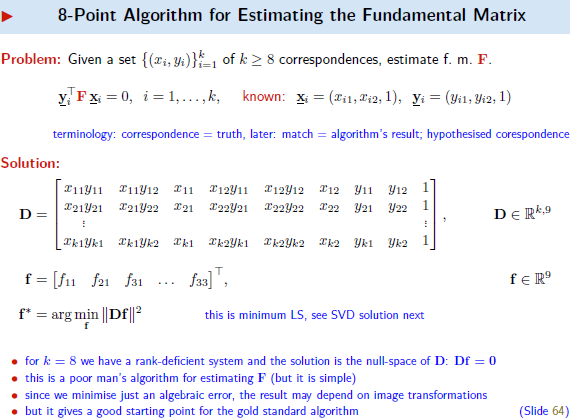
# 17. Odhad parametrů epipolární geometrie z korespondencí, osmibodový a sedmibodový algoritmus. Metoda vyrovnání svazku.

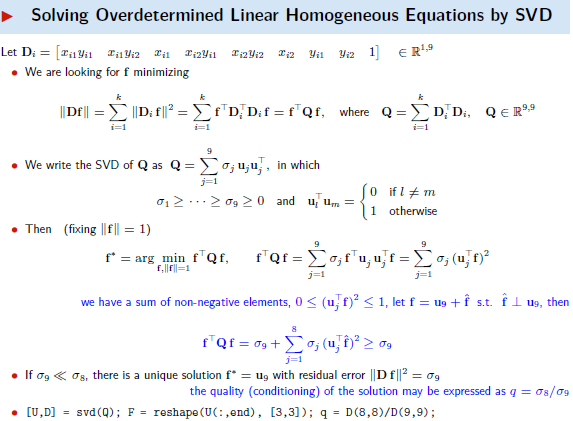
1. **Odhad parametrů epipolární geometrie z korespondencí** (viz. TZ)
2. **Osmibodový a sedmibodový algoritmus.**

****

Mame y^T\*F\*x = 0, kdyz si to cele roznasobime a F prevedeme na dlouhy vektor f 9x1, takze dostaneme D\*f = 0

D, protoze nemuzeme dosahnout toho, ze Df = 0 kvuli chybam, tak se snazime chybu alespon **minimalizovat**

f\* = argmin |Df|^2, coz se resi metodou **nejmensich ctvercu** (tedy pomoci SVD, jejiz resenim je vektor, ktery je vysledkem nejmensich ctvercu).



Tady ukazují, jak SVD resi ten problem (dost hard-core)

**[U D V] = svd(D)  
f = V(:,end)**

tady on ale dela: [U D] = svd(D^T \* D) neboli [U D] = svd(Q), f = U(:,end)

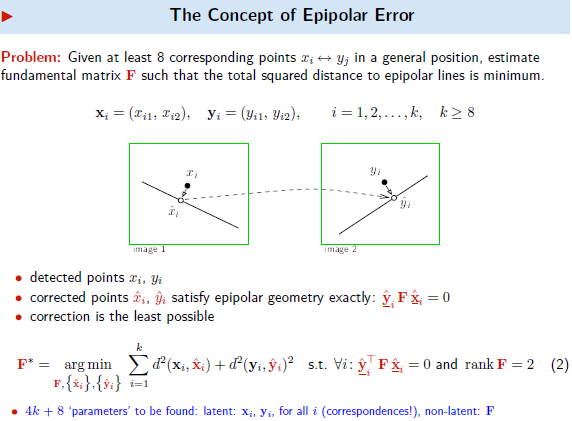
vyzkousel jsem to v matlabu :) a vychazi to uplne stejne

mimochodem v nejmensich ctvercich je ve vzorci: (D' \* D)^-1 \* ..., takze je tam taky D' \* D takze proto to asi funguje.. ma to nejakou souvislost.

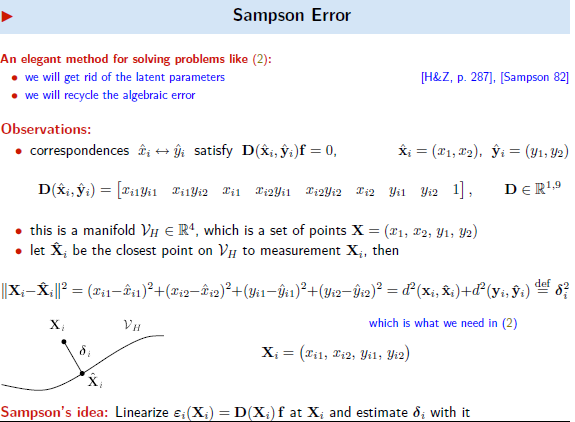


Timhle postupem dostaneme F s rankem 3, jenze **F musi mit rank 2**,aby se to zlepsilo tak se dela 7-point alg.**,** takze nejen pro zrychleni ! ale taky pro nalezeni **F s rankem 2**, coz je presnejsi (ale 8-point alg. najde F s rankem 3 !!!), pak uz tam neni nic zajimaveho... proste ze jsou degenerovane pripady a tak. tak dal

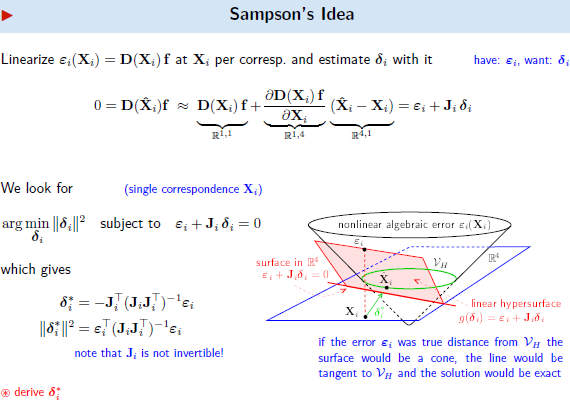
**takze chce to F\* ktery ma rank = 2 a pritom je nejblizsi k F**

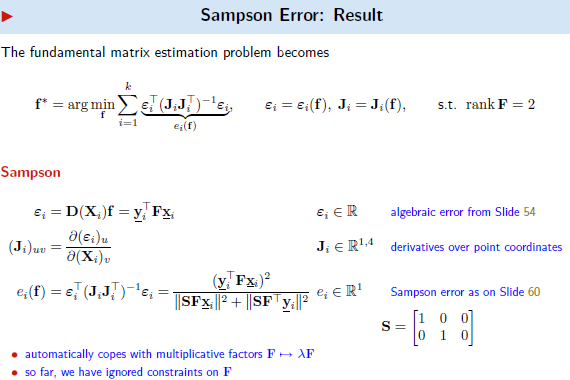


Ten vztah dole je celkem jasnej, F\* je takove, pro ktere je ta chyba **minimalni** a chyba je soucet ctvercu vzdalenosti.

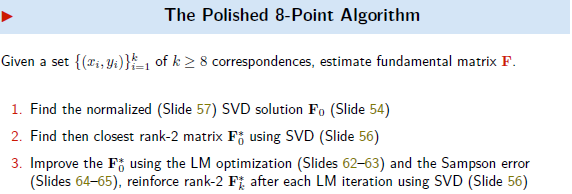


Takze x1, x2, y1 a y2 - souradnice projekci nacpeme do jednoho 4D vektoru, plati-li epipolarni geom. tak tyto 4D body musi lezet na nejakem v (manifold - asi varieta), najdeme nejblizsi 4D bod X' ktery lezi na variete a je nejblize k X.





Zde tu je taky ten vzorec v trochu jine forme, jacobian je nejaka matice parcialnich derivaci, ale nevim jak to tam presne funguje, ale dokonce tam i vynucuji, aby **rank F = 2**.



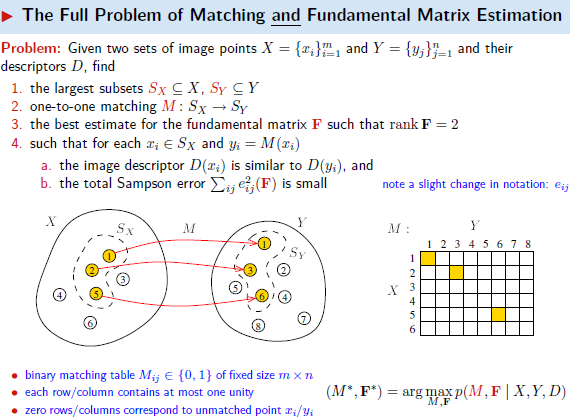
Takze tady je postup ve 3 bodech:  
**1)** najdeme F0 pomoci SVD - svd(D) a posledni sloupec z V

**2)** najdeme F0\* s rankem 2 nejblizsi k F0 pomoci SVD

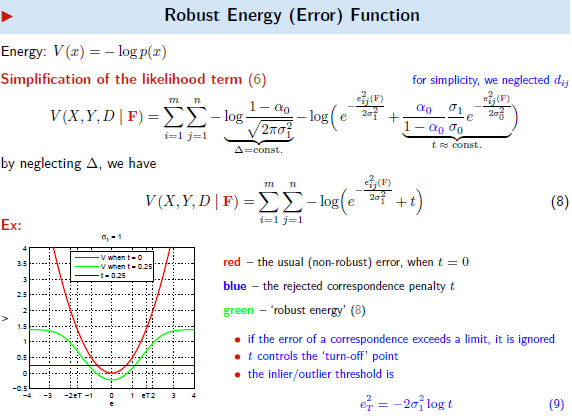
**3)** smysl je takovy, ze svd rozlozi na: rotace \* scale \* rotace, scale matice je diagonalni, obsahuje tedy 3 cisla na diagonalach, co cislo to jakeho zvetseni dosahneme na dane ose, chceme-li aby hodnost F byla 2, tak potrebujeme aby hodnost teto scale matice S byla 2, matice S ma take na diagonale cisla serazena od nejvetsiho po nejmensi

takze na pozici S(3,3) je nejake malinke cislo, kdyz toto cislo vynulujeme, tak dostaneme nejblizsi reseni s rankem 2

takze oni tady na str. 56 v tom vzorci, si svd rozepsali na sigma u a v a vsimni si, ze maji jen **sigma1 a sigma2,** sigma3 uvazuji jako **nulove,** takze to je oprava F\*, aby mela rank 2, po oprave F0\* na rank 2 se  
**3)** F0\* vylepsi sampsonovym errorem a znova se dela **2)**

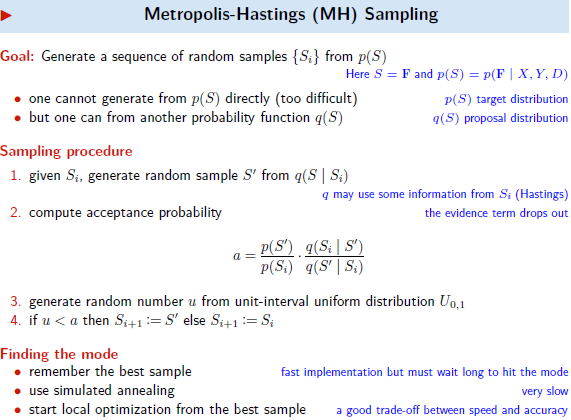


Takze hledame inliery, to je asi celkem jasny, proste hledame nejvetsi mnozinu korespondenci (nejvetsi pocet inlieru ty zlute body jsou **inliery**) tedy spravné korespondence, které vyhovuji modelu, takze cely tenhle problem je zapsan tim vztahem dole v rohu, hledame max M a nejlepsi F, pro ruzna X,Y a jejich deskriptory D.



Takze v = e^2 - tedy parabola, jenze, co kdyz je nejakej bod proste porucha a je hodne daleko, zbytecne mi dobry model bude kazit, protoze pricte **velikou chybu**, tak radeji pouziju zelenou robust energy, kdyz je e prilis vysoke, tak ta chyba kterou pricte bude temer konstantni, takze nejaky hodne vzdaleny bod mi to nemuze prislis pokazit

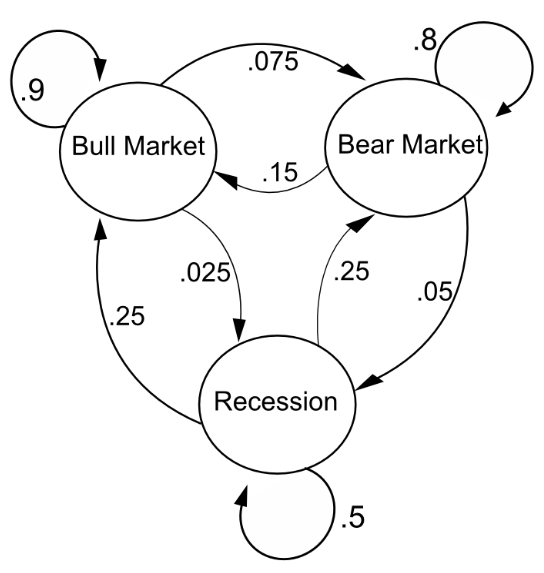
no a my jsme delali aproximaci nejakeho ML estimatoru.. jestli si dobre pamatuju, tak se to jmenovalo, coz je neco velmi podobneho te zelene krivce. t je tedy nejaka konstanta, proto je zelena krivka robustnejsi, ze ji nemohou nejake poruchy vzdalene rozhodit, jinak ty vztahy jsou slozity docela.. premyslim, kterej z nich by popisoval tu zelenou, ta 8.



**Cil** - vygenerovat nahodne samples podle nejake pravdepodobnosti, podle pravdepodobnost p(S) vygenerovat vzorky S je slozite, proto se to dela pres pravdepodobnost q(S|Si)**,** vyuziva se markov-chain

generuju posloupnost ruznych tech 3 prvku a nachazim se vzdycky v **nejakem stavu** a s urcitou pravdepodobnosti vygeneruju neco a prejdu do jineho stavu

**staci jen vedet, ze se pohybujes mezi stavama a s urcitou pravdepodobnosti generujes nejakej vzorek**

****

no takze tyhle cisla zrejme predstavuji ty pravdepodobnosti q(S|Si), prechody mezi stavama, takze aby jsi vygeneroval treba tu cervenou krivku na 72, tak si napred asi musis nejak odhadnout ty pravdepodobnosti pro prechody mezi stavama, tak zpet na 73

takze asi to chce si zapamatovat ty 4 kroky

treba jsem ve stavu Bull market, tak z 90% vygeneruju Bullmarket a zustanu v tom stavu nebo s nizsi pravd. vygeneruju neco jineho a prejdu tam, tak takhle to funguje, **generuju a prechazim mezi stavama**

1) jsme ve stavu Si (naposledy jsme vygenerovali vzorek Si) a s urcitou pravdepodobnosti q(S|Si) podle te sipky v grafu vygenerujeme vzorek S',

2) vypocteme si a = ... vygenerujeme u od 0 do 1,

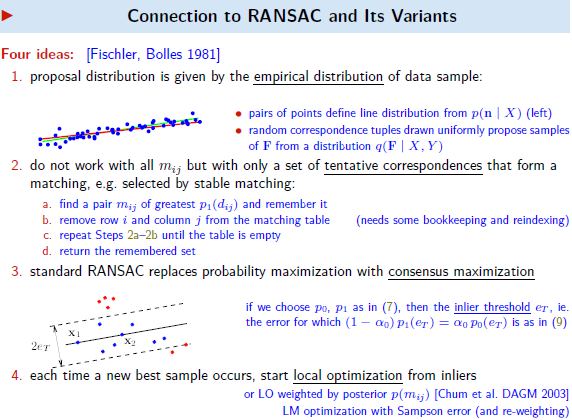
3) vygenerujeme u od 0 do 1,

4) pokud je u < a, pak si vygenerovany vzorek zapamatujem a prejdeme do toho stavu, kdyz ne tak nic nedelame a zustaneme ve stejnem stavu

a tady tedy generujeme ruzne matice F, podle pravdepodobnosti dane velicinami X, Y, D

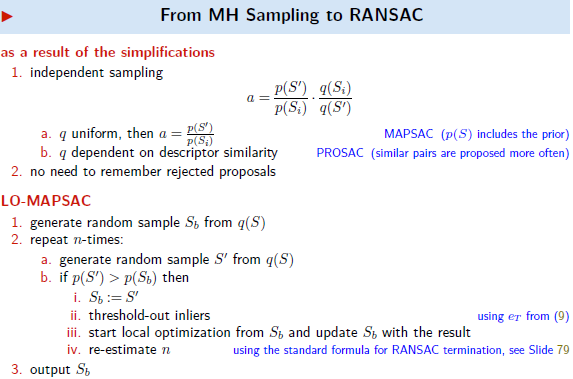
u je proste nahodne cislo

misto toho aby jsi vzdycky rekl, ze prejdes do dalsiho stavu, tak tady je to upravene jeste o tu acceptance, ze pokud je nahoda < a, tak jo prejdi, pokud ne, tak zustan, S' je nove F

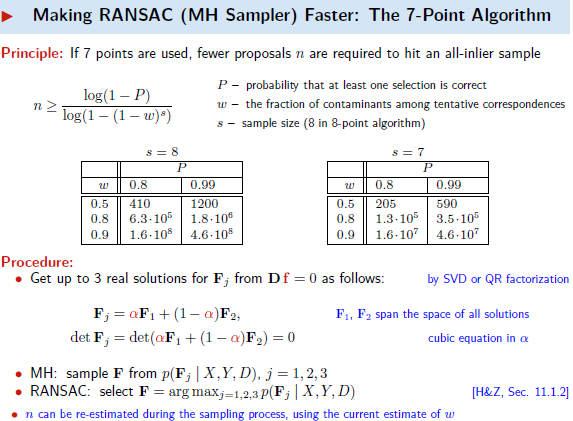


Takze tady se neco dela s tou matching table, najde se **par** m\_ij s **nejvyssi pravdepodobnosti,** zapamatuje se

a odstrani z tabulky, radek a sloupec, zrejme se hledaji inliery, proste jeden bod muze s urcitou pravdepodobnosti mit korespondenci s jinymi body no a vybere se takovy s nejvyssi pravdepodobnosti a pak se tam dela lokalni optimalizace, kterou asi navrh Chum :-D



LO-MAPSAC jen ze dela navic tu lokalni optimalizaci.



Je rychlejsi, ten vzorec nahore je pocet opakovani, tak si muzes spocitat o kolik.

Dalsi vlastnost 7-point je ze umoznuje udelat, aby **F mela rank 2**, a o tom je dole to Procedure, takze jak jsem zjistil, obvykle **F** je **posledni sloupec matice V** u **SVD**, tady je ale F1 a F2 a pry to jsou dva posledni sloupce, takze skutecne F je nekde mezi temito dvema F1 a F2.

F1 a F2 tedy tvori nejakou primku a pomoci alfa se na te primce pohybuju bud bliz k F1 nebo bliz k F2, optimalni reseni je tedy nekde mezi nimi**,** F = a\*F1 + (1-a)\*F2.

Spocita se det F, aby F mela rank 2, tak jeden radek musi byt nulovy, tudiz det F = 0, takze to plati i pro ten vyraz s alfou, a to ma nejakou souvislost s **vlastnimi cisly** - a protoze je to matice 3x3, tak jsou 3 vlastni cisla a proto existuji 3 reseni! (myslim, ze se na to ptali i v testu).

No jasně, det F = 0, determinant se pocita jako nasobeni po diagonale a protoze je to 3x3 tak dostaneme kubickou rovnici, ktera se rovna nule a ta ma 3 koreny, a na ty diagonále budou ty vlastní čísla lambdy.

1. **Metoda vyrovnání svazku** (Bundle Adjustment, viz 16)