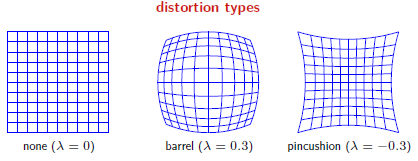
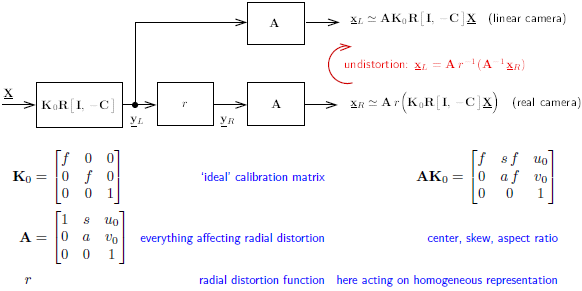
# 16. Kalibrace reálné perspektivní kamery s radiálním zkreslením. Rekonstrukce systému mnoha kamer. Autokalibrace.

1. **Kalibrace reálné perspektivní kamery s radiálním zkreslením**

**Real Camera with Radial Distortion** *(žádné, soudkové a jehelníček)*

****

**Real And Linear Camera Models**

****

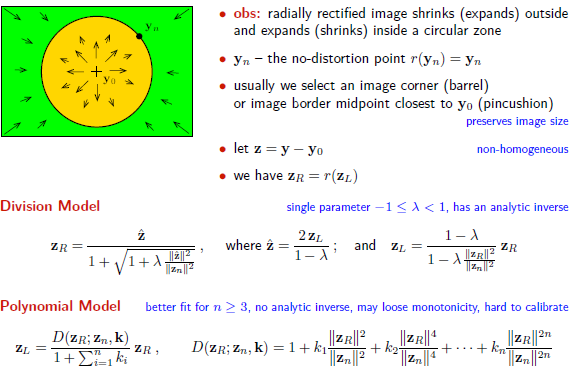
V tom schematu, dole je skutecny obraz (zkazeny), nahore je odzkresleny (lineární), takze linearni kamera je odzkreslena ... K0 **idealni kalibrace,** A je nejaka matice, ktera ovlivnuje nebo kterou ovlivnuje radial distortion

provadi pouze zvetseni - zadne zkoseni, zadne posunuti.. ctvercove pixely, radial lens dis. a elliptic dis. dela ta A, to znamená, že ona to kazí ... zkresluje, takže tu A musíme napravit, odstranit ... v kazdem pripade pri **undistortu** se to nejprve odAckuje, pak s tim udela neco funkce r^-1, a znova se pouzije Acko, coz mi neni moc jasne, kdyz to acko nam to kazi, tak by se asi snazil to A odstranit...

**Radial Distortion Models**

Obrazek mluvi za vse, existuje yn, kde k zadnemu zkresleni nedochazi, to jsou všechny body, na te kruznici

nejvice zkresleni je uprostred a na krajich, kdyz to chceme prevest - **odzkreslit**, tak muzeme pouzit na to nejakej model, treba division, chceme treba ZL, jinak Z = y-y0, tedy vzdalenost od stredu, resp. vektor, chtelo by si to pamatovat ten vzorec, jako pokud das ZR = Zn, tak zlomek = 1 a pak plati ZL = ZR, což je fajn, kdyz ZR = 0, tak je to nula, takze uprostred se s tim taky nic neudela



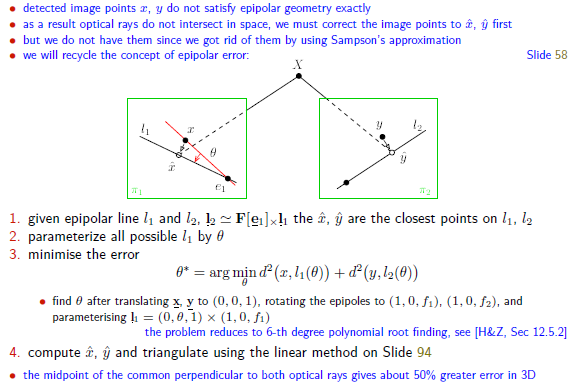
1. **Rekonstrukce systému mnoha kamer**

**3D Structure Computation by Triangulation**

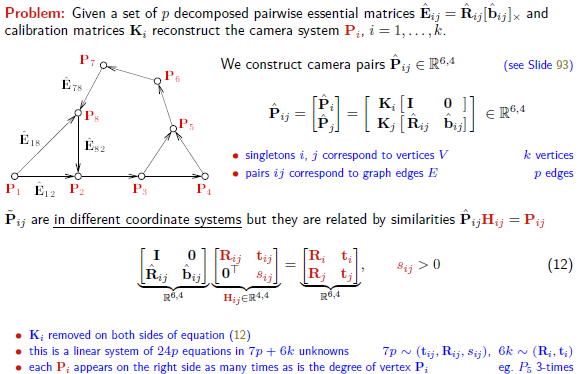
****

Napiseme si rovnice pro projekci bodu X do dvou obrazku**,** tak mame projekce x a y, treti souradnice je p13^T \* X, takze lambda1 = p13^T \* X, takze: x1 \* p13^T \* X = p11^T \* X, totez udelame pro ostatni x2, y1, y2, prevedeme vsechno na jednu stranu, vytkneme X a dostaneme DX = 0**,** pomoci svd vyresime, pak je tam poznamka, ze to **neni invariantni vuci H.**

**Optimal Triangulation**

****

**Reconstruction Camera System**

****

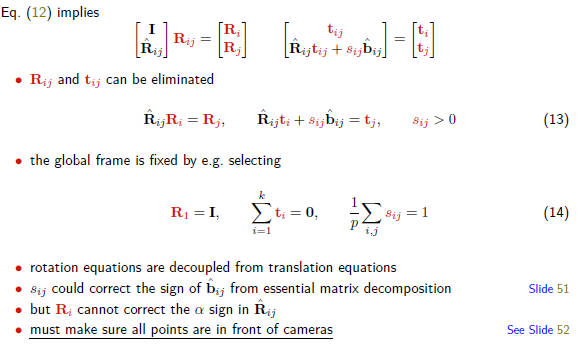
Mame kamery P1 .. Pk a relativni translace mezi nimi E, takze muzeme se posouvat z kamery do kamery pres nejake E, ktere vyjadruje relativni prechod mezi nimi, to je temi sipkami znazorneno, ale je jasne, ze kdyz takhle po sipkach pujdu nekam daleko, tak chyba narusta, takze je potreba to nejak globalne optimalizovat.

Takze pro dvojici kamer Pi' a Pj', **sestavime matici Pij**, tak ze obe matice **zapiseme pod sebe**, tim dostaneme 6x4 matici Pij', Pij jsou ale v **ruznych souradnych systemech**, ale existuje jim odpovidajici **homografie Hij**, ktera to napravi, jinak predtim jsem psal Pij' - tak byly se strechou.

Takze bez strechy (ty se stejnym sourad. systemem) urcime jako Pij = Pij' \* Hij a tomu odpovida ten vzorec pod tim, takze mame dve kamery pod sebou, to je prvni matice, pak homografii, ktera provadi **nejakou rotaci Rij**, **translaci tij** a **scale sij** a vysledkem je dvojice spravnych kamer Ri ti a Rj tj, cervene je oznaceno to co nezname, takhle si napiseme vsechny rovnice ze vsech sipek v grafu, a mame tedy obrovskou soustavu rovnic.

**Presneji:** 24p rovnic o 7p+6k neznamych. Proc 7 a 6? Protoze 7p odpovida tij, Rij a sij - 3 translace + 3 rotace + 1 scale. 6k protoze 3 rotace + 3 translace, no a resime tyhle rovnice.

**Pamatovat:**  
1 kamera, pod ni druha kamera**,** krat homografie, ktera dela **rotaci**, **translaci** a **scale** = spravna kamera a druha spravna kamera a pak to sestavis z toho.

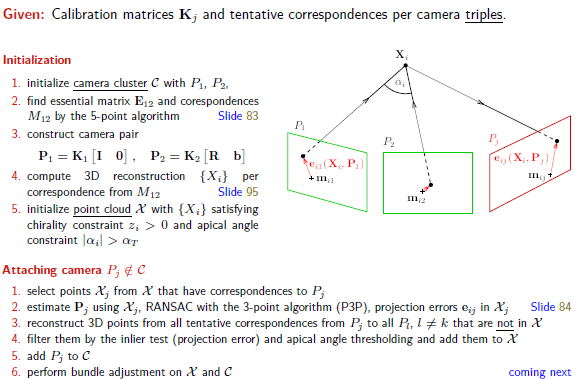
****

Tady to muze vypadat slozite ale v podstate tu rovnici rozkouskovali, ze jednotlive cleny vynasobili samostatne

a dostali:  
**Rij' \* Ri = Rj  
Rij' ti + sij bij' = tj**, takze ve skutecnosti resi pak soustavu tehle zjednodusenych rovnic a pak maji podminky

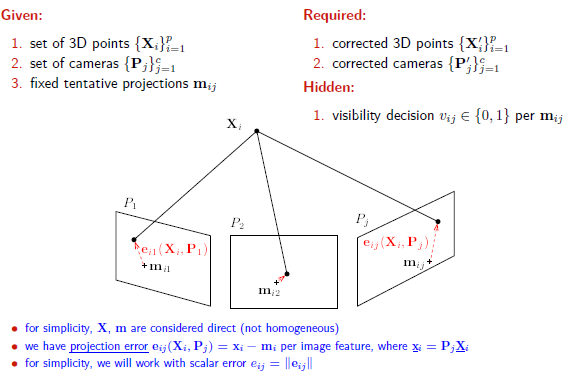
R1 = I - tedy prvni kamera se kouka **defaultnim smere**m do osy z, ostatni se pocitaji podle toho, suma vsech translaci mezi kamerami se musi rovnat nule, tedy kdyz se zacnu posouvat od jedne kamery k druhe tak nakonec skoncim na stejnem miste a nakonec prumerne **sij = 1**, tedy nesmi mi nejaka matice obraz zvetsit, to je scale, tedy někde přiblížím a někde oddálim, prumerne obraz zustane stejne veliky, aby kamery zhruba stejne zvetsovali, nebo aby celkove se scale nezmenil, pak jsou tam dole nejake poznamky a ze je potreba se ujistit, ze body jsou pred kamerou.

**Solving Eg. (13) Stepwise Gluing**



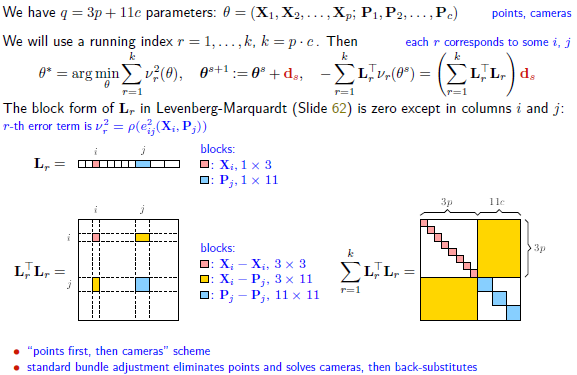
Jako tohle je obecna optimalizace. Stepwise gluing je nejaka aproximace. Jako tady je potreba si zapamatovat jen ten postup.

**Bundle Adjustment** *(Metoda narovnání svazku)*



Mame mnozinu kamer P, mnozinu bodu X a mnozinu projekci m**,** potrebujeme P a X opravit, takze zavedeme funkci eij(Xi, Pj), ktera bod Xi promitne pres Pj a urci chybu od mij.

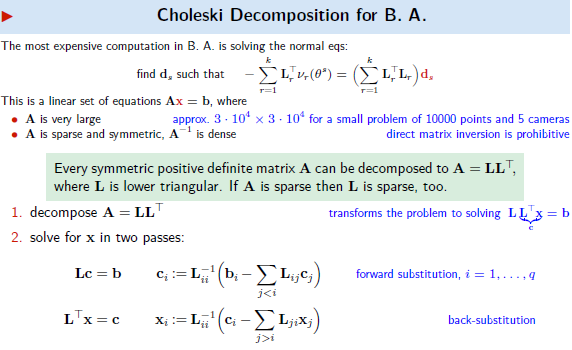
**Sparsity** (řídkost) **in Bundle Adjustment**



Takze mame mnozinu (vektor) Theta = (X1,X2,....., P1, .... ), budeme alg. spoustet iterativne p\*c krat, kazdy krok je r = 1 .... p\*c, kde p je **pocet bodu** X a c je **pocet kamer,** Theta je tedy nejake reseni, mnozina Xi a Pj.

Takze vylepsene theta se rovna spatne theta + ds (ds je zrejme nejaka oprava), pro kazdou iteraci r si vytvorime vektor Lr a na misto i (tedy pro ten bod X ) zapiseme souradnice bodu X a na misto j (tedy pro kameru Pj) zapiseme 11 parametru z Pj, na i 3 parametry z Xi, takze dostaneme takovej divnej vektor, udelame Lr^T \* Lr, tim dostaneme takovou mapu, jak je dole vlevo, ty zlute by meli predstavovat ruzna pronasobeni parametru p z matice P a bodu x, napr. p11\*x1 nebo p32\*x2 apod.. proste ruzne kombinace takovýchhle pronasobeni, pro vsechny iterace r tyto mapy poscitame, a dostaneme tu mapu vpravo, pro kazde i a j to bude trochu posunute**,** a vsechno poscitane

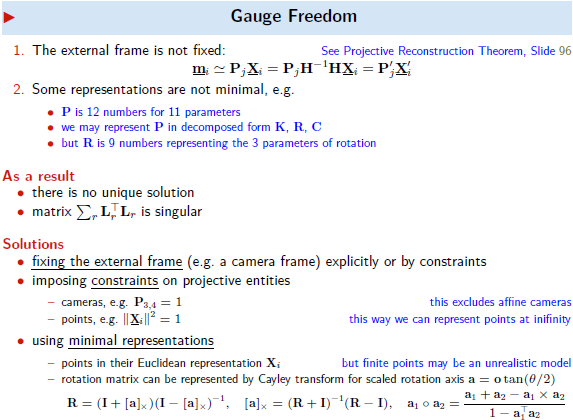
asi tim nejak rychle vypoctou promitnuti vsech bodu, kdyz se x nasobi tema p.



Snazi se tady najit to ds - kterym asi posunou to Theta, aby bylo spravne A = LL^T je dekompozice choleski,

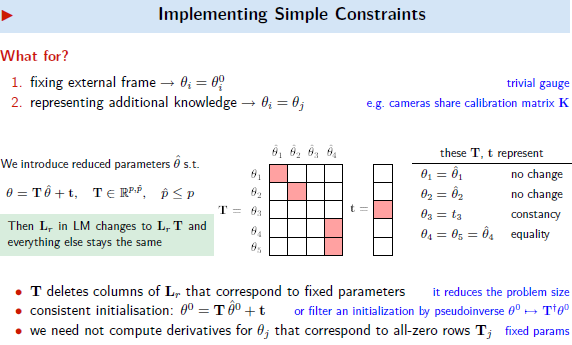
proste ta rovnice se da napsat jako Ax = b ... ta nahore, jenze A je moc velka a pocitat A^-1 je obtizne, takze se to **zjednodusi tim**, ze se provede **choleski dekomp.,** ta je rychla - **nedotyka se zero blocks.**

A pak se to resi ve dvou fazich, kdyz znas tedy L a b, tak se resi L\*c=b a zjistis si c, pak druha faze, znas L a c, zjistis x: L^T\*x = c a a **x je tedy reseni = ds.**

****

External frame - asi to rozmisteni kamer? No tak mluvi se tu o tom, ze reseni neni unique, ze R ma 9 cisel a predstavuje jen 3 parametry, takze se ten frame opravuje, tak ze P(3,4) = 1 tim vyradi nejake afinni kamery (to jsou ty, kde asi P(3,4) != 1)

|xi|^2 = 1



**LM (Levenberg–Marquardt algorithm)**,takze je to nejaka interpolace dvou metod - gauss-newton a gradientni metody, pro nalezeni minimalni chyby a je robustni vic nez GNA, tzn. najde minimalni chybu i kdyz zacne hodne daleko od te konecne minimalni chyby. Je proto taky o neco pomalejsi. See slide 62

mluvi se tu o jacobianech a u nas o maticich L, tak asi je to to same, nebo neco podobneho.

1. **Autocalibration**

Auto-calibration is the process of determining internal camera parameters directly from multiple uncalibrated images.

Auto- (or self-) calibration is the computation of metric properties of the cameras and/or the scene from a set of uncalibrated images. This differs from conventional calibration where the camera calibration matrix K is determined from the image of a known calibration grid (chapter 7) or properties of the scene, such as vanishing points of orthogonal directions (chapter 8). Instead, in auto-calibration the metric properties are determined directly from constraints on the internal and/or external parameters.

Given a set of matched points across several views and constraints on the calibration matrices K^i compute a metric reconstruction of the points and cameras.

Takze cilem je, kdyz zname projekce a K zrekonstruovat body ve 3D - chapu to dobre? a pozice kamer ... jj

Ale je to automaticky s tim, ze se nam K urci samo. Takze dole je napsano, ze K muzeme urcit snadno z choleski dekompozice.

Compute the calibration matrix K\* from the equation w\* = KK^T by Cholesky factorization.

Ale to je alternativne K muze byt spocitana tim postupem, ale hlavni cil algoritmu je neco jineho:

1. vypocteme body ve 3D ze znamych projekci a matic P

2) urcime Q\*\_inf

3) rozlozime Q\*\_inf na H\*I\*H^T

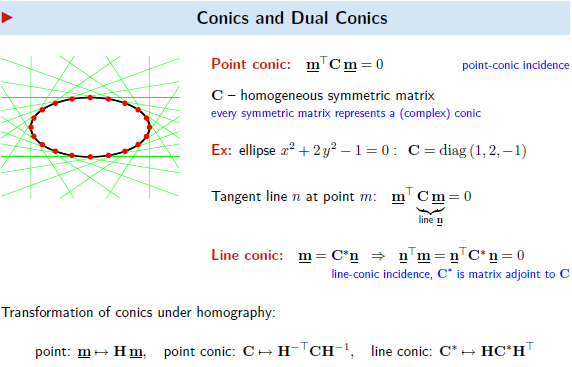
4) pouzijeme H^-1 na body a kamery a dostaneme je v metrickem systemu

5) a pak pomoci least squares vylepsime

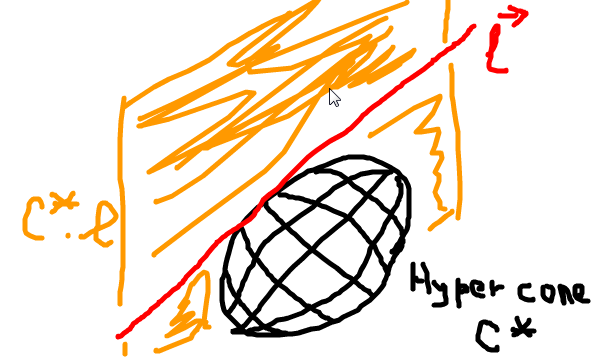
no a co jsme tedy ziskali? body a kamery v metrickem systemu, predtim asi byli v jinem systemu

Takze **framework** neboli **projective frame** je asi **mnozina kamer** a **bodu X.**

Je potreba asi urcit H a potom K.

****

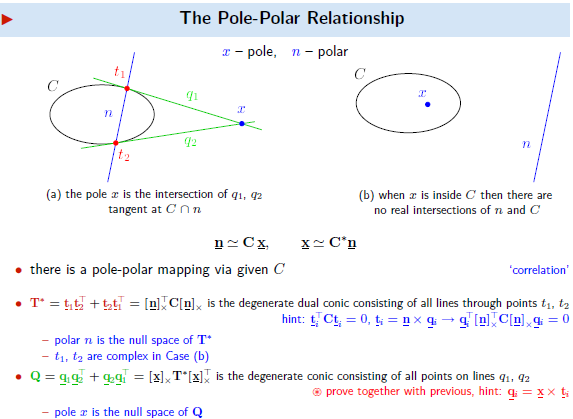
|  |  |
| --- | --- |
| C je nejaka matice, ktera predstavuje kuzelosecku a dokonce komplexni (no asi ze to nemusi byt jen v realnych cislech, ale i v komplexnich), prepis na rovnici je jasnej - podle diag(...), zelena primka je n = Cm a bod m lezi na n, proto m^T \* n = m^T\*C\*m = 0, ted je tam neco s hvezdickou, to je ta sdružená matice, vime, ze n = C\*m  kdyz udelame inverzi: C^-1 \* n = m, takze C\* bude neco co odpovida inverzi C^-1, mozna se to tak znaci kvuli komplex. cislum**,** v realnych by platilo C^-1 v komplexnich se mluvi o sdruzene matici, v kazdem pripade, rekl bych, ze asi plati: C \* C\* = I, ne C je mozna taky v komplexnich anebo muzu nasobit, proste realne cislo je taky prece komplexni, R je podmnozina C  C\* tedy predstavuje kuzel, tak ze y\*x > 0, takze asi je to prostor mimo kuzel podle toho obrazku prvniho  kužel (dual cone) je C\* a kuželosečka (cone) je C, C\* je vždy konvexni, dokonce i kdyz C neni konvexni ani kuzelosecka.  Takze podle knihy  C je conic a plati x^T \* C \* x = 0,  kde x je bod, ktery lezi na te conic, podobne jako pro body, to ale muzeme definovat i pro primky - proto se tomu rika dual conic (body je single conic, primky dual conic) ve zvlastnim pripade, pokud je C neni singularni, tak plati C\* = C^-1  jinak je to proste pridruzena matice  a pro primky plati:  l^T \* C\* \* l = 0 |  |



ke kazdemu conic C existuje C\*, coz lze chapat jako nejakej hyper cone, treba elipsoid, predtim to bylo pro body, ted je to pro primky, mame C\* a nejakou cervenou primku l, ktera se dotyka C\*, C\* \* l bude tedy plocha, ktera se dotyka C\*, predtim u bodu jsme meli C krat bod = primka**,** tady C\* krat primka je plocha a na te plose lezi ta primka l, proto plati ze je to nula, l \* C\* \* l = 0.

Takze mame C pro body a C\* pro primky, pak na 111 je jak se to transformuje pres homografii**,** coz je asi celkem jasne, m^T \* C \* m = 0

(Hm)^T \* H^-T\*C\*H^-1 \* Hm = 0, coz plati, kdyby sis to nezapamatoval, tak muzes odvodit... m se pretransformuje jako H\*m tak aby se ti ta nula nezmenila, musis vhodne Hackem vynasobit i C a zjistis jak se transformuje C.



**x je tedy pol, n je polára**

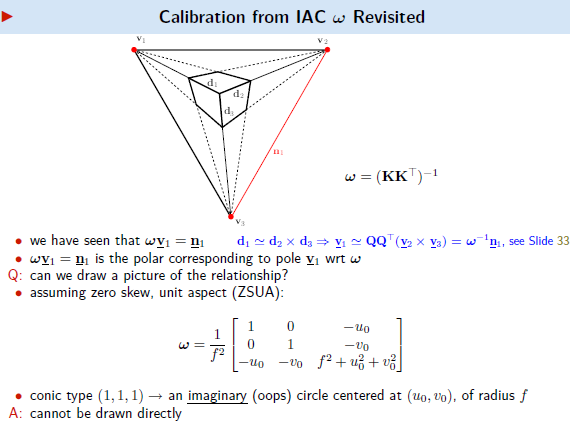
a plati, ze n = Cx, tedy secna = C \* pol, tady je zajimave to, ze pokud budeme bod x priblizovat k C, az nakonec se stane, ze x bude lezet na C, tak bude platit C\*x = tecna, tedy secna se zmeni v tecnu a je to stejne jako na predchozim slajdu

C \* bod na C = tecna

C \* bod mimo C = secna

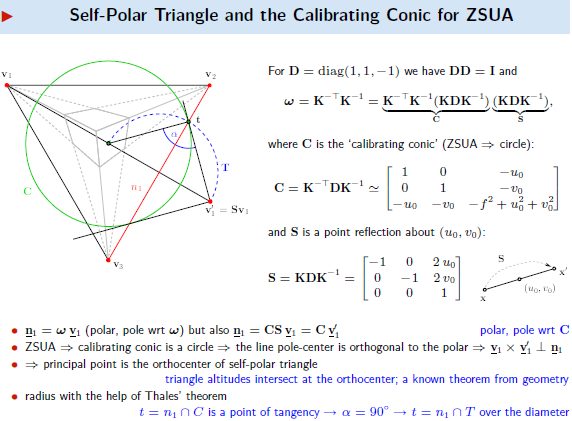
obdobne plati obracene x = C\* \* n

T\* takze pro libovolne x budeme dostavat ruzna n a ty mi vytvori asi nejakej conic, tady to je sloupec \* radek, takze dostanes nejakou matici 3x3 a sectes s jinou matici (obracenou), takze T\* bude nejaka asi symetricka 3x3, coz predstavuje ten novy conic, obdobne je Q.



v1 je **pol**, n1 je **polar,** takze v2 a v3 budou body, kde se ty primky dotykaji kuzelosecky (v1 je pol, n je secna tedy polar, takze v1 - v2 a v1 - v3 jsou tecny)

podle obrazku predtim bych rekl ze to tak musi byt, ale na dalsim obrazku je to jinak, omegu ziskali z Kacek a omega je tedy conic, je symetricka, coz je dobre, conic ma symetrickou matici a z toho zjistili, ze ten conic musi mit nasledujici vlastnosti: **stred** v u0, v0 a **polomer** f a ze je to circle, tak ze znalosti Kacka jsme dospeli k tomu, ze omega predstavuje conic, ktery je kruznice.



Takze vygenerujeme nejakou matici D, pro kterou plati DD = I, takze to bude D = diag(1,1,-1), pomoci takoveho D pak muzeme omega rozepsat podle toho vzorce. Jak ten vzorec vznikl.

Tak jedna cast je C, druha je S

K^-1\*K = I

D\*D = I a pak dalsi K\*K^-1 = I

takze I\*I\*I = I, takze C pak upravili na tvar K^-T\*D\*K^-1, proste uprostred cecka se vyrusi K^-1\*K, vyjadrili si to pres parametry toho K stejne to udelali i s S, ted plati:

n = omega \* v - to je jasny to je podle toho conic, secna = omega \* pol

n = CS \* v, kde omega = CS

necht v' = S\*v

pak plati: n = C \* v'

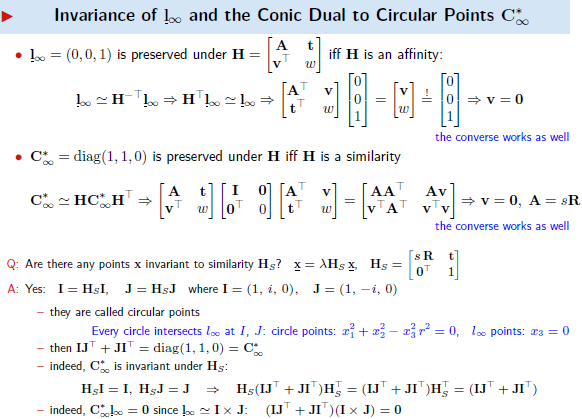
takze postup: nejprve pomoci S pretransformujeme bod v na v', to bude nas pol, zname secnu n1 a pol v1' tak muzeme urcit C, polomer tedy podle thalova theoremu, jak oni zjisti S, kdyz neznaji u0 s v0?

Udelaji kolmice z vrcholu na na prusecik kolmic je stred, to je u0, v0.

**Takze takhle:**1) udelame kolmice z vrcholu vi na hrany ni - jejich prusecik je stred (u0, v0)  
2) sestavime S a provedeme v1' = S\*v1  
3) Thalova veta: udelame kruznici s prumerem: Stred-v1' a t1, t2 budou pruseciky n1 a teto kruznice, potom sestrojime kruznici se stredem Stred = (u0, v0) a na ktere lezi t1 a t2

4) polomer teto kruznice je f

takze tim zname vsechny parametry: u0, v0, f - tedy zname K



afinni ransformace provadi: **rotaci, posunuti, scale, zkoseni**  
similarita: **rotaci, posunuti, scale**

H nemá vliv na l inf pod podminkou ze H je affinity

homografie je afinni, kdyz posledni radek je 0 0 1

takze mame obecnou primku l\_inf

a tu transformujeme pres H^-1 na sebe samou

l\_inf = H^-T \* l\_inf, tohle je jasne

klidne to H^-T muzeme prehodit na druhou stranu

l\_inf = H^-T \* l\_inf  
H^T \* l\_inf = l\_inf, coz je taky jasne, ted to H^T si rozepiseme

A je tedy rotacni cast 2x2, t je translacni cast, v a w je posledni radek matice H (posledni radek v H, posledni sloupec v H^T)

za l\_inf dosadime tu nekonecnou primku (0,0,1) a vynasobime

A je 2x2 ta se vynasobi prvnima nulama z l\_inf a vyrusi se, pak nasobime v\*1 = v

takze [v1; v2; w] = [0 0 1], takze v = 0

odtud plyne, ze **H musi mit tvar**  
  
h11 h12 h13  
h21 h22 h23  
0 0 1

coz je affinni H, **takze tuhle vlastnost maji pouze afinni matice H**

tak ted dalsi cerveny bod

C\*\_inf is preserved if H is similarity, poznamka ... **preserved a invariant** je to same

nevim jestli je dulezite si pamatovat, ze nekonecny conic je diag(1,1,0)

l\_inf^T \* C \* l\_inf = 0 jake musi byt C? a dospeli bysme k tomu, ze C\*\_inf = diag(1,1,0)

takze nekde jsme meli jak se transformuje C pres H (slide 111)

C -> **H^-T \* C \* H^-1** a C\* ->  **H \* C\* \* H^T**

tak my potrebujeme   
C\* -> H \* C\* \* H^T

C\* \_inf je tedy [I 0 ; 0 0 0]

a z obou stran nasobime tim H a H^T odtud dostanes tu dalsi matici s Ackama a musi byt stejna jako C\*\_inf

takze:  
AA^T = I  
Av = 0  
v^T\*A^T = 0  
v^T\*v = 0

odtud vyplyva, ze v = 0 a ze A musi byt rotacni matice

protoze u rozacnich matic plati ze R^T = R^-1 a tady mame AA^T = AA^-1 = I

navic je tam nejakej s jako scale, proste nejaky lambda

H si muzeme rozepsat jako je dole Hs a to je similarita

similarita je: **rotace, posunuti, scale**

afinita: **navic je zkoseni**

jak se tedy lisi matice similarity a afinni?

obe dve maji posledni radek: 0 0 1

lisi se tak, ze **afinni maji prvni dva sloupce a radky libovolne**

a similarita tam nema libovolna cisla, ale takova, ktera odpovidaji rotaci

takze pokud: det H(1:2,1:2) = 1, pak je H similarita

to je jen tak na okraj.. **protoze det R = 1**

i kdyz s tim s (scale) to asi nemusi uplne platit s tim determinantem  
spis plati:  
det H(1:2,1:2) = s

ale to je celkem jedno, proste u similarity prvni sloupce a radky 2x2 odpovidaji rotaci a scalu, nikoliv zkoseni

similarita asi proto, ze zachovava pomer stran .... a tudiz nedochazi ke zkoseni, jj, mozna

pak je tam otazka jestli jsou body invariantni vuci H? ano, říká se jim circular points

protože lajna je, a body jsou všechny na lajně, něco v tom smyslu? jo to je dobrej duvod :)

to I a J tošíš co je? to jsou ty body, ktere se zachovavaji, I=H\*I

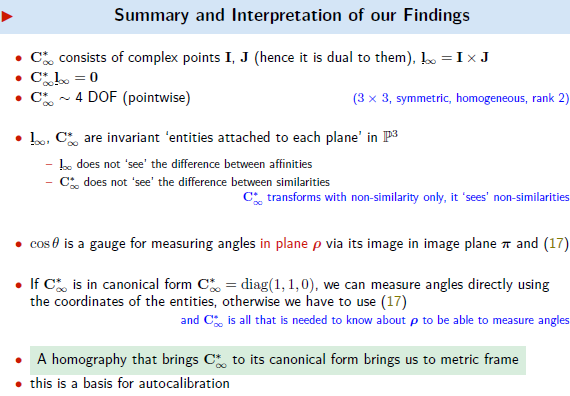
jsou v **nekonecnu** (na *nekonecne primce jak jsi psal*) proto maji treti sour. 0

tak tady vzali nejake dva opacne body, kdyz si predstavis tu sipku v homogennich sour. (ta, ktera protne obr až v nekonečnu) tak jedna smeruje na jeden smer a druha na druhy, i kdyz tohle neni uplne na obracenou stranu, to by musela byt prvni souradnice taky -1, proste jsou to dva nejake ruzne nekonecne body (rekl bych, ze je to asi jedno, proste **body na nekonecne primce**) no a ty tvori nejakou C\*\_inf

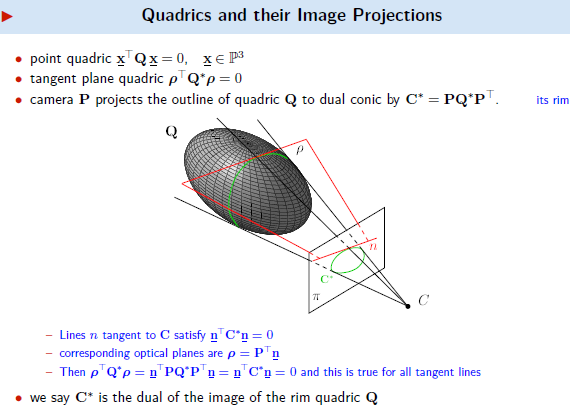
a zrejme plati IJ^T + JI^T je C\*\_inf **= diag(1, 1, 0)**

**a protoze ty body jsou invariantni vuci H tak ukazuji, ze i C\*\_inf je kvuli tomu invariantni vuci H** (Invariance of l \_inf and the conic dual to circular points C\*\_inf)

na 114 je **T\* = t1\*t2^T + t2\*t1^T**  
takze kdyz vezmu nejake body t1 a t2 a udelam s nimi tohle, tak dostanu nejakej dual conic T\*  
  
zde na 121 je neco podobneho:  
vemu nejake dva body v nekonecnu a udelam: **IJ^T+JI^T** a dostanu C\*\_inf

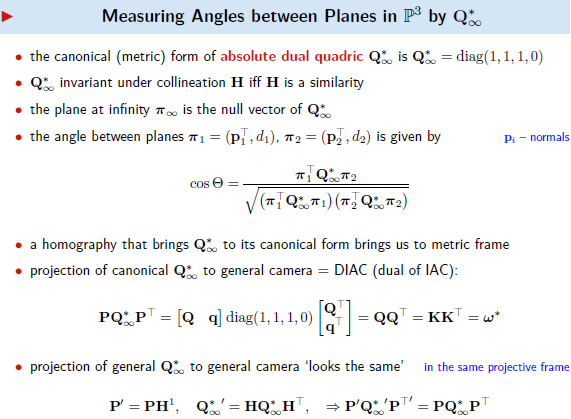


Takze nekonecna primka je nek. bod I x nek. bod J, primka l \_inf v nekonecnu lezi na C\*\_inf kdyz: skalar. soucin je nulovy .. jj nekonecna primka lezi na nekonecnym conic.



Pro bod na Q plati: x^T Q x = 0, to je jasné, rekl bych tedy, ze Qx bude rovina ro a x lezi na teto rovine ro, proto se to rovna nule, obdobne to plati pro roviny ro, ale s tim, ze mame Q\*, mám tu poznámku z minula u toho prvního vzorce: bod x 3D lezi na Q (obalu), když se to rovná 0, pokud Qx je rovina ro, tedy ro = Qx, tak pro kazdy bod, ktery lezi na rovine plati, ze skalarni soucin je nulovej a ted **projekce conicu.**

quadratic Q\* se promitne jako C\* a plati: C\* = PQ\*P^T rovina ro se promitne jako primka n takze v projekci pak plati n^T C\*n = 0, ro = P^T n - tohle jsme uz meli nekde na zacatku, takze se to z toho odvodi.



Takze Q\*\_inf je diagonala 1,1,1,0, C\*\_inf byla 1,1,0 - **takze jednicky a na konci nula**,pak taky plati, ze Q\*\_inf je invariantni vuci similarite stejne jako C\*\_inf, ta kolineace je co? to H? H je homografie - ktera je similarita

**rovina v nekonecnu**.. hm.. asi mnozina nekonecnych primek :)

uhel mezi rovinami, vypada to podobne jako uhel mezi vektorama, cos alfa = u\*v / (|u|\*|v|)

chce si to asi zapamatovat, ale vzhledem k podobnosti s uhlem mezi vektory, to asi neni tak slozite

to p a d v te rovine je co? p1 je normala roviny a d1 nejake posunuti

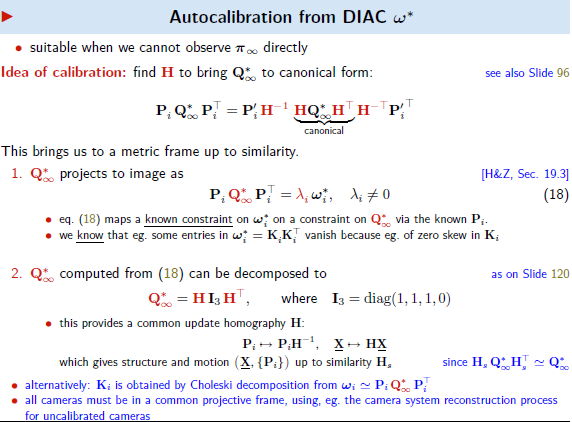
to mas jako rovnici roviny: **p1\*x + p2\*y + p3\*z + d = 0**

pritom (p1,p2,p3) je normala a d je posunuti

**a homography that brings C\*\_inf to its canonical form brings us to metric frame** tuhle vetu uz vidim 2x, asi bude dulezita

kdyz **promitneme** Q\*\_inf tak dostaneme omega\* tedy asi omega^-1, Q\*\_inf se promita na w\* coz je w^-1 = KK^T

a posledni poznamka, je ze Q\*\_inf se promita na stejnou kuzelosecku pro kamery upravenymi libovolnou homografii H



Takze kdyz zname Q\*\_inf muzeme urcit w\* a odtud take kalibraci K, takze ta poznamka dole na 125 je asi dulezita

ze P' Q\*inf P'^T = P Q\*inf P^T, toho ted vyuzijeme a dovnitr nacpeme H^-1\*H aby se to nezmenilo, tak je to ten DIAC. HQ\*\_inf H^T bude kanonicka forma.

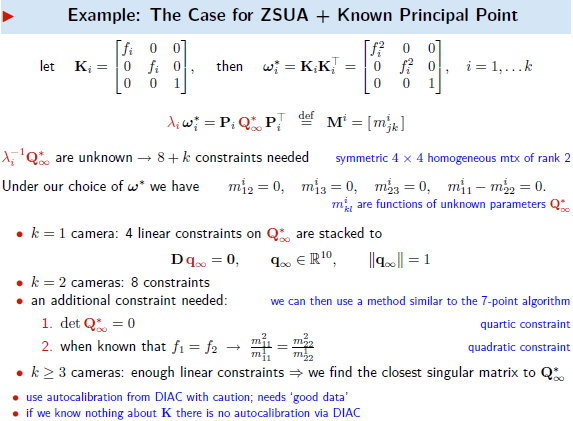
1) to je asi opakovani toho, ze Q\*inf se promita na w\*

Q\*inf se decomposuje na H I3 H, H I3 H^T, pomoci ziskaneho H se upravi P a body X

nevim proc to delaji, ale v kazdem pripade z w\* lze pomoci choleski dekompozice ziskat K

protoze choleski dela rozklad **omega\* = KK^T**, tak jde o kalibraci, autokalibraci ... tady chceme právě to K dostat

A to s tim Hackem se dela asi proto, ze to musi byt ve stejnem projective framu, coz je to rozmisteni kamer,jo? Snad.



Kdyz zname principal point, tedy u0, v0, tak K se nam zjednodusi, K = diag(f,f,1) a w\* = diag(f^2,f^2,1).