon \right\rangle$$

$$y'' \in \left\langle y' - \epsilon, y' + \epsilon \right\rangle,$$

onda se točka (x, y) uklapa u model h.

Ako je problem takav da se na neki bolji i smisleniji način može provoditi određivanje n podataka koji će ulaziti u podskupove S_i, onda je bolje koristiti takav deterministički pristup nego nasumično biranje.

3.3.2 Primjena homografije na sliku

Neka imamo sliku koju možemo nazvati *src*. Prvo je potrebno odrediti dimenziju slike: označimo s h visinu, a s w širinu slike. Zatim se pravi matrica dimenzije $3 \times (w \cdot h)$:

$$C = \begin{bmatrix} 0 & 1 & 2 & ... & w-1 & 0 & 1 & 2 & ... & w-1 & ..... & 0 & 1 & 2 & ... & w-1 \\ 0 & 0 & 0 & ... & 0 & 1 & 1 & 1 & ... & 1 & ..... & h-1 & h-1 & h-1 & ... & h-1 \\ 1 & 1 & 1 & ... & 1 & 1 & 1 & 1 & ... & 1 & ..... & 1 & 1 & 1 & ... & 1 \end{bmatrix}$$

Ova matrica predstavlja lokacije piksela na slici u homogenim koordinatama i s njome množimo inverz matrice homografije kako bi dobili transformirane lokacije piksela sa slike

$$C' = H^{-1} \cdot C, C' \in \mathbb{R}^{3 \times (w \cdot h)}.$$

Koristi se inverz matrice H jer će se kod pridavanja vrijednosti novoj slici koristiti *forward* mapiranje.
Točke u homogenim koordinatama koje se dobiju moraju se normalizirati kako bismo dobili nehomogene koordinate, tj. prvi i drugi redak matrice C' moramo podijeliti s trećim retkom.
Formula za primjenu homografije na točku (x, y) može se zapisati na sljedeći način:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Dakle, piksel na lokaciji (x, y) na novoj slici nalazit će se na lokaciji (x', y').

Pošto matrica homografije sadrži i negativne vrijednosti, možemo dobiti i negativne lokacije u novoj matrici C' pa se retcima dodaju njihove najmanje vrijednosti kako bi sve lokacije bile pozitivne i ne bismo izgubili dio slike. Dakle prvom retku od C' dodaje se najmanji broj iz prvog retka, a drugom retku najmanji broj iz drugog retka.
Kreira se nova slika čije se dimenzije određuju iz najvećih vrijednosti lokacije u novim lokacijama C'. Novoj slici, koju možemo nazvati *dest*, pridodaju se vrijednosti na sljedeći način :

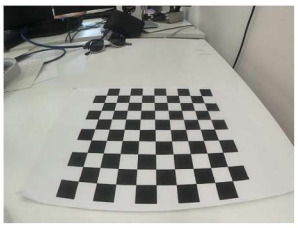
$$dest[C'[0, x], C'[1, y]] = src[x, y].$$

Pošto su nove lokacije decimalne vrijednosti, potrebno ih je zaokružiti na cijeli broj, npr. uzme se njihov strop ili pod. Kako bismo imali što manje praznina na slici zbog zaokruživanja, najbolje je uzeti i strop i pod.
Kod *forward* mapiranja svejedno se vrlo lagano dogodi da dobivena slika ima praznine. Razlog tome je što, pogotovo kod prebacivanja u ptičju perspektivu, nova slika može biti veća, a stara slika nema toliko piksela. Svi pikseli iz stare slike pridodani su svojim odgovarajućim lokacijama na novoj slici, ali nema ih dovoljno pošto je nova slika veća, odnosno razvučena na određenim dijelovima. Primjer ovoga može se vidjeti na slici 3.1b.

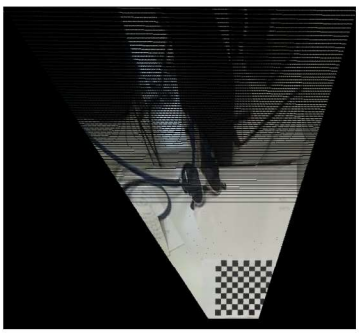
Jedno moguće poboljšanje za smanjivanje praznih prostora je koristiti gušću mrežu za homogene koordinate lokacija, tj. mrežu koja ne sadrži samo cjelobrojne vrijednosti lokacija piksela nego i decimalne vrijednosti. Ako npr. koristimo prepolovljene lokacije piksela, matrica C je :

$$C = \begin{bmatrix} 0 & 0.5 & 1 & ... & w-1 & 0 & 0.5 & 1 & ... & w-1 & ..... & 0 & 0.5 & 1 & ... & w-1 \\ 0 & 0 & 0 & ... & 0 & 0.5 & 0.5 & 0.5 & ... & 0.5 & ..... & h-1 & h-1 & h-1 & ... & h-1 \\ 1 & 1 & 1 & ... & 1 & 1 & 1 & 1 & ... & 1 & ..... & 1 & 1 & 1 & ... & 1 \end{bmatrix}$$

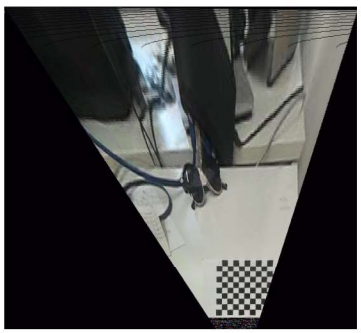
Dimenzije su joj tada $3 \times ((2w\text{-}1) \cdot (2h\text{-}1))$ što je mana ovog pristupa jer povećava memorijski prostor koji je potreban za zapis podataka, a može povećati i vrijeme izvršavanja ovisno o tome koliko gustu mrežu lokacija piksela napravimo.
Kod ovog pristupa potrebno je zaokružiti i vrijednosti originalnih lokacija piksela. Ako se koristi 0.5, može se koristiti strop i pod kako bi bilo što manje praznina.



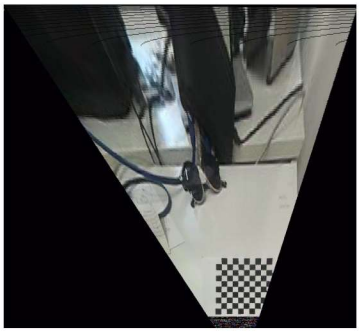
(a) Originalna slika



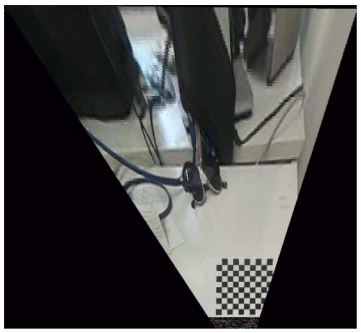
(b) Ptičja perspektiva slike s cjelobrojnim gridom lokacija



(c) Ptičja perspektiva slike s gridom razmaka lokacija 0.5.



(d) Ptičja perspektiva slike s gridom razmaka lokacija 0.25



(e) Ptičja perspektiva slike s gridom razmaka lokacija 0.2. Ovdje više nema praznih piksela

Slika 3.1: Prikaz praznih piksela koji nastaju kod primjenjivanja homografije na slici. Vidljivo je da ako koristimo gušći grid lokacija s float vrijednostima, praznina ima manje.

Još jedno poboljšanje je korištenje neke od metoda interpolacija kao što je npr. bilinearna interpolacija. Tada se na lokaciji (x, y) vrijednosti piksela određuju aritmetičkom sredinom vrijednosti na lokacijama $(x-1, y)$, $(x+1, y)$, $(x, y-1)$

i $(x, y+1)$. No, ovaj pristup nije dobar kod slika gdje se praznine sastoje od više uzastopnih piksela.

OpenCV ima funkciju warpPerspective() koja primjenjuje homografiju na sliku i daje odlične rezultate bez praznih prostora. Ali, ako se koristi transformiranje slike u stvarnom vremenu, kao kod npr. pomoći pri parkiranju u nekim automobilima, ova je funkcija prespora kod većih slika. Prebacivanjem slike u ptičju perspektivu nastaje puno veća slika od originalne. Npr., na slici 6.5 originalna slika je veličine 1280×720, a slika u ptičjoj perspektivi 5879×2715. Koliko će slika biti veća ovisi o položaju kamere u odnosu na tlo, ali, osim ako je kamera nagnuta da je slika već gotovo u ptičjoj perspektivi, nova slika će biti veća.
Sve dok ne mijenjamo položaj kamere, sve slike iz nje imat će jednaku matricu homografije za prebacivanje u ptičju perspektivu. Zato od postupka primjene homografije većinu možemo odraditi samo jednom jer je potrebna samo veličina slike koja je jednaka za sve slike iz kamere. Dakle, samo jednom računamo matricu C'. Remapiranje vrijednosti jedini je dio postupka koji se radi za svaku pojedinu sliku jer ovisi o vrijednostima piksela slike. Tada je proces transformacije puno brži i može se koristiti u stvarnom vremenu.

4 | Heuristike i metaheuristike

Heuristički algoritmi koriste se kod optimizacijskih problema za koje je teško pronaći egzaktno rješenje. To može biti jer je prostor rješenja (skup mogućih rješenja) prevelik za kompletno pretraživanje, ali i sam problem može biti prekompleksan ili imati ograničenja zbog kojih je teško pronaći bilo koje dopustivo rješenje. Heuristike ne pretražuju cijeli prostor rješenja pa nije osigurano da će pronaći optimalno rješenje, ali, ako su dobro prilagođene problemu, uglavnom pronalaze dovoljno dobra rješenja koja su blizu optimalnom, a često i optimalna.
Cilj rješavanja optimizacijskih problema je pronaći globalno optimalno rješenje pa se problemi optimizacije dijele na probleme minimizacije i probleme maksimizacije, ovisno o tome traži li se globalni minimum ili globalni maksimum funkcije.

Definicija 2. *Funkcija $f : S \to \mathbb{R}$ u točki $s^* \in S$ postiže globalni minimum na S ako je $f(s^*) \leq f(x), \forall\, x \in S$.*

Definicija 3. *Funkcija $f : S \to \mathbb{R}$ u točki $s^* \in S$ postiže globalni maksimum na S ako je $f(s^*) \geq f(x), \forall\, x \in S$.*

Problemi minimizacije lako se mogu prebaciti u probleme maksimizacije i obratno:

$$f(x) \to \min_x \quad \leftrightarrow \quad -f(x) \to \max_x$$

Optimizacijski problemi definirani su skupom S i funkcijom f, gdje je S prostor rješenja, odnosno skup svih mogućih rješenja koji se pretražuje, a f funkcija cilja koju treba optimizirati, odnosno čiji se globalni optimum traži. Problem može sadržavati i više funkcija koje treba minimizirati ili maksimizirati. Prostor rješenja definira se pomoću ograničenja/uvjeta koji su postavljeni na rješenje, odnosno varijable modela.
Jedan primjer heuristika su pohlepni algoritmi. Oni spadaju u konstruktive heuristike jer rješenje kreiraju pridjeljivanjem vrijednosti jednoj po jednoj varijabli odluke. To znači da kompletno rješenje dobivamo tek na kraju izvršavanja algoritma. U svakom koraku pohlepnog algoritma određenoj varijabli odluke pridaje se vrijednost koja je najbolja s obzirom na funkciju cilja, a ta varijabla više se ne mijenja. Pohlepni algoritmi kreiraju se za svaki specifični problemi, odnosno nemaju neki općeniti postupak koji se prilagodi većem broju problema.

Metaheuristike su *high-level* tehnike koje su generalnije od heuristika, odnosno mogu se upotrijebiti za rješavanje različitih skupina problema. Sadrže upute i strategije pomoću kojih se mogu razviti heuristički optimizacijski algoritmi [1].

Naziv metaheuristike dao je Glover 1986. godine, a kombinacija je grčkog prefiksa meta, što znači iznad, i grčke riječi *heuriskein*, koja znači tražiti [1].
Pri dizajniranju svake metaheuristike vodi se računa o načinu prikaza rješenja i definiranju funkcije cilja. Prikaz (kodiranje) rješenja utječe na efikasnost pretrage prostora rješenja, a samim time i na kvalitetu pronađenog rješenja. Prikaz rješenja ovisi o samom problemu i podacima koji trebaju biti upisani u rješenje (npr. jesu li u pitanju realne ili cijele vrijednosti). Svako moguće rješenje mora se moći prikazati odabranim kodiranjem te mora postojati put između svaka dva rješenja iz prostora pretraživanja. Za većinu problema postoji više izbora za prikaz rješenja koji zadovoljava te kriterije, a odabrati najefikasniji prikaz može biti dosta izazovno. Funkcija cilja još se zove funkcija dobrote, a pomoću nje se mjeri kvaliteta svakog rješenja u prostoru pretraživanja i omogućava njihovo međusobno uspoređivanje. Funkcija cilja zapravo vodi pretragu prostorom rješenja tako da, ako nije dobro definirana, može dovesti do loših rješenja. Može se koristiti funkcija cilja izvornog problema, ali nekad je potrebno funkciju cilja transformirati kako bi pretraživanje bilo efikasnije.
Metaheuristike pri rješavanju problema koriste dvije bitne tehnike - diverzifikaciju i intenzifikaciju. Diverzifikacija potiče istraživanje dijelove prostora rješenja koji se nisu još posjetili dovoljno često, odnosno osigurava ravnomjerno pretraživanje cijelog prostora rješenja. Na taj način isprobava se više različitih rješenja i povećava vjerovatnost da algoritam neće završiti u lokalnom optimumu. Intenzifikacija potiče istraživanje dijelova prostora rješenja koja se čine boljima. Na taj način se omogućava da se u slučaju kada se isprobava rješenje blizu optimalnog, nastavlja istraživati slična rješenja kako bi algoritam stigao do optimalnog. Čak i ako nismo blizu optimalnom u potrazi, slična rješenja vjerovatno daju jednako dobre rezultate pa se trenutno najbolje rješenje može poboljšati nekim sličnim izborom.
Najveći nedostatak metaheuristika je što može biti teško prilagoditi sve parametre modela specifičnom problemu. Parametri utječu i na kvalitetu rješenja i na efikasnost pretrage prostora rješenja pa ih je potrebno pažljivo odabrati. Ako za problem nemamo već neku intuiciju ili logiku kakve parametre treba koristiti, svaki parametar, kao i njihove različite kombinacije, moraju se isprobati kako bi se pronašao model koji će najbolje riješiti problem. To može biti vrlo osjetljiv i dugotrajan proces, a teorijski rezultati koji su nam dostupni uglavnom nisu dovoljni u praksi pri suočavanju s novim teškim optimizacijskim problemima [2].
Najpoznatiji primjeri metaheuristika su lokalno pretraživanje, tabu pretraživanje, evolucijski algoritmi, optimizacija kolonijom mravi, optimizacija rojem čestica, simulirano kaljenje, itd.

4.1 Evolucijski algoritmi

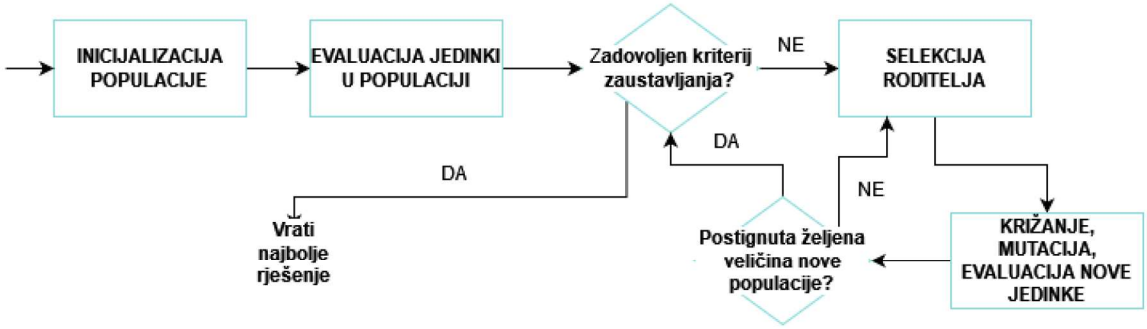
Evolucijski algoritmi (EA) su tehnike pretraživanja inspirirane biološkom evolucijom, a pojavili su se krajem 1950-tih. Oni teže imitirati ponašanja živih organizama, kao i u Darwinovoj teoriji, gdje jedinki ima više nego sredstava pa samo najjače jedinke opstaju. U početku se nisu puno koristili zbog predugog vremena izvršavanja, no u posljednjih 10 godina su se povećale računalne mogućnosti i

razvile su se paralelne arhitekture pa je popularnost evolucijskih algoritama porasla [2]. Ideja evolucijskih algoritama je da postupno evoluiraju preko populacija u svakoj iteraciji, točnije da se populacije stalno poboljšavaju.
Populacije se koriste kako bi se postigla komunikacija među rješenjima. Može se reći da sama inteligencija pojedinaca u prirodi proizlazi iz populacije. Npr. mrav će teško preživjeti sam u prirodi, ali jedna kolonija mravi lako pronalazi najkraći put do hrane, može stvoriti povezane mreže tunela, itd. [7]. Dakle, komunikacija je ono što pridonosi inteligenciji populacije i zato se koristi u evolucijskim algoritmima.
Jedan način kako se pridonosi diverzifikaciji su slučajnosti. Većinom pri rješavanju problema slučajnosti ne smatramo dobrima jer želimo pronaći egzaktno rješenje i mora postojati izravan put prema njemu bez drugih utjecaja. No, slučajnosti su zapravo bitna komponenta evolucijskih algoritama. Pomoću slučajnosti pretražuje se veći i raznolikiji dio prostora pretraživanja. Određena doza nasumičnosti isto je jedna komponenta inteligencije živog svijeta [7]. Nasumičnosti u jednom evolucijskom algoritmu ne bi trebalo biti previše kako se pri pretraživanju ne bi skroz odmaklo od nekih smislenijih rješenja, ali opet dovoljno da se ne bi isprobavala samo slična i skoro pa ista rješenja. Evolucijski algoritmi zbog nasumičnosti spadaju u stohastičke metaheuristike jer za isto početno rješenje mogu dobiti različita konačna rješenja.

4.1.1 Glavni parametri evolucijskih algoritama

Kao i kod nekih od ostalih metaheuristika, među komponentama dizajna EA nalaze se prikaz rješenja, kriterij zaustavljanja, inicijalna populacija te funkcija cilja. Komponente koje su specifične za EA su selekcija roditelja, reprodukcija te selekcija jedinki i zamjena u populaciji.
Pretraživanje prostora rješenja provodi se iterativno dok se ne dostigne kriterij zaustavljanja. Najčešći kriterij zaustavljanja je broj iteracija, ali može se promatrati i poboljšanje kroz iteracije, vrijednost najboljeg postignutog rješenja, i slično.
Pri odabiru inicijalne populacije, diverzifikacija se može pojačati nasumičnim generiranjem jedinki. Ako se jedinke biraju na drugačiji i smisleniji način, uzimajući u obzir problem, pojačava se intenzifikacija. Veličina populacije u algoritmu može biti dinamična, tj. mijenjati se kroz iteracije, no uglavnom se koristi statična veličina populacije.
Nakon odabira inicijalne populacije, kreće prva iteracija algoritma. To znači da je idući korak selekcija roditelja. Iz populacije se biraju jedinke iz kojih će rekombinacijom nastati nova jedinka. Ima više metoda kojim se roditelji mogu odabrati, ali uvijek mora vrijediti da vjerojatnost odabira neke jedinke ovisi o njenoj prikladnosti, tj. jedinke koje su bolje trebale bi imati veću vjerojatnost da budu izabrane. No, i one lošije jedinke bi trebale imati vjerojatnost veću od 0 da budu izabrane za roditelja. Najčešće korištena metoda je k-turnir. Iz populacije se bira k jedinki koje se zatim rangiraju po vrijednosti funkcije cilja i najbolje jedinke postaju roditelji.
Zatim slijedi reprodukcija koja se dijeli na rekombinaciju i mutaciju. Reprodukcijom nastaju nove jedinke. Rekombinacija je više-roditeljska reprodukcija, a mutacija jedno-roditeljska.

Ideja rekombinacije je preuzeti dio karakteristika iz svakog od roditelja te kombinacijom tih karakteristika dobiti njihovog potomka. Najčešće se koristi metoda križanja kod koje se koriste dva roditelja.
Nakon što je rekombinacijom dobivena nova jedinka, ona može mutirati čime nastaje drugačija jedinka. Mutacija se koristi radi diverzifikacije, a vjerojatnost da jedinka mutira se definira pri pokretanju algoritma. Uglavnom se koristi mala vjerojatnost, između 0.001 i 0.015. Način na koji je mutacija definirana također ovisi o prikazu rješenja te o prostoru pretraživanja pošto želimo dobiti jedinku koja je dopustivo rješenje. Kod prikaza rješenja kao binarnog vektora, primjer mutacije je preokretanje bita, kod vektora s realnim vrijednostima dodavanje nekog nasumičnog broja elementu, itd. Kao i kod križanja, može se kombinirati više operatora mutacije radi još veće diverzifikacije, a može se mutirati i više od jednog gena jedinke.



Slika 4.1: Glavni koraci generacijskog evolucijskog algoritma.

4.1.2 Genetički algoritmi

Genetički algoritmi (GA) su jedna vrsta evolucijskih algoritama. Dijele se na eliminacijske i generacijske.
Kod eliminacijskih algoritama u svakoj iteraciji mijenja se samo jedna jedinka dok ostatak populacije ostaje nepromijenjen. Npr. nova jedinka u populaciji može zamijeniti trenutno najgoru jedinku.
Kod generacijskih algoritama u svakoj se iteraciji kreira nova populaciija. Stara generacija koristi se za selekciju roditelja, a nove jedinke dodaju se u privremenu populaciju koja kada se popuni zamijenjuje staru. Populacija je popunjena kada dostigne željenu veličinu koja je uglavnom jednaka veličini stare populacije. Dakle, u svakoj iteraciji kreira se nova populacija, a prethodna se briše kako bi ju nova mogla zamijeniti.
Može se dogoditi, pogotovo kod generacijskih GA, da se tijekom izvršavanja algoritma izgubi najbolje pronađeno rješenje. Npr. kod generacijskih GA možemo izgubiti najbolje rješenje pri brisanju prethodne populacije nakon kreiranja nove. Elitizam je svojsto da se najbolje pronađeno rješenje ne može izgubiti. Elitizam se u GA dodaje jednostavno - npr. u generacijskom GA se pri kreiranju nove populacije može dodati jedno ili više najboljih rješenja iz stare populacije u novu. U većini slučajeva ovo donosi znatno poboljšanje rezultata.

5 | Pretvorba u ptičju perspektivu

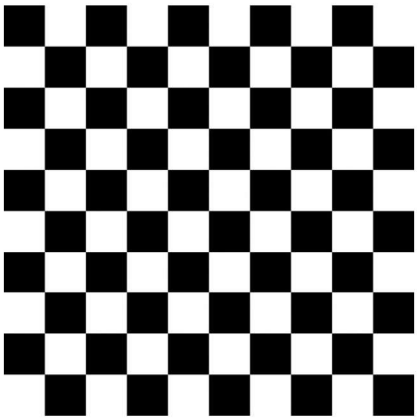
5.1 Homografija za ptičju perspektivu

Da bismo sliku prebacili u ptičju perspektivu, potrebna nam je homografija pomoću koju ćemo transformirati sliku. Ako promotrimo uzorak šahovske ploče, kao na slici 5.1, možemo primijetiti da je on u ptičjoj perspektivi. To znači da ako slike sadrže ovaj uzorak, možemo doći do ptičje perspektive računanjem homografije između uzorka na samoj slici i uzorka u ptičjoj perspektivi. Homografija se onda može računati koristeći samo podudarajuće točke na uzorku. OpenCV ima funkciju koja pronalazi kuteve šahovske ploče (u ovom slučaju je to $9 \times 9 = 81$ točaka). Korištenjem te funkcije umjesto pronalažanje podudarajućih ključnih točaka povećava se točnost pronađenih točaka te odmah znamo koje su podudarajuće točke s obzirom da su u vektoru u određenom poretku koji bi trebao biti jednak na obje slike. Mana te funkcije je što ne pronađe uvijek dobru orijentaciju, odnosno poredak točaka nije uvijek jednak, kao što je vidljivo na slici 5.3 pa se mora odraditi dodatna provjera i ispravak poretka točaka u vektoru kako bi poredak bio jednak kao kod uzorka u ptičjoj perspektivi. Često se, zbog te neosjetljivosti na orijentaciju, koriste uzorci šahovske ploče s različitim brojem redaka i stupaca jer se u kodu unose njihovi brojevi da bi ih algoritam znao razlikovati. No, tada se opet može dogoditi da vrhovi budu u obrnutom poretku. U problemu koji se obrađuje u ovom radu uzorak mora biti s jednakim brojem redaka i stupaca zbog funkcije cilja koja će biti objašnjena u nastavku.
Pronalaskom homografije između uzorka na sceni i uzorka u originalnoj slici dobije se ptičja perspektiva slike, ali samo dijela gdje se nalazi uzorak, a ostatak slike se prilagođava tom dijelu. Za svaki dio scene koji želimo prebaciti u ptičju perspektivu trebao bi se uslikati uzorak na tom dijelu i izračunati homografija. Znači, da bi se prebacila cijela slika u ptičju perspektivu, morao bi se uzorak snimiti na različitim dijelovima scene. No, cilj je imati jednu zajedničku transformaciju za cijelu sliku, a ne transformacije po dijelovima slike. Iz tog razloga bi se homografije dobivene iz različitih dijelova trebale spojiti u jednu. Takva homografija neće biti najbolja za pojedinačne dijelove, ali će se sveukupno najviše podudarati sa svim dijelovima.

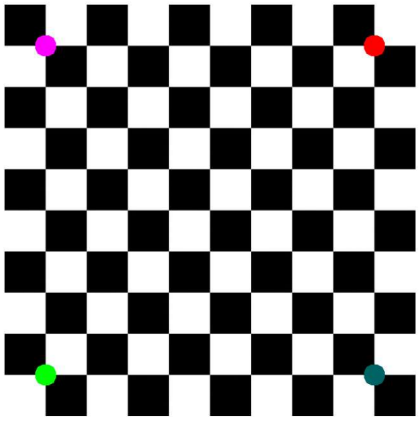
5.2 Genetički algoritam za optimizaciju homografije

Postavlja se pitanje kako uopće mjeriti koliko je homografija dobro prebacila uzorak iz slike u ptičju perspektivu. Promotrimo li opet sliku uzorka 5.1, pošto ima

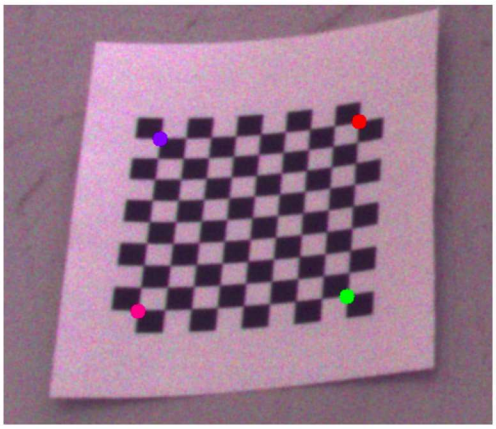
jednak broj vertikalnih i horizontalnih polja, ima jednake duljine stranica, tj. ima oblik kvadrata. Dakle, u ptičjoj perspektivi uzorak bi trebao biti kvadrat. Greška transformiranog uzorka na slici se onda može mjeriti tako da se pronađu vrhovi uzorka, kao na slici 5.2, izračunaju udaljenosti između susjednih vrhova i uzmu minimalna i maksimalna udaljenost te se od maksimalne udaljenosti oduzme minimalna. Ako je uzorak potpuno točno prebačen u ptičju perspektivu, onda bi ta razlika trebala biti 0 jer su kvadratu sve stranice jednake duljine. Iz te ideje stvorila se mogućnost korištenja ove funkcije za mjerenje greške kao funkcije cilja koju je potrebno minimizirati u metaheuristici.
Slična funkcija ovoj korištena je u radu [5], u kojem je za optimizaciju funkcije greške korištena Powellova kvadratno konvergentna metoda. Zbog toga što GA koristi populacije, tj. iz više jedinki dobiva najbolje rješenje, može se koristiti za dobivanje jedne zajedničke, najbolje moguće, homografije za cijelu scenu.



Slika 5.1: Uzorak šahovske ploče s jednakim brojem redaka i stupaca, korišten za izračun homografije.



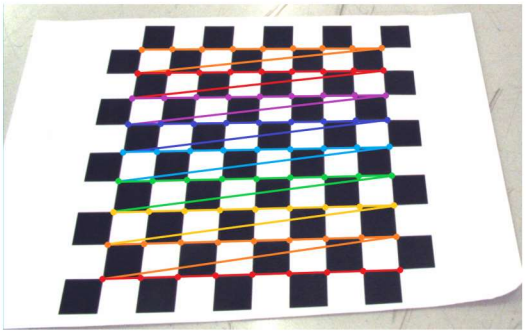
(a) Originalna slika uzorka u ptičjoj perspektivi čije su stranice jednake duljine, tj. susjedni vrhovi jednako udaljeni



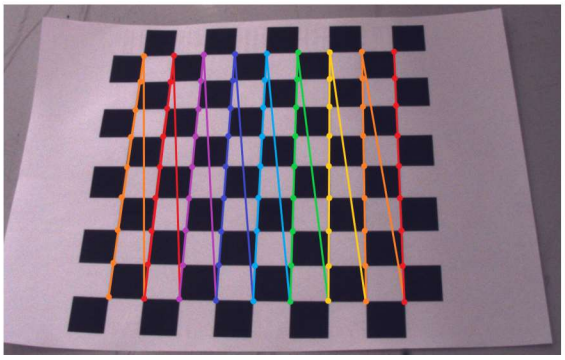
(b) Slika uzorka na sceni koji nije u ptičjoj perspektivi. Susjedni vrhovi uzorka nisu jednako udaljeni.

Slika 5.2: Razlika u uzorku šahovske ploče na dvije slike gdje je jedna slika u ptičjoj perspektivi.

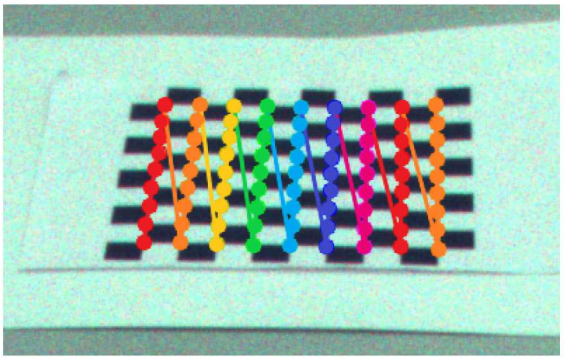
(a) Ispravan poredak vrhova uzorka.



(b) Krivi (obrnuti) poredak vrhova.



(c) Krivi poredak vrhova. Za ispravak se koristila kombinacija rotiranja i reversanja vektora vrhova.



(d) Krivi poredak vrhova. Za ispravak se koristila rotacija vektora vrhova.

Slika 5.3: Različiti poredak točaka pronađenih na istom uzorku. Poredak se ispravlja s obzirom na poredak koji je na uzorku u ptičjoj perspektivi (slika 5.1), tj. taj poredak smatramo ispravnim.

5.3 Parametri genetičkog algoritma

U poglavlju o evolucijskim algoritmima opisani su glavni parametri evolucijskih algoritama, a time i genetičkog algoritma. Parametri poput prikaza rješenja, kriterija zaustavljanja i funkcije cilja su za ovaj problem bili odabrani pri početku kodiranja. Drugi parametri, poput selekcije roditelja, reprodukcije i mutacije, a donekle i inicijalne populacije, su morali biti testirani kako bi se odredili najbolji za ovaj problem.
Korišten je generacijski genetički algoritam u kojem se u svakoj iteraciji kreira nova populacija, te je dodan elitizam pa se pri početku svake iteracije u novu populaciju dodavala prva dva najbolja rješenja iz prethodne populacije.

5.3.1 Prikaz rješenja i populacija

Za prikaz rješenja odabran je vektor s realnim vrijednostima, duljine 9, koji predstavlja matricu homografije. Vrijednosti u homografiji mogu jako varirati te iz tog razloga nije bilo puno drugih mogućnosti što se tiče odabira prikaza.
Populacija se sastoji od matrica homografije, a inicijalna populacija kreira se iz

skupa slika. Dakle, snima se skup slika koji prikazuje jednu scenu, a svaka slika sadrži uzorak šahovske ploče s 10×10 polja. Na svakoj slici se uzorak nalazi na drugačijem mjestu u sceni. U početku se pri kreiranju skupa slika trudilo da bude podjednak broj slika s uzorkom na svim mjestima, no kasnije se došlo do zaključka da je bolje kreirati više slika s uzorkom u prednjem dijelu scene zato što je taj dio najbitnije točno prebaciti u ptičju perspektivu. Npr. ako se ptičja perspektiva koristi u vozilu pri parkiranju, dio koji je najbliži vozilu je najbitnije prikazati točno. Time se povećala vrijednost tog dijela jer homografije koje ne prebacuju prednji dio dobro u ptičju perspektivu imati će veću grešku. Općenito se dio najbliži kameri najbolje može prebaciti u ptičju perspektivu jer je u prednjem dijelu scene objektima najvidljiviji njihov gornji dio, dok objekte u stražnjim dijelovima scene vidimo samo s njihove prednje strane, a gornji dio vidimo samo malim dijelom.
Dakle, za svaku sliku, izračuna se homografija između njenog uzorka i originalne slike uzorka kako bi se dobila transformacija uzorka u ptičju perspektivu na tom dijelu scene. Te homografije čine inicijalnu populaciju.
Korištena je statična veličina populacije koja je jednaka broju slika u skupu, a algoritam je testiran na skupovima različitih veličina, od minimalno 6 slika do maksimalno 50-ak. Skup od 6 slika korišten je zbog brzine izvršavanja algoritma pri testiranju korektnosti funkcija. Većina skupova sadržavala je oko 20 slika. Također su skupovi imali različite udaljenosti kamere do scene. Maksimalna udaljenost najudaljenijeg objekta je bila 170 cm. Sve dok su slike bile dovoljno kvalitetne da se točke uzorka mogu jasno očitati, udaljenost do kamere nije utjecala na kvalitetu dobivenog rješenja.

5.3.2 Kriterij zaustavljanja

Za kriterij zaustavljanja korišten je unaprijed određen broj iteracija. Taj broj ovisi o skupu slika, no kod većine skupova korišten je manji broj od veličine populacije. Broj iteracija uglavnom je bio između 2 i 15. Poboljšanje populacije kroz iteracije također bi se mogao koristiti kao kriterij zaustavljanja.

5.3.3 Funkcija cilja

Rješenje, tj. homografija, primjenjuje se na svaku sliku u populaciji, zatim se računa greška na način kao što je već opisano. Greške svake od slika u populaciji se zbrajaju i to čini vrijednost funkcije cilja te homografije.
Može se dogoditi da je homografija jako loša, tj. da rješenje uopće nije dopustivo jer slika bude prazna nakon transformacije. To se penalizira unaprijed određenom velikom vrijednosti funkcije cilja kako bi se osiguralo da se ta homografija ne izabere. Iznos koji je korišten za vrijednost funkcije cilja takve slike je 100000.
Može se dogoditi da homografijom uzorak na slici izađe iz vidljivog dijela slike, ili da slika izgubi kvalitetu pri transformaciji (npr. ako se jako poveća pa uzorak bude premalen na slici) ili zbog nekog drugog razloga da se uzorak uopće ne može locirati. Hoće li se ovo dogoditi dosta ovisi i o slikama. Može biti da u skupu slika puno slika sadrži uzorak na rubu scene pa zato izađe iz vidljivog dijela slike pri transformaciji. Može biti da je slika tako uslikana da se prebacivanjem uzorka u

ptičju perspektivu stvarno poveća te se uzorak onda ne može pronaći jer je malen. Zato ovo i dalje smatramo dopustivim rješenjem, ali se mora penalizirati većom vrijednosti funkcije cilja. Ako je stvarno do skupa slika, većina homografija će i dalje imati sličnu prikladnost pa će se moći uspoređivati i odabrati najbolja. Ako nije do skupa, već do homografije, zbog ovog penala se ta homografija neće odabrati. Za ovo je korištena vrijednost funkcije cilja 10000. Naravno, te vrijednosti, i u ovom slučaju i u slučaju kada je slika prazna, morale bi se prilagoditi vrijednostima kakve se izračunaju formulom kada je uzorak pronađen jer moraju biti veće tako da se te homografije ne bi izabrale ako postoje druge bolje čijom se primjenom uzorak na slikama može detektirati. Iz ovog je vidljivo da točnost pronađene homografije ovisi puno i o korištenom skupu slika.

5.3.4 Selekcija roditelja

Za selekciju roditelja korišten je k-turnir. Isprobano je korištenje 3-turnira i 4-turnira. U 3-turniru izabrane su 3 jedinke iz populacije, a dvije najbolje koriste se u križanju kako bi nastala nova jedinka. 4-turnir funkcionira na isti način, ali umjesto 3 jedinke izabire 4 jedinke iz populacije od kojih se biraju roditelji.

5.3.5 Mutacija

Korištena je vjerojatnost 0.01 da se mutacija dogodi novoj jedinki nakon njenog kreiranja. Isprobane su tri metode mutacije.
Prva mutacija je dodavanje nasumičnom elementu u matrici broj slučajno izabran između 0 i 1.
Druga mutacija je da se nasumično odabere element u matrici, točnije indeks retka i indeks stupca (i,j), i, j $\in \{1,2,3\}$ i elementu na tom indeksu dodaje se petina njega samog, tj.

$$H[i,j] = H[i,j] + \frac{1}{5}H[i,j].$$

Na ovaj način ostajemo u okviru dopustivih rješenja jer je promjena mala i nova jedinka je slična staroj, a opet se isprobava malo drugačije rješenje što utječe na diverzifikaciju rješenja.
Nakon ove mutacije, isprobana je treća mutacija u kojoj je moguće mutirati više elemenata. Prvo se nasumično bira broj elemenata koje će mutirati. Taj broj je ograničen na maksimalno 5. Nakon toga se opet nasumično biraju indeksi i svaki element mutira po prethodno opisanoj mutaciji.

5.3.6 Rekombinacija

Korištena je rekombinacija metodom križanja. Isprobano je više funkcija križanja, pojedinačno, a zatim one bolje u kombinaciji. Za križanje se uvijek koriste 2 roditelja, a funkcije koje su isprobane proizvode jednu ili dvije nove jedinke.

Križanje u jednoj točki

Prvo se nasumično odabere indeks točke križanja. Kreiraju se dvije nove jedinke. Prva jedinka uzima elemente prvog roditelja do indeksa točke križanja, a od točke do kraja vektora uzima elemente drugog roditelja. Druga jedinka radi obrnuto, tj. prvo uzima elemente drugog roditelja, a zatim prvog.

Aritmetičko križanje

Ovim križanjem nastaje jedna nova jedinka, a njeni elementi jednaki su aritmetičkoj sredini elemenata njenih roditelja:

$$H[i,j] = \frac{P_1[i,j] + P_2[i,j]}{2},$$

gdje su P_1 i P_2 izabrani roditelji nove jedinke H.

Uniformno križanje

Svaki element nove jedinke izabran je od jednog od roditelja s jednakom vjerojatnošću.

Križanje s alternirajućim pozicijama

Nova jedinka uzima jedan element iz prvog roditelja, idući element iz drugog, itd.

Flat križanje

Element na indeksu (i,j) nove jedinke uzima se iz segmenta određenog vrijednostima roditelja na tom indeksu:

$$min_{ij} = min\{P_1[i,j], P_2[i,j]\}$$

$$max_{ij} = max\{P_1[i,j], P_2[i,j]\}$$

$$H[i,j] \sim U[min_{ij}, max_{ij}]$$

Ovo križanje pridonosi diverzifikaciji populacije jer se vrijednost elemenata bira nasumično iz segmenta između roditeljskih elemenata, tj. nije samo kombinacija roditelja kao npr. aritmetičko križanje.

Lokalno križanje

Prvo se odabire broj $\lambda \sim U[0,1]$, a zatim se elementi nove jedinke postavljaju na kombinaciju roditelja :

$$H[i,j] = \lambda P_1[i,j] + (1-\lambda)P_2[i,j], \quad \forall (i,j), i,j = 1,2,3.$$

Ovo križanje je, kao i flat križanje, pridonosi diverzifikaciji populacije.

BLX-alpha križanje

Ovim križanjem nastaje jedna nova jedinka čiji se element na indeksu (i, j) računa na sljedeći način :

$$min_{ij} = min\{P_1[i,j], P_2[i,j]\}$$
$$max_{ij} = max\{P_1[i,j], P_2[i,j]\}$$
$$I = max_{ij} - min_{ij}$$
$$H[i,j] = (min_{ij} - I \cdot a) + 0.5 \cdot |(max_{ij} + I \cdot a) - (min_{ij} - I \cdot a)|$$

Za parametar a mora vrijediti $a > 0$, tj. da je pozitivan realan broj. Korištena je vrijednost :

$$a = \frac{|P_1[i,j] + P_2[i,j]|}{4}.$$

Kao i flat i lokalno križanje, ovo križanje pridonosi diverzifikaciji populacije. Ta tri križanja opisana su u [6].

Križanje zamjenom jednog elementa

Ovim križanjem nastaju dvije nove jedinke. Nasumično se odabere indeks jednog elementa te se vrijednosti roditelja na tom indeksu zamjene.

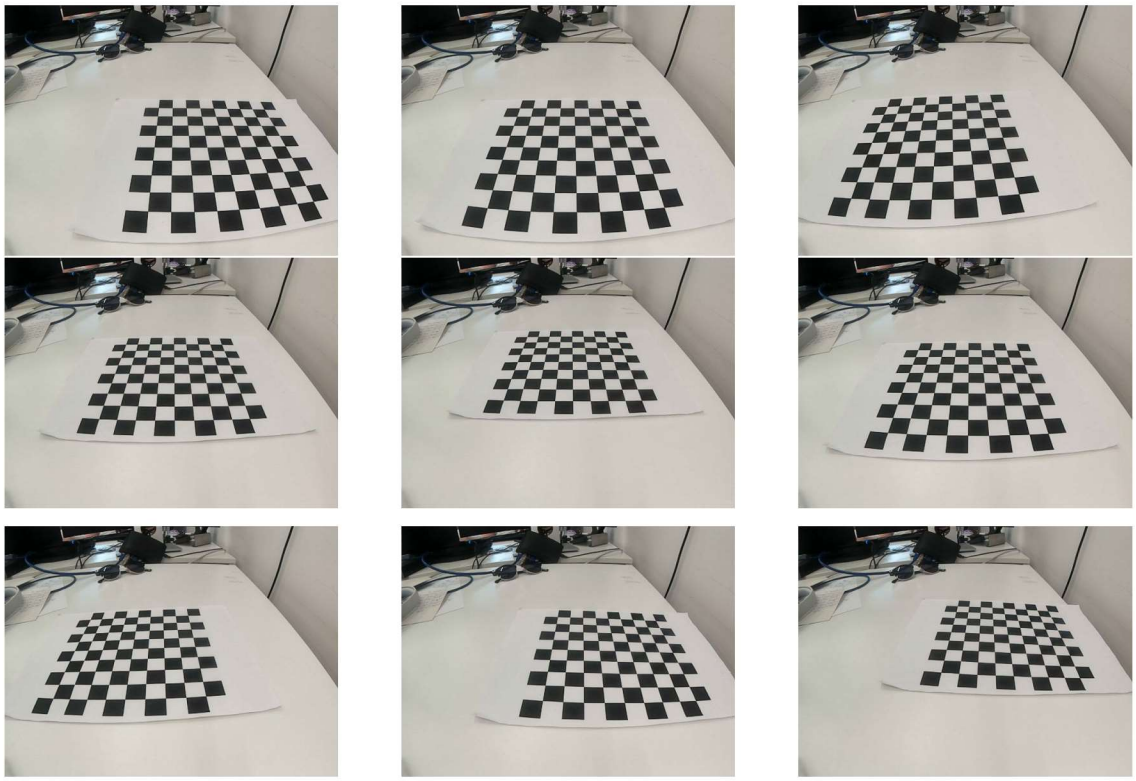
Križanje zamjenom više elemenata

Ovo križanje funkcionira na jednak način kao križanje zamjenom jednog elementa, ali se zamjenjuje više elemenata među roditeljima. Broj elemenata koji će se zamijeniti se nasumično bira između 1 i 8. Kada bi taj broj bio 9, dobili bismo dvije jedinke koje su iste kao i njihovi roditelji pa je zato ograničen s 8.

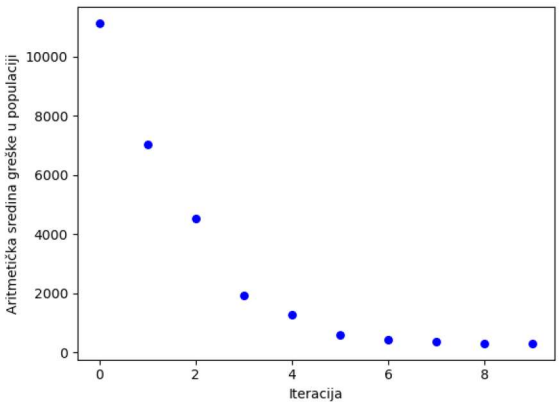
6 | Rezultati i zaključci testiranja

6.1 Populacija

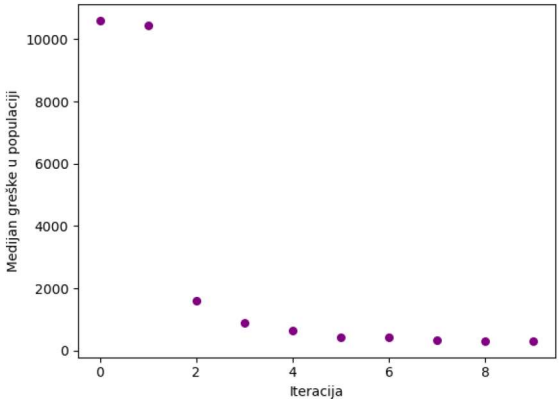
Primjer jednog skupa slika koji je korišten za stvaranje inicijalne populacije može se vidjeti na slici 6.1. Radi analize poboljšanja tijekom iteracija, dodan je ispis kvalitete trenutne populacije. Ispisuje se (po vrijednosti funkcije cilja) minimalno (tj. najbolje) i maksimalno (tj. najgore) rješenje u populaciji, aritmetička sredina i medijan vrijednosti svih rješenja u populaciji. Primjer tih vrijednosti može se vidjeti na dijagramima 6.2. Aritmetička sredina, medijan i maksimalna vrijednost funkcija cilja od prve populacije do posljednje u tom primjeru poboljšali su se za 97-99%, a minimalna vrijednost za 52%. Rezultat najbolje dobivene homografije primjera može se vidjeti na slici 6.6.



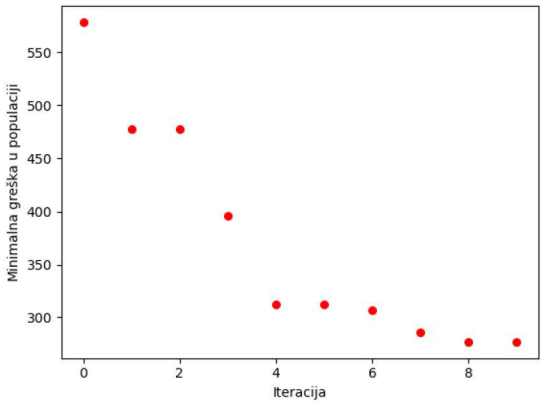
Slika 6.1: Primjer skupa slika iz kojeg se može napraviti jedna manja populacija.



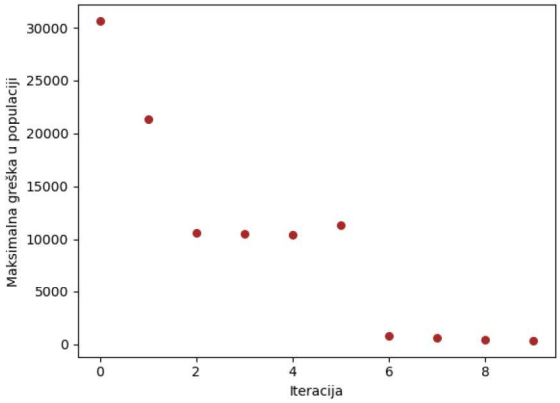
(a) Aritmetička sredina



(b) Medijan



(c) Vrijednosti funkcije cilja najboljih rješenja u populaciji



(d) Vrijednosti funkcije cilja najlošijih rješenja u populaciji. U 5. iteraciji vidljivo je da je prihvaćeno lošije rješenje u 3-turniru

Slika 6.2: Analiza poboljšanja populacije kroz iteracije.

Zbog nasumičnosti algoritma, nekad je već među prvim iteracijama pronađeno najbolje rješenje koje se više nije poboljšalo, a nekad tek među zadnjim iteracijama. Iz analize populacija kroz iteracije, vidjelo se da se populacije poboljšavaju u svakoj iteraciji te je jedino najbolje rješenje ostajalo isto po par iteracija. To se može vidjeti i u tablicama 6.3 i 6.4. U tablici 6.3 rješenje se poboljšalo u gotovo svakoj iteraciji, dok u tablici 6.4 vidimo da se od 3. do 7. iteracije najbolje rješenje nije mijenjalo. Kretanje najboljih vrijednosti funkcije cilja u populaciji iz tablice 6.3 može se vidjeti na dijagramu 6.2c. Rješenje dobiveno tom tablicom je najbolje pronađeno u 26 pokretanja algoritma nad skupom slika veličine 19, a slika 6.6 sadrži primjer slike na koju je to rješenje primijenjeno. Prikaz kvalitete rješenja pronađenih u ostalih 25 pokretanja algoritma može se vidjeti na slici 6.4, a primjer slike na koju je primijenjeno jedno od rješenja dobiveno nakon 4 iteracije nalazi se na slici 6.5.

6.2 Selekcija roditelja i mutacija

Za selekciju roditelja uglavnom je korišten 3-turnir. 4-turnir imao je slične rezultate, ali zbog veličine populacije je uglavnom korišten 3-turnir kako bi se smanjila mogućnost da se jednaki roditelji odabiru po više puta.
Od mutacija prvo je isprobano dodavanje nasumično odabranom elementu homografije broj slučajno izabran između 0 i 1. Ova mutacija isprobana je u kombinaciji s aritmetičkim križanjem i producirala je nedopustiva rješenja, tj. primjenom mutiranih homografija nastajale su prazne slike.
Zatim su isprobane mutacija dodavanja jednom elementu njegovu petinu te nadogradnja te mutacija u kojoj mutira slučajan broj elemenata, između 1 i 5. Obje mutacije davale su dobra i uvijek dopustiva rješenja. Nakon njihovog testiranja, izabrana je mutacija gdje mutira slučajan broj elemenata. Imali su slične rezultate, ali ova mutacija uključuje i slučaj kada mutira samo jedan element iako je to onda manje vjerovatno. Također, nakon više iteracija može se dogoditi, ovisno o križanju, da većina jedinki u populaciji ima sličnu vrijednost funkcije cilja i da su jedinke postale slične pa je dobro povećati diverzifikaciju.

6.3 Križanja

Prvo je isprobano križanje u jednoj točki s 3-turnirom bez mutacije. Križanje nije dalo dopustiva rješenja pa nije više korišteno nakon inicijalnog testiranja.

Iduće je isprobano aritmetičko križanje s 3-turnirom, prvo bez mutacije, a zatim i s mutacijama opisanim u poglavlju 5.3.5. Davalo je dobra rješenja, a primjer rezultata može se vidjeti na slici 6.3b. Nedostatak korištenja samog aritmetičkog križanja je što su jedinke u populaciji postajale sve sličnije te su se nakon par iteracija razlikovale samo u jednom ili dva elementa i to samo po par brojeva u decimali.

Nakon aritmetičkog križanja isprobano je križanje zamjenom elemenata - prvo jednog elementa, a zatim i slučajnog broja elemenata. Pošto ova križanja vraćaju dvije nove jedinke, isprobane su u kombinaciji s aritmetičkim križanjem koje se koristilo kada je novoj populaciji nedostajala samo jedna jedinka da bi bila potpuna, tj. najviše jednom po iteraciji. Križanja su isprobana s mutacijom jednog elementa i s mutacijom više elemenata. Križanje zamjenom slučajnog broja elemenata isprobano je i u kombinaciji s 4-turnirom. Sva križanja zamjenom dala su jednako dobre rezultate, ali se nove jedinke ne razlikuju puno od svojih roditelja pa ih je, kao i aritmetičko, bolje koristiti u kombinaciji nego samostalno. Primjer rezultata samostalnog korištenja križanja zamjenom slučajnog broja elemenata može se vidjeti na slici 6.3h.
Dvije varijante križanja zamjenom više elemenata su križanje zamjenom jednog elementa po retku i križanje zamjenom jednog elementa po stupcu. Ta križanja su također isprobana, ali nisu pronašla bolja rješenja pa nisu korištena nakon inicijalnog testiranja. Može se dogoditi da križanje slučajnog broja elemenata baš odabere indekse tako da se vrši zamjena jednog elementa u svakom retku ili stupcu

pa se ova križanja ipak mogu dogoditi, samo s malom vjerojatnošću.

Sljedeće su isprobani uniformno križanje i križanje s alternirajućim pozicijama. Ova križanja također direktno uzimaju elemente svojih roditelja, tj. ne isprobavaju nove elemente, pa ne pridonose diverzifikaciji. Oba križanja dala su dopustiva rješenja, ali u većini slučajeva rezultati nisu bili bolji u odnosu na ostala križanja pa zato nisu korištena nakon inicijalnog testiranja. Primjer se može vidjeti na slikama 6.3c i 6.3d. Također im je bilo potrebno puno više pokušaja kreiranja nove jedinke da bi se stvorila jedinka koja je bolja od jedinki izabranih u 3-turniru. Kod križanja s alternirajućim pozicijama to je dovelo do toga da se nova jedinka stvorila samo pri odabiru istih roditelja pa su se u populaciju dodavale samo jednake ili slične jedinke. Primjer ovoga može se vidjeti u tablici 6.1. Nakon 4. iteracije vrijednosti funkcije cilja aritmetičke sredine i medijana populacije te minimalne i maksimalne vrijednosti u populaciji su bile skoro pa jednake. Većina jedinki bile su identične, a par jedinki se razlikovalo u jednom ili dva elementa. Odabirom jedne od tih jedinki koja je malo drugačija u 3-turniru 5. iteracije ipak se uspjela stvoriti bolja jedinka.

Na kraju su isprobani flat, lokalno i BLX-alpha križanje. Ova tri križanja dala su odlične rezultate samostalno kao što se može vidjeti na slikama 6.3e, 6.3f i 6.3g. U kombinaciji s ostalim križanja su najviše došla do izražaja jer povećavaju raznolikost populacije, tj. spriječavaju da zbog ostalih križanja, koja uglavnom koriste samo elemente roditelja ili konstantno jednaku kombinaciju njihovih elemenata, dođe do toga da populacija ima sve elemente skoro pa jednake.

Nakon što su sva križanja testirana samostalno, napravljena je kombinacija boljih križanja. To znači da se, nakon selekcije roditelja, nasumično odabire jedno od križanja kojim će se kreirati nova jedinka. Vjerojatnosti su jednake za sva križanja. U kombinaciju su uključena sljedeća križanja - aritmetičko, flat, lokalno, BLX-alpha, zamjena jednog elementa te zamjena više elemenata.
Ovom kombinacijom postignuta su puno bolja rješenja nego kada se koristilo samo jedno križanje. Aritmetičko križanje te križanja zamjene su davali dobre rezultate, ali nije bilo puno promjene kod novih jedinki u odnosu na roditelje. Flat, lokalno i BLX-alpha križanja potaknula su raznolikost u populaciji te su istražena raznolika rješenja bez većeg pomicanja od granica dopustivosti. Na taj način je postignuta ravnoteža između intenzifikacije i diverzifikacije.

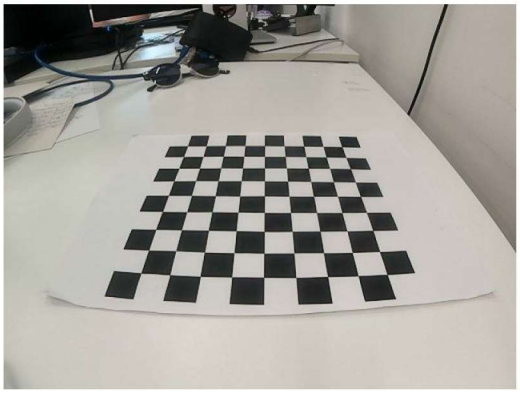
U početku su sve jedinke dobivene reprodukcijom bile prihvaćene, ali to je povećavalo broj iteracija potrebnih da bi se pronašla bolja rješenja, odnosno bolje populacije. Dodana je provjera da je nova jedinka bolja od najlošije jedinke izabrane u k-turniru. Radi diverzifikacije, s vjerojatnosti 0.05 se jedinka prihvaća i ako je lošija. Ovo je pomoglo da se pronađu bolja rješenja u manje iteracija te se populacije poboljšavaju s gotovo svakom iteracijom. Na slici 6.2d vidi se da je u 5. iteraciji prihvaćena jedinka s lošijom vrijednosti funkcije cilja u odnosu na one izabrane u 3-turniru. S obzirom da su ostale nove jedinke populacije bile bolje, aritmetička sredina i medijan vrijednosti u populaciji su se smanjili i u toj iteraciji

te je ovdje pronađeno najbolje rješenje od ukupno 26 pokretanja algoritma nad tim skupom slika.

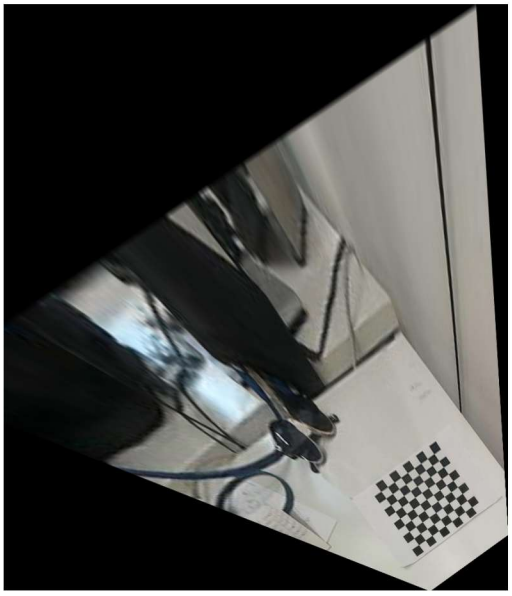
Analiza tijeka poboljšanja populacije s ovakvom kombinacijom križanja i prihvaćanja jedinki može se vidjeti u tablicama 6.2, 6.3 i 6.4. U tablici 6.2 prikazano je koliko je koje križanje korišteno u svakoj od 5 iteracija. U početku su križanja koja proizvode jedinke sličnije roditeljima, poput aritmetičkog, proizvodila većinu prihvaćenih jedinki jer su jedinke ostalih križanja većinom bile lošije od jedinki iz turnira. U kasnijim iteracijama, kada su jedinke unutar populacije postale malo sličnije, bolje jedinke proizvela su križanja poput flat i lokalnog, čije se jedinke više razlikuju od roditelja.

Analiza vrijednosti funkcije cilja jedinki u populaciji	Aritmetičko križanje	Uniformno križanje	Križanje s alternirajućim pozicijama	Flat križanje	Lokalno križanje	BLX – alpha križanje	Križanje zamjenom više elemenata	Broj iteracije
MIN	53.4797	53.4797	53.4797	53.4797	53.4797	53.4797	53.4797	0
AVG	7546.23	7546.23	7546.23	7546.23	7546.23	7546.23	7546.23	
MED	10042.5	10042.5	10042.5	10042.5	10042.5	10042.5	10042.5	
MAX	10052	10052	10052	10052	10052	10052	10052	
MIN	26.1867	53.4797	53.4797	32.5638	53.4797	26.1867	53.4797	1
AVG	5031.73	185.46	121.658	1312.57	5096.08	5031.73	5087.87	
MED	10012.6	124.126	66.0353	58.751	10037.1	10012.6	10013.2	
MAX	10043.6	519.587	298.447	10021.7	10049	10043.6	10042.5	
MIN	9.88216	53.4797	53.4797	15.4705	36.4582	9.88216	53.4797	2
AVG	1272.58	123.977	63.5556	44.4783	1433.46	1272.58	1404.64	
MED	26.1867	58.751	66.0353	36.1888	97.2525	26.1867	221.111	
MAX	10013.3	370.644	66.0353	113.41	10041.3	10013.3	10006.6	
MIN	9.88216	53.4797	53.4797	8.07722	36.4582	9.88216	49.721	3
AVG	19.4149	108.055	54.1386	21.7655	216.968	19.4149	153.528	
MED	22.4291	58.751	53.4797	22.3646	119.562	22.4291	53.4797	
MAX	27.4795	221.472	58.751	35.6457	816.487	27.4795	571.743	
MIN	9.88216	53.4797	53.4797	7.44933	36.4582	9.88216	49.721	4
AVG	16.2823	78.1298	53.4797	11.5994	119.515	16.2823	134.307	
MED	12.7011	58.751	53.4797	12.0447	104.337	12.7011	53.4797	
MAX	25.6087	219.053	53.4797	17.8743	312.02	25.6087	371.194	
MIN	9.70335	53.4797	26.5055	3.55365	36.4582	9.70335	49.721	5
AVG	10.191	58.0921	36.8128	8.19696	88.6982	10.191	79.0796	
MED	10.0326	58.751	27.4515	8.6963	66.3265	10.0326	53.4797	
MAX	11.9817	58.751	53.4797	13.7938	216.902	11.9817	188.902	

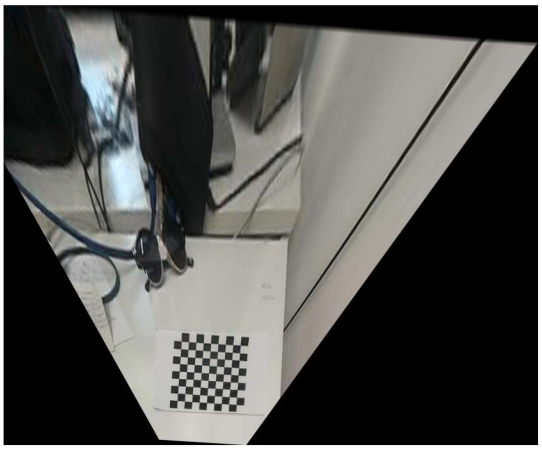
Tablica 6.1: Vrijednosti funkcije cilja jedinki unutar populacije za križanja isprobana nad skupom slika 6.1.



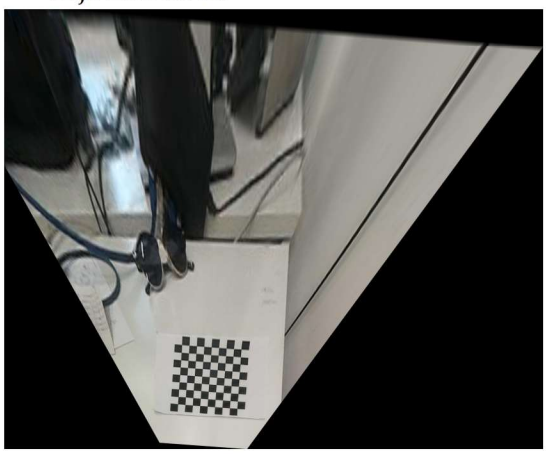
(a) Originalna slika



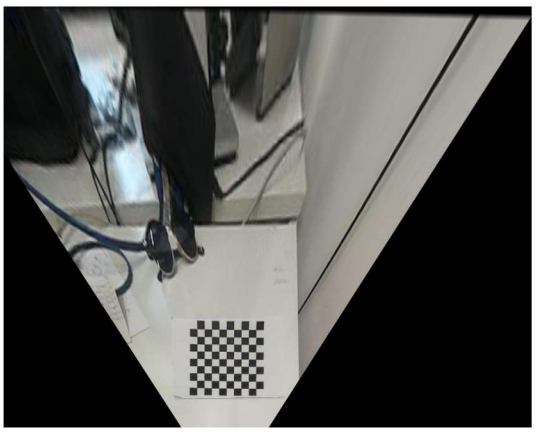
(b) Rezultat dobiven aritmetičkim križanjem. Nakon 5 iteracija postiglo se poboljšanje minimalne vrijednosti funkcije cilja za 31.8%.



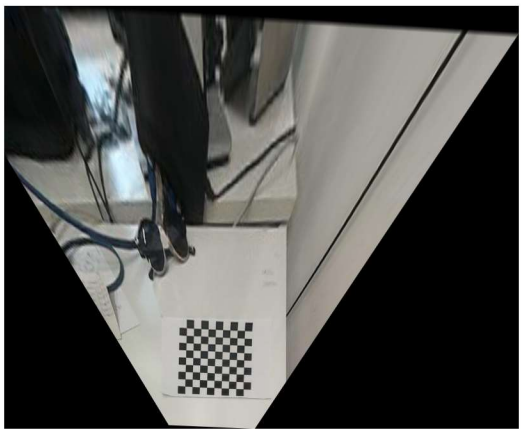
(c) Rezultat dobiven uniformnim križanjem. Ovim križanjem nije postignuto poboljšanje u odnosu na početnu najbolju homografiju.



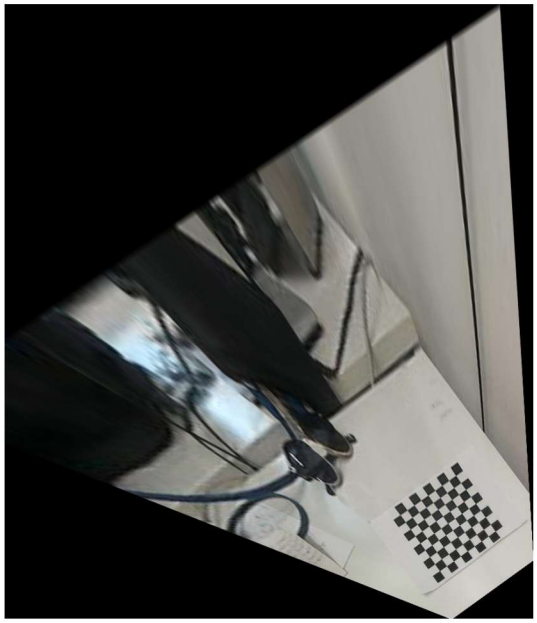
(d) Rezultat dobiven križanjem s alternirajućim pozicijama. Nakon 5 iteracija postiglo se poboljšanje minimalne vrijednosti funkcije cilja za 50.4%.



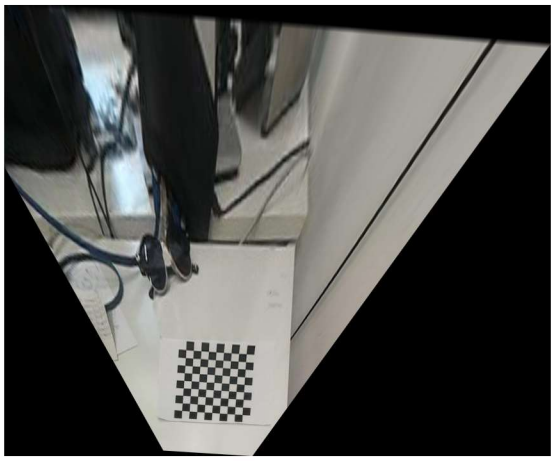
(e) Rezultat dobiven flat križanjem. Nakon 5 iteracija postiglo se poboljšanje minimalne vrijednosti funkcije cilja za 93.3%.



(f) Rezultat dobiven lokalnim križanjem. Nakon 5 iteracija postiglo se poboljšanje minimalne vrijednosti funkcije cilja za 31.8%.



(g) Rezultat dobiven BLX-alpha križanjem. Nakon 5 iteracija postiglo se poboljšanje minimalne vrijednosti funkcije cilja za 81.8%. Kao i kod aritmetičkog križanja, ovdje je sliku još potrebno rotirati. Rotacija slike ne utječe na vrijednost funkcije cilja jer se samo uspoređuju duljine stranica uzorka.



(h) Rezultat dobiven križanjem zamjenom više elemenata. Nakon 5 iteracija postiglo se poboljšanje minimalne vrijednosti funkcije cilja za 7%.

Slika 6.3: Rezultati dobiveni različitim križanjima.

Broj iteracije	MIN	AVG	MED	MAX	Broj jedinki stvorenih aritmetičkim križanjem	Broj jedinki stvorenih flat križanjem	Broj jedinki stvorenih lokalnim križanjem	Broj jedinki stvorenih BLX – alpha križanjem	Broj jedinki stvorenih križanjem zamjenom jednog elementa	Broj jedinki stvorenih križanjem zamjenom više elemenata
0	53.4797	7546.23	10042.5	10052						
1	28.6792	2603.7	169.031	10045.3	2			1	1	2
2	10.7333	149.726	58.751	716.276	5	1				
3	3.45028	96.6608	28.6792	615.741	2	2	1	1		
4	3.45028	62.9797	19.1208	378.643		2	1	1		2
5	3.45028	6.84649	3.7612	18.8316			2	2		2

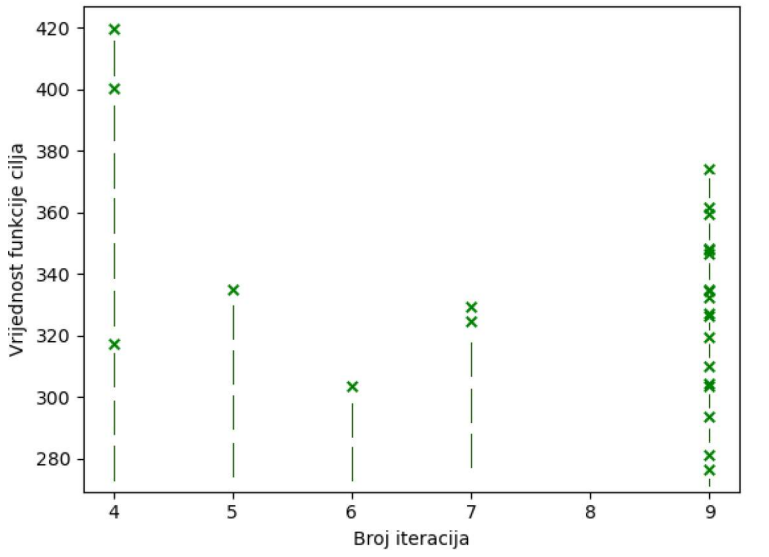
Tablica 6.2: Rezultati dobiveni kombinacijom 6 križanja, mutacije slučajnog broja elemenata i 3-turnira nad veličinom populacije 9 čiji je skup slika 6.1.

Broj iteracija	MIN	AVG	MED	MAX
0	578.309	11140	10602.7	30660.6
1	477.358	7045.04	10434.6	21320.8
2	477.358	4524.53	1607.29	10625.3
3	395.919	1937.32	887.404	10485.4
4	311.489	1275.31	637.843	10419.7
5	311.489	607.426	442.621	11368.84
6	306.234	447.425	418.6	812.639
7	285.777	362.934	335.555	591.267
8	276.55	314.94	312.799	388.43
9	276.55	302.669	305.656	323.304

Tablica 6.3: Vrijednosti funkcije cilja jedinki u populaciji tijekom 9 iteracija u jednom od 26 pokretanja algoritma nad populacijom veličine 19.

Broj iteracija	MIN	AVG	MED	MAX
0	578.309	11140	10602.7	30660.6
1	477.358	7055.54	10479.6	21320.8
2	474.699	5066	2106.73	12072.9
3	368.518	2787.09	651.204	10607.7
4	368.518	2153.06	494.041	10399
5	368.518	618.134	461.219	1625.72
6	368.518	463.796	450.498	581.474
7	368.518	433.847	437.806	478.949
8	365.374	416.779	424.503	451.964
9	334.84	396.384	396.934	437.368

Tablica 6.4: Vrijednosti funkcije cilja jedinki u populaciji tijekom 9 iteracija u jednom od 26 pokretanja algoritma nad populacijom veličine 19.



Slika 6.4: Vrijednosti funkcije cilja najboljih rješenja u 26 pokretanja algoritma na skupu slika veličine 19 s različitim brojem iteracija.

(a) Originalna slika

(b) Slika u ptičjoj perspektivi

Slika 6.5: Prijelaz u ptičju perspektivu homografijom koja je pronađena nakon 4 iteracije algoritma.

(a) Originalna slika

(b) Slika u ptičjoj perspektivi

Slika 6.6: Rezultat dobiven najtočnijom dobivenom homografijom nakon 9 iteracija.

Literatura

[1] P. Coussy, A. Rossi *Metaheuristics*, 2010.

[2] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard *Metaheuristics for Hard Optimization*, 2006.

[3] M.A. Fischler, R.C. Bolles *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.*, Communications of the ACM **24**(1981), 381–395.

[4] R. Hartley, A. Zisserman *Multiple View Geometry in Computer Vision*, 2004.

[5] H. Kano, K. Asari, Y.I. Ishii, H. Hongo *Precise Top View Image Generation without Global Metric Information*, IEICE Transactions on Information and Systems **E91-D**(2008), 1893–1898.

[6] S. Picek, D. Jakobović, M. Golub *On the Recombination Operator in the Real-Coded Genetic Algorithms*, 2013 IEEE Congress on Evolutionary Computation (2013), 3103-3110.

[7] D. Simon *Evolutionary Optimization Algorithms*, 2013.

[8] G. Wang, Q.M.J. Wu *Guide to Three Dimensional Structure and Motion Factorization*, 2011.

[9] Z. Zhang *A Flexible New Technique for Camera Calibration*, Pattern Analysis and Machine Intelligence, IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(2000), 1330-1334.

[10] Edmund Optics, Aberration. Dostupno na https://www.edmundoptics.com/knowledge-center/application-notes/imaging/how-aberrations-affect-machine-vision-lenses/

[11] Edmund Optics, Distortion. Dostupno na https://www.edmundoptics.com/knowledge-center/application-notes/imaging/distortion/0

[12] LearnOpenCV, Understanding Lens Distortion. Dostupno na https://learnopencv.com/understanding-lens-distortion/

[13] Projektivna geometrija. Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2021. Dostupno na https://www.enciklopedija.hr/natuknica.aspx?ID=50580

Sažetak

U ovom radu opisana je metoda za izračun homografije koja prebacuje slike u ptičju perspektivu te je optimizirana matrica homografije za cijelu scenu vidljivu kamerom. Prvo je potrebno napraviti skup slika uzorka šahovske ploče na različitim područjima scene. Ako je prisutna radijalna distorzija, potrebno ju je ukloniti sa svake slike. Zatim se za svaku sliku pronalazi homografija koja uzorak na njoj prebacuje u ptičju perspektivu. Dobivene homografije čine inicijalnu populaciju genetičkog algoritma koji se koristi za pronalazak optimalne homografije za prebacivanje cijele scene u ptičju perspektivu.

Ključne riječi

genetički algoritam, evolucijski algoritam, metaheuristika, populacija, križanje, mutacija, generacija, homografija, distorzija

Warping an image to bird's eye view using genetic algorithm

Summary

This paper describes a method for computing homography that will warp images to a bird's eye view and optimises the homography matrix for the entire scene visible by the camera. First, a set of images is created that contains chessboard pattern on different areas of the scene. If radial distortion is present, it must be removed from each of the images. Next, for each image, a homography is found that transfers the pattern on the image to the bird's eye view. These homographies form the initial population for the genetic algorithm used to find the optimal homography for transferring the entire scene to the bird's eye view.

Keywords

genetic algorithm, evolutionary algorithm, metaheuristic, population, crossover, mutation, generation, homography, distortion

Životopis

Zovem se Branka Miholek i rođena sam 10. listopada 1998. godine u Osijeku. Pohađala sam Osnovnu školu Jagode Truhelke u Osijeku te sam nakon nje upisala I. gimnaziju Osijek. Zatim sam 2017. godine upisala preddiplomski studij Matematike i računarstva na Odjelu za matematiku u Osijeku. Završila sam ga 2020. godine s temom završnog rada "Spajanje slika i videa u panoramu". Iste godine upisala sam diplomski studij na Odjelu za matematiku u Osijeku, smjer Matematika i računarstvo. Tijekom diplomskog studija radila sam u tvrtki Orqa u Osijeku kao software developer.