# Vaja 5: Epipolarna geometrija in triangulacija

Laboratorijske Vaje

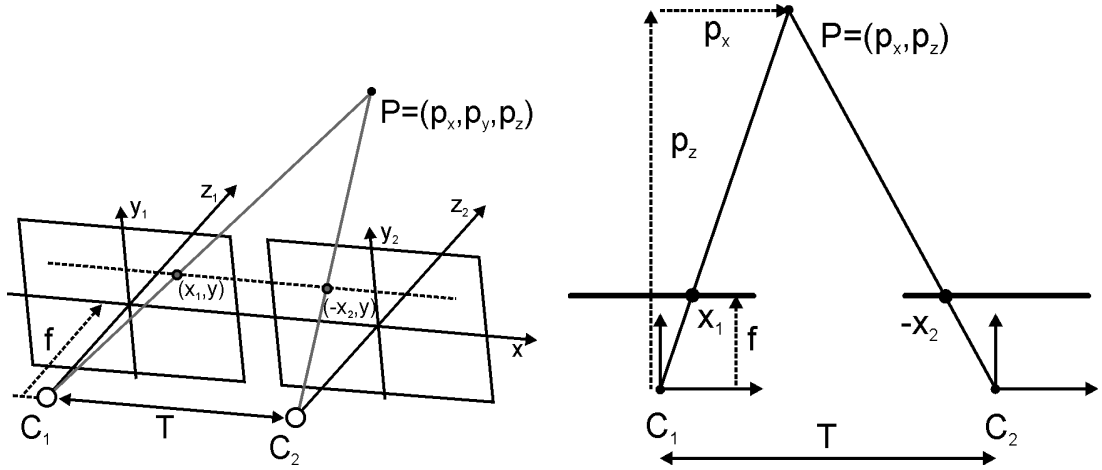
20. december 2011

**Povzetek**

V tej vaji si bomo pogledali del teorije epipolarne geometrije [1] (poglavje 10.1) in robustnega ocenjevanja [1] (str. 346). V prvem delu se bomo posvetili teoretičnim nalogam nato teorijo implementirali.

## 1 Dispariteta

V tej nalogi se bomo posvetili osnovam izračuna disparitete dveh kamer. Naše analize se bodo nanašale na najenostavnejšo postavitev stereo sistema kamer, ker imamo identični kameri postavljeni tako, da imata vzporedni optični osi, njuni slikovni ravnini (npr., CCD tipali) pa ležita v isti ravnini (Slika 1a).




(a) (b)

Slika 1: Leva slika prikazuje najpreprostejšo postavitev stereo sistema z dvemi enakimi vzporednimi kamerami. Desna slika prikazuje geometrijske relacije med preslikavo točke $\mathbf{p}$ v 3D prostoru na os $x$ v slikovnih ravninah kamer.

V Sliki 1(b) vidimo, da lahko s pomočjo podobnih trikotnikov zapišemo naslednji relaciji:

$$\frac{x_1}{f} = \frac{p_x}{p_z} \quad , \quad \frac{-x_2}{f} = \frac{T - p_x}{p_z}. \tag{1}$$

(a) S pomočjo enačb (1) izpeljite izraz za *dispariteto*, ki je definirana kot $d = x_1 - x_2$. Kakšna je odvisnost med oddaljenostjo objekta (točke $\mathbf{p}$ v Sliki 1) od kamere in dispariteto $d$? Kaj se dogaja z dispariteto, če je objekt blizu kamer, in kaj če je zelo daleč? Da lažje odgovorite na vprašanje, napišite v Octave/Matlab program, ki bo za različne vrednosti $p_z$ izračunal dispariteto. Graf izrišite na zaslon in označite

osi ter merske enote. Za vhodne podatke vzemite: goriščno razdaljo $f = 2.5mm$ in razmik med centroma kamer (angl. baseline) $T = 12cm$. To vajo rešite na list, ki ga prinesete na zagovor, programsko rešitev pa napišite v `vaja2_naloga1a.m`.

(b) Da dobite boljši občutek koliko razdalja do objekta vpliva na dispariteto v *slikovnih elementih* senzorja si poglejmo realen primer, kjer bomo za podatke vzeli specifikacije komercialne stereo kamere Bumblebee2 podjetja PointGray: $f = 2.5mm$, $T = 12cm$, slikovni senzor je velikosti $648 \times 488$ slikovnih elementov, slikovni elementi so kvadratni širina slikovnega elementa je $7.4\mu m$ in predpostavimo, da med slikovnimi elementi ni praznega prostora. Denimo, da opazujemo nek objekt s takim sistemom in predpostavimo, da sta kameri res vzporedni in enaki. Ugotovimo, da se center objekta v levi kameri vzdolž osi $x$ nahaja na 550-tem slikovnem elementu, medtem ko ga v desni kameri vidimo na 300-tem slikovnem elementu. Koliko daleč od kamere (v metrih) se objekt nahaja? Koliko daleč se objekt nahaja, če njegov center v desni kameri vidimo na 540-tem slikovnem elementu? To nalogo rešite na list papirja in rešitev skupaj s postopkom prinesite na zagovor.

# 2 Fundamentalna matrika, epipoli in epipolarne premice

V prejšnji nalogi smo si pogledali poseben primer, kjer so bile slikovne ravnine kamer poravnane. Videli smo, da se v takih primerih 3D točka preslika na isto koordinato na osi $y$ v obeh kamerah, razlika v projekciji vzdolž osi $x$ pa nam pove koliko je objekt oddaljen od kamer. Taka postavitev nam zagotavlja, da so *epipolarne črte v kamerah vzporedne z vrsticami v senzorju* kamere, kar močno poenostavi iskanje korespondenc (npr., točka v levi kameri in točka v desni kameri, kjer obe pripadata isti 3D točki na opazovanem objektu). V splošnem pa *epipolarne črte niso poravnane* z vrsticami in moramo najprej izračunati *fundamentalno matriko* med dvema kamerama.

V tej nalogi boste implementirali preprosto različico *osem točkovnega algoritma* [1], s katerim boste izračunali fundamentalno matriko med dvema slikama in si vizualizirali epipolarne črte. Za boljše razumevanje, najprej ponovimo najpomembnejše dele teorije, ki jo boste potrebovali. Predpostavimo, da imamo znan spisek popolnih korespondenc točk v levi $\mathbf{x} = [u, v, 1]^T$ in desni $\mathbf{x}' = [u', v', 1]^T$ sliki v homogenih koordinatah[1]. Pravilo fundamentalne matrike pravi, da mora vsaka taka korespondenca ustrezati enačbi

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0 \quad ; \quad \mathbf{F} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}, \tag{2}$$

Kjer je $\mathbf{F}$ fundamentalna matrika. Podobno kot smo to storili pri izpeljavi enačbe za ocenjevanje homografije (Vaja 4), lahko zapišemo za en par koresponenčnih točk kot

$$\begin{bmatrix} uu' & uv' & u & vu' & vv' & v & u' & v' & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0. \tag{3}$$

[1]Spomnimo se, da točko zapišemo v homogenih koordinatah tako, da ji dodamo še eno koordinato z vrednostjo 1 na zadnje mesto.

Če sedaj zložimo $N \geq 8$ takih enačb v matriko $\mathbf{A}$, dobimo matrično enačbo:

$$\mathbf{Af} = \mathbf{0}$$

$$\begin{bmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_Nu'_N & u_Nv'_N & u_N & v_Nu'_N & v_Nv'_N & v_N & u'_N & v'_N & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ \vdots \\ F_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (4)$$

Podobno kot smo to storili pri ocenjevanju homografije v Vaji 3, lahko zgornjo enačbo rešimo v smislu najmanjših kvadratov z metodo dekompozicije po singularnih vrednostih (SVD). Matriko $\mathbf{A}$ razcepimo po SVDju na $\mathbf{A} = \mathbf{UDV}^T$. Rešitev po najmanjših kvadratih ustreza lastnemu vektorju $\mathbf{v}_n$ z najmanjšo lastno vrednostjo – zadnjemu stolpcu matrike[2] $\mathbf{V}$.

Spomnimo se, da je **epipol** kamere tista točka, v kateri se sekajo vse epipolarne črte te kamere. To zahteva, da je rank matrike $\mathbf{F}$ enak 2, vendar v primeru šumnih podatkov matrika ocenjena po zgornji enačbi ne bo tega ranka. To pomeni, da se epipolarne črte ne bodo sekale v eni točki, pač pa se bodo presečišča razprostirala v neki majhni okolici epipola. Zato moramo rank matrike naknadno omejiti na 2. To storimo tako, da apliciramo SVD razcep na matriko $\mathbf{F} = \mathbf{UDV}^T$, njeno najmanjšo lastno vrednost ($D_{33}$ v matriki $\mathbf{D}$) postavimo na nič in matriko nazaj rekonstruiramo (zmnožimo $\mathbf{UDV}^T$). Tako rekonstruirana matrika bo sedaj zadovoljila omejitev rank = 2 in lahko izračunamo dva epipola za katera velja

$$\mathbf{Fe}_1 = 0 \quad \text{in} \quad \mathbf{F}^T\mathbf{e}_2 = 0, \quad (5)$$

tako, da ponovno razcepimo rekonstruirano $\mathbf{F}$ in izračunamo *levi* in *desni*[3] lastni vektor matrike $\mathbf{F}$,

$$\mathbf{e}_1 = \begin{bmatrix} V_{13} & V_{23} & V_{33} \end{bmatrix} / V_{33} \text{ , } \mathbf{e}_1 = \begin{bmatrix} U_{13} & U_{23} & U_{33} \end{bmatrix} / U_{33}, \quad (6)$$

da dobimo epipole v normaliziranih homogenih koordinatah.

Preprost osem-točkovni algoritem za izračun fundamentalne matrike in epipolov strnemo v:

- Konstruiraj matriko $\mathbf{A}$ kot v (4), razčleni s SVD $\mathbf{A} = \mathbf{UDV}^T$, in zadnji lastni vektor $\mathbf{v}_9 = \mathbf{V}(:, 9)$ preoblikuj v $3 \times 3$ matriko $\mathbf{F}_t$ (e.g., `reshape(v9,3,3)'` – bodite pozorni na znak za transponirano, premislite če ga rabite v vašem primeru, in pazite, da bodo elementi matrike $\mathbf{F}$ pravilno urejeni po reshape!).
- Razčleni $\mathbf{F}_t = \mathbf{UDV}^T$, postavi najmanjšo lastno vrednost na 0, t.j., $\mathbf{D}(3, 3) = 0$, in rekonstruiraj $\mathbf{F} = \mathbf{UDV}^T$ s spremenjeno $\mathbf{D}$.
- Oba epipola sta `e_1 = V(:,3)/V(3,3)` in `e_2 = U(:,3)/U(3,3)`.

Ko imamo enkrat znano fundamentalno matriko, lahko za poljubno točko $\mathbf{x}_1$ v eni kameri definiramo njeno epipolarno premico v drugi kameri $\mathbf{l}_2 = \mathbf{Fx}_1$ (enako za $\mathbf{x}_2$ v drugi kameri najdemo premico $\mathbf{l}_1 = \mathbf{F}^T\mathbf{x}_2$ v prvi kameri). Spomnimo se, da je premica v projektivni geometriji definirana kot en sam 3D vektor. Za boljšo predstavo, si zapišimo premico $\mathbf{l}_1 = [a, b, c]$ s klasično Evklidsko formulo

$$ax + by + c = 0, \quad (7)$$

in vanjo vstavimo koordinate točke $\mathbf{x}_1 = [X, Y, W]$, iz česar dobimo

$$aX + bY + cW = 0 \quad (8)$$

$$\mathbf{l}_1^T\mathbf{x}_1 = \mathbf{x}_1^T\mathbf{l}_1 = 0. \quad (9)$$

[2]Ta rešitev je v stolpičini obliki kot v enačbi (3), ki jo moramo preurediti v obliko v enačbi 2

[3]Pozor: Termina levi in desni lastni vektor sta matematična termina za dva različna tipa lastnih vektorjev in nimajo nobene povezave z levo in desno sliko v kamerah.

Interpretacija parametrov je sledeča: $-a/b$ je naklon premice, $-c/a$ ter $-c/b$ pa sta presečišči na osi $x$ in $y$.

(a) Rešite naslednjo preprosto računsko nalogo na papir. Imamo dve kameri, za kateri smo izračunali po zgorjih postopkih fundamentalno matriko iz leve kamere v desno

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 0.002 \\ 0 & 0 & -0.012 \\ -0.001 & 0.011 & -0.085 \end{bmatrix}. \tag{10}$$

***Vprašanje:** Izračunajte Evklidsko enačbo epipolarne premice v desni kameri, ki ustreza točki, ki se nahaja v levi kameri na vrstici row = 120 in stolpcu col = 300. Upoštevajte, da morate točko naprej zapisati v homogenih koordinatah $\mathbf{x} = [\text{col}, \text{row}, 1]^T$. Sedaj izračunajte epipolarno premico za še eno točko v levi kameri na koordinatah row = 170 in stolpcu col = 300. *Neobvezni del: Izračunajte presečišče teh dveh premic. V kateri pomembni točki se ti premici sekata? Kako imenujemo to presečišče?* To nalogo rešite na papir in s postopkom vred prinesite na zagovor.

## 2.1 Izračun Fundamentalne matrike

V tej nalogi boste implementirali in preizkusili izračun fundamentalne matrike, ter si v praksi pogledali del teorije epipolarne geometrije. S spletne strani predmeta si prenesite in razpakirajte datoteko `vaja5.zip` v mapo `vaja5/`. V paketu se nahaja prototip funkcije `data/vaja5_naloga2.m` v katero boste pisali kodo. Koda vam z diska naloži dve sliki objekta, ki smo ga posneli z levo in desno kamero, prav tako vam nariše deset korespondenčnih točk med obema slikama.

(a) Implementirajte funkcijo `fundmatrixest.m`, ki za parametre vzame nabor (vsaj osmih) korespondenčnih točk med slikami in s pomočjo osem-točkovnega algoritma izračuna fundamentalno matriko ter epipole.

```
function [F e1 e2] = fundmatrixest(x1, x2)
% Vhod:
% x1,x2 : 3xN matrika N homogenih točk v 2D
% Izhod:
% F : 3x3 fundamentalna matrika, tako da velja x2'*F*x1 = 0
% e1 : epipol v prvi sliki, tako da velja F*e1 = 0
% e2 : epipol v drugi sliki, tako da velja F'*e2 = 0
...
```

(b) Zaradi numerične stabilnosti osem-točkovnega algoritma v praksi ne izvajamo direktno na korespondenčnih točkah, pač pa jih najprej normaliziramo tako, da so centrirane, njihova povprečna razdalja do centroida pa je $\sqrt{2}$. Za nabor točk $\mathbf{x}_1$ z ene kamere to dosežemo tako, da jih transformiramo s transformacijsko matriko $\mathbf{T}_1$. Preučite spodnjo funkcijo kako izračunamo transformacijo.

```
function [newpts, T] = normalize2dpts(pts)
 % pts ... 2xN matrika točk s slike
 % newpts ... 3XN matrika točk (homogene koordinate)
 pts = pts(1:2,:) ; Mu = mean(pts,2) ;
 pt = pts - repmat(Mu,1,size(pts,2)) ; mean_dist = mean(sqrt(sum(pt.^2,1))) ;
 scale = sqrt(2)/mean_dist ;
 T = [scale, 0, -Mu(1)*scale ;...
      0 scale -Mu(2)*scale; 0 0 1] ;
 newpts = zeros(3,size(pts,2)) ;
 for i = 1 : size(pts,2)
```

```
  newpts(:,i) = T*[pts(:,i);1] ;
end
```

(c) Sedaj implementirajte `fundmatrixest_norm.m`, ki najprej *normalizira* vhodne točke z leve kamere (dobimo transformirane točke in transformacijo $\mathbf{T}_1$), nato normalizira vhodne točke z desne kamere (dobimo transformirane točke in transformacijo $\mathbf{T}_2$). Na transformiranih točkah izračuna fundamentalno matriko $\hat{\mathbf{F}}$, jo pretransformira v originalni prostor po enačbi $\mathbf{F} = \mathbf{T}_2^T \hat{\mathbf{F}} \mathbf{T}_1$ in izračuna epipole po enačbi (6).

```
function [F e1 e2] = fundmatrixest_norm(x1, x2)
% Vhod:
% x1,x2 : 3xN matrika N homogenih točk v 2D
% Izhod:
% F : 3x3 fundamentalna matrika, tako da velja x2'*F*x1 = 0
% e1 : epipol v prvi sliki, tako da velja F*e1 = 0
% e2 : epipol v drugi sliki, tako da velja F'*e2 = 0
...
```

(d) Preizkusite delovanje vašega algoritma: Na desetih točkah, ki jih naložite z datoteke `data/demo_points_house.txt` izračunajte fundamentalno matriko $\mathbf{F}$ in za točko $p = [85, 233]^T$ v levi sliki izračunajte epipolarno premico v desni sliki. Premico lahko izrišete s priloženo funkcijo `function prikazipremico( l, w, h, col )`. *Po epipolarni geometriji bi morala korespondenčna točka za izbrano točko v levi sliki ležati na njeni epipolarni premici v desni sliki. (Korespondenčna točka v desni sliki se nahaja na koordinatah* $[67, 219]^T$*.)* Ali to drži za vaše točke? Za referenco je spodaj napisana pravilna fundamentalna matrika:

$$F = \begin{bmatrix} 2.2714 \cdot 10^{-5} & 1.2959 \cdot 10^{-4} & -0.0346 \\ -1.1222 \cdot 10^{-4} & -6.9218 \cdot 10^{-6} & 0.0293 \\ 0.0219 & -0.0267 & 0.5611 \end{bmatrix} \tag{11}$$

Nalogo napišite v skripto `vaja5_naloga2_1d.m`.

(e) Kot kvantitativno mero kvalitete ocenjene fundamentalne matrike uporabljamo reprojekcijsko napako. V vašem primeru je to povprečna oddaljenost točk od njihovih pripadajočih epipolarnih premic. Uporabite spodnjo funkcijo `d = reprojOdstopanje(x1, x2, F)`, ki prejme za parameter točko v prvi sliki, njeno korespondenco v drugi sliki in fundamentalno matriko, vrne pa oddaljenost druge točke od njene epipolarne premice (na list si napišite kako izračunamo oddaljenost točke od premice). Napišite novo funkcijo za izračun *simetrične reprojekcijske napake* `d = reprojOdstopanjeSim(x1, x2, F)`. Ta funkcija naj izračuna koliko je $x1$ oddaljena od pripadajoče epipolarne premice, ki jo izračunamo iz $x2$, in koliko je $x2$ oddaljena od njene epipolarne premice, ki jo izračunamo iz $x1$. Funkcija naj *vrne povprečje* teh dveh razdalj. Napišite skripto `vaja5_naloga2_1e.m`, s katero boste preverili ali vaša koda deluje pravilno. Skripta naj izvede dva testa: (i) Najprej izračuna simetrično reprojekcijsko napako med točko $p_1 = [85, 233]^T$ v levi sliki in točko $p_2 = [67, 219]^T$ v desni sliki, pri tem pa uporabite fundamentalno matriko, ki ste jo izračunali v prejšnji točki (če ste pravilno implementirali vse dele naloge, mora biti ta napaka okoli 0.15 slikovnih elementov). (ii) Naloži točke oz datoteke `data/demo_points_house.txt`, izračuna povprečje simetričnih reprojekcijskih napak vseh točk in to vrednost izpiše na ekran. Če deluje vaša koda pravilno, je za pričakovati napako okoli 0.33 slikovnih elementov.

```
function d = reprojOdstopanje(x1, x2, F)
 % izračunaj epipolarno premico za x1
 l2 = F*x1 ;
 % izračunaj razdaljo x2 do l2
 d = abs(sum(l2.*x2)) / sqrt(l2(1).^2+l2(2).^2 ) ;
```

## 2.2 RANSAC

V praksi avtomatski algoritmi za iskanje korespondenc nikoli ne vrnejo popolnih ujemanj. Lokacije korespondenčnih točk so pošumljene, prav tako pa nabor korespondenc lahko vsebuje popolnoma napačna ujemanja (ven-ležeče elemente) – to ste že opazili v Vaji 4. Kot ste slišali na predavanjih, v takih primerih uporabljamo algoritem RANSAC. Za vaš primer je algoritem skiciran po spodnjih korakih:

- Naključno izberite minimalno število simetričnih ujemanj, ki jih potrebujete za izračun vašega modela (v primeru ocenjevanja fundamentalne matrike 8).
- Izračunajte fundamentalno matriko s pomočjo izbranega nabora točk.
- Določite notri-ležeče elemente, ki ustrezajo fundamentalni matriki (t.j., simetrična ujemanja, katerih reprojekcijska napaka je manjša od izbrane pragovne vrednosti).
- Če je odstotek notri-ležečih elementov dovolj velik ($\geq m$) na njih ponovno izračunajte fundamentalno matriko, tokrat z najmanjšimi kvadrati (SVD), in končajte postopek.
- Sicer ponavljajte postopek $k$ iteracij.

Vrednost parametra $k$ je določena z lastnostmi naših podatkov, te pa določimo s poznavanjem problema, ki ga rešujemo. Na primer. Recimo, da vemo, da se v naših aplikacijah vedno pojavi med vsemi ujemanji vsaj $w$ procentov pravih ujemanj. Najmanjše število ujemanj za oceno fundamentalne matrike $\mathbf{F}$, vemo, da je $n = 8$. Sedaj lahko izračunamo verjetnost uspeha ocene $\mathbf{F}$ – torej, verjetnost, da bo vseh $n$ izbranih točk notri-ležečih je $w^n$. Verjetnost, da to ne drži, pa je $1-w^n$. Verjetnost, da nam v $k$-tih poskusih niti enkrat ne uspe izbrati nabora notri-ležečih točk je $p_{\text{neuspeh}} = (1-w^n)^k$. *V praksi zato izberemo $k$ tako visok, da je verjetnost neuspeha $p_{\text{neuspeh}}$ dovolj majhna.*

Sedaj boste implementirali vse potrebne korake RANSACa za robustno ocenjevanje matrike $\mathbf{F}$ z naborom korespondenčnih točk, ki vsebuje ven-ležeče elemente.

(a) Implementirajte funkcijo `get_inliers()`, ki vzame za vhod ocenjeno fundamentalno matriko in nabor vseh korespondenčnih točk, vrne pa podmnožico korespondenc, ki so notri-ležeči elementi pod trenutno fundamentalno matriko. Dve točki, $\mathbf{x}$ in $\mathbf{x}'$, sta notri-ležeči, če je razdalja med $\mathbf{x}$ in epipolarno črto $\mathbf{F}^T\mathbf{x}'$, kakor tudi razdalja v drugo smer, manjša od predpisanega $\varepsilon$.

```
function [x1in x2in] = get_inliers(F, x1, x2, eps)
 ...
```

(b) Implementirajte funkcijo `ransac_fundmatrix`, ki izvede algoritem RANSAC za ocenjevanje fundamentalne matrike s pomočjo normaliziranega osem-točkovnega algoritma. Za naključno izbiro točk lahko uporabite funkcijo `randperm`, ki vam vrne naključno permutacijo zaporednih indeksov. Na primer, če želite deset naključnih številk od 1 do 100, potem poženite `a=randperm(100)`, in izberite `a(1:10)`. Implementirajte preprosto verzijo RANSACa, ki teče $k$ iteracij in vrne rešitev z največjim številom notri-ležečih elementov.

```
function [F, e1, e2, x1, x2] = ransac_fundmatrix(x1, x2, eps, k)
 % Vhod:
 % x1,x2 : 3xN matrika N homogenih točk v 2D
 % eps : pragovna vrednost za notri-ležeče elemente
 % k : število iteracij
 % Izhod:
 % F :  3x3 fundamentalna matrika, definirana kot x2'*F*x1 = 0
 % e1 : Epipol v sliki 1, tako da velja F*e1 = 0
 % e2 : Epipol v sliki 2, tako da velja F'*e2 = 0
 % x1,x2 : 3xNi matrika Ni homogenih notri-ležečih točk
 ...
```

(c) Aplicirajte vašo implementacijo RANSACa s parametrom $\varepsilon = 5$ in $k = 100$ na korespondence, ki jih dobite v datoteki `data/house_matches.txt`. Za vizualno pregledovanje uspešnosti ocene matrike $\mathbf{F}$ si izberite točko v levi sliki in prikažite njeno epipolarno premico v desni sliki (`function prikazipremico( l, w, h, col )`). Preverite ali gre ta premica skozi korespondenčno točko v desni sliki. Lahko si izrišete epipolarne premice za več točk. Nastavite parametra $\varepsilon$ in $k$ za najboljše delovanje – od delovanja je odvisna ocena te naloge. Izrišite si v eno sliko vse potencialne ujemajoče pare točk pred uporabo RANSAC (`data/house_matches.txt`), nato pa v drugo sliko vse pare točk, ki jih RANSAC izbere za notri-ležeče. Ali je kakšna razlika med tema dvema množicama? Rešitev te naloge pišite v skripto `vaja5_naloga2_2c.m`.

(d) Sedaj lahko poskusimo avtomatizirati postopek ocenjevanja fundamentalne matrike in parov korespondenčnih točk v obeh slikah. Rešitev te naloge pišite v skripto `vaja5_2_2d.m`. Korespondenčne točke detektirajte z vašo funkcijo `M=najdi_vsa_ujemanja(Slika1,Slika2)`, ki ste jo implementirali v Vaji 4. Ta funkcija uporablja Harrisov ali Hessov detektor (ali oba?), mag/lap opisnik in simetrično primerjavo s Hellingerjevo razdaljo. Namesto, da bi kopirali vso kodo z vaje 4 v vajo 5, raje dodajte na začetku vaše skripte naslednjo vrstico:

```
newPath = '../Vaja4/' ; rmpath(newPath) ; addpath(newPath) ;
```

Ta vrstica vam bo dodala pot na vajo 4 za trenutno uporabo skripte. Na ujemanjih, ki jih dobite z detektorjem, boste zagnali vašo implementacijo RANSACa. Kot rezultat izrišite eno sliko z vsemi ujemajočimi točkami pred RANSACom in eno sliko z vsemi notraj-ležečimi točkami, ki jih RANSAC poišče. Kolikšen odstotek parov ujemajočih točk je RANSAC sprejel, in kolikšna je reprojekcijska napaka? Nastavite parametre za kar najboljše delovanje algoritma. Rešitev pišite v skripto `vaja5_naloga2_2d.m`.

# 3 Triangulacija

Kot zadnji korak želimo rekonstruirati ujemanja med slikama v 3D. Brez predhodne kalibracije to seveda ni mogoče. Za vsako kamero zato lahko najdete kalibracijsko matriko v datotekah `data/house1_camera.txt` in `data/house2_camera.txt`. Kalibracijski matriki sta velikosti $3 \times 4$ in sta shranjeni v tekstni obliki. Matriko preberete z ukazom `P1=load('data/house1_camera.txt')`.

Za triangulacijo bomo uporabili linearni algebraični pristop. Če imamo 2D korespondenco med $\mathbf{x}_1$ v levi sliki in $\mathbf{x}_2$ v desni sliki (v homogenih koordinatah), je 3D lokacija pripadajoče točke $\mathbf{X}$ dana z relacijami

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} \tag{12}$$
$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X}. \tag{13}$$

Vemo, da je vektorski produkt med vzporednimi vektorji enak 0, zato pišemo $\mathbf{x}_1 \times \lambda_1 \mathbf{x}_1 = 0$, in dobimo:

$$\mathbf{x}_1 \times \mathbf{P}_1 \mathbf{X} = [\mathbf{x}_{1\times}] \mathbf{P}_1 \mathbf{X} = 0 \tag{14}$$
$$\mathbf{x}_2 \times \mathbf{P}_2 \mathbf{X} = [\mathbf{x}_{2\times}] \mathbf{P}_2 \mathbf{X} = 0, \tag{15}$$

kjer smo uporabili naslednji zapis (strig-simetrično formo), da se znebimo vektorskega produkta:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{x}_{1\times}]\mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \mathbf{b}. \tag{16}$$

Za vsaki 2D točki dobimo dve linearno neodvisni enačbi za tri neznanke. Torej, če zložimo skupaj prvi dve vrstici produkta $[\mathbf{x}_{1\times}]\mathbf{P}_1$ in prvi dve vrstici produkta $[\mathbf{x}_{2\times}]\mathbf{P}_2$ v matriko $\mathbf{A}$, lahko izračunamo srednje kvadratično oceno $\mathbf{X}$ z rešitvijo sistema $\mathbf{AX} = 0$. Kot smo že mnogokrat omenili, je rešitev takega problema lastni vektor matrike $\mathbf{A}$ z najmanjšo lastno vrednostjo. Bodite pozorni, da je rešitev sistema točka $\mathbf{X}$ v homogenih koordinatah (4D). Zato morate to točko še normalizirati tako, da bo njena zadnja koordinata enaka 1. Za spremembo vektorja v strig-simetrično formo uporabite funkcijo `crossForm`.

```
function Cx = crossForm(a)
 Cx = [  0  , -a(3),  a(2) ;...
        a(3),   0  , -a(1) ;...
       -a(2),  a(1),   0 ] ;
```

(a) Implementirajte funkcijo, ki za vhod vzame korespondenčne točke ter kalibracijski matriki kamer, vrne pa triangulirane 3D točke. Za izhodišče lahko uporabite funkcijo `triangulate`, ki je napisana spodaj, rezultat si vizualizirajte s priloženo funkcijo `prikaziTriangulacijo(X)`. Triangulacijo preizkusite na desetih točkah v datoteki `demo/demo_points_house.txt`. Te točke si narišite v levo in desno sliko, nato pa s `plot3()` izrišite še njihovo triangulacijo – triangulacija bo zrcalno preslikana preko ene od osi, ker so tako nastavljene kalibracijske matrike kamer. Rešitev napišite v skripto `vaja4_naloga3a.m`.

```
function X = triangulate( pts_1, pts_2, P_1, P_2)
% vhod:
% pts_1 ... 3xN točk leve kamere v homogenih koordinatah
% pts_2 ... 3xN točk desne kamere v homogenih koordinatah
% P_1 ... 3x4 kalibracijska matrika leve kamere
% P_2 ... 3x4 kalibracijska matrika desne kamere
% izhod :
% X ... 4XN vektor 3D točk v homogenih koordinatah
 X = [] ;
 for i = 1 : size(pts_1,2)
    a1 = crossForm(pts_1(:,i)) ;
    a2 = crossForm(pts_2(:,i)) ;
    c1 = a1*P_1 ;
    c2 = a2*P_2 ;
    A = [ c1(1:2,:); c2(1:2,:) ] ;

    % naredite SVD dekompozicijo matrike A
    % rešitev za točko zapišite v vektor x
    X = [X, x] ;
end
```

(b) **Dodatna naloga (neobvezno) za dodatnih** 10%**:** Sedaj imate vse pripravljeno za popolno avtomatsko preprosto (redko) detekcijo 3D točk na objektu v sceni, ki jo opazujete s kalibriranim sistemom kamer: Detektirajte vse ujemajoče točke in uporabite RANSAC, da odstranite napačne korespondence (`vaja5_2_2d.m`). Tako filtrirane točke triangulirajte z upoštevanjem kalibracijskih matrik in si rezultat prikažite v 3D (`vaja4_naloga3a.m`). Rešitev napišite v skripto `vaja5_naloga3b.m`. Nastavite parametre tako, da bo delovanje najboljše.

# Literatura


[1] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2002.