Technical University of Cluj - Napoca
Computer Science Department

Interactiune Om-Calculator

Curs 6

Potrivirea si urmarirea trasaturilor in secvente de imagini

(Features matching & tracking)

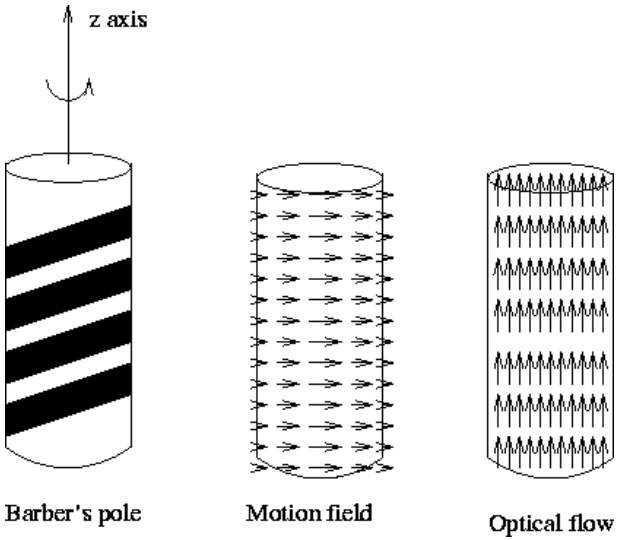
Campul de miscare si fluxul optic

Camp de miscare (motion field):= setul vectorilor (vitezelor) de miscare ale punctelor din imagine (2D) induse de miscarea relative dintre scena (obiecteale scenei) si camera

- nu este masuarabil direct din imagine !

Fluxul optic (optical flow) := miscarea aparenta a paternurilor de intensitate din imagine

- se poate masura direct din imagine
- este o aproximare a campului de miscare cu o rata de eroare mica in puncte cu gradient mare (daca directia gradientului si directia miscarii coincid)



Masurarea campului de miscare

1. Detectia unor trasaturi relevante (fara ambiguitati de pozitionare) in fiecare imagine

-trasaturi primare: colturi [muchii]

-trasaturi complexe: obiecte

2.Potrivirea acestor trasaturi in imagini succesive

- potrivirea (matching) intre 2 imagini consecutive

- urmarirea (tracking) in imagini consecutive multiple

Potrivirea trasaturilor in imagini succesive

Algoritmul FEATURE_POINT_MATCHING [1] (Trucco)

- Derivat din algoritmul de optical flow CONSTANT_FLOW (bazat pe metoda least squares)

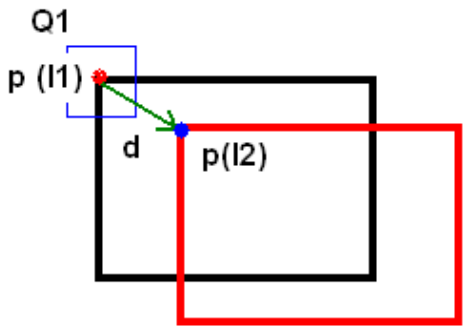
- Aplicare pe coltrui (Harris, KLT)

-Se aplica pe 2 imagini succesive I_1 si I_2 dintr-o secventa

Idee:

- Pt. fiecare punct de interes ***p*** din imaginea $I_1(t-1)$ se considera o fereastra $Q_1(NxN)$

-Se cauta deplasamentul ***d*** al acestui punct in imaginea $I_2(t)$



Potrivirea trasaturilor in imagini succesive

Algoritmul FEATURE_POINT_MATCHING [Trucco]

Pentru fiecare punct de interes $\boldsymbol{p}(x_2,y_2)$ din imaginea I_2:

1. Se seteaza $d = 0$ si se centreaza Q_1 la locatia $\boldsymbol{p} = \boldsymbol{p} + d$ in I_1
2. Se estimeaza $d_0=[-v_x,-v_y]$ folosind algoritmul de optical flow:

   si se actualizeaza d: $d=d+d_0$

$$v = \left(A^T A\right)^{-1} A^T B$$

3. Fie Q' fereastra din I_1 centrata in $\boldsymbol{p}+\boldsymbol{d}_0$. Se calculeaza diferenta S (SAD sau SSD) intre ferestrele Q_2 si Q':

$$S_{SAD} = \sum_{i=-\frac{w}{2}}^{\frac{w}{2}} \sum_{j=-\frac{w}{2}}^{\frac{w}{2}} \left| I_1(x_1+i, y_1+j) - I_2(x_2+i, y_2+j) \right|$$

4. Daca S>*Th* $\Rightarrow$ Q1 = Q' si salt la pasul 1. Altfel (S<*Th*) $\Rightarrow$ exit

Output: Set de deplasamente d pentru toate punctele de interes din imagine

Potrivirea trasaturilor in imagini succesive

Lucas Kanade Feature "Tracker" (impl. piramidala)

http://robots.stanford.edu/cs223b04/algo_tracking.pdf

u(x,y) – punct in imaginea I (t) → locatia *v* = *u*+*d* in imaginea J (t+Δt) ???

Scop: gasirea vectorului de deplasament **d**

Solutia pt d: minimizarea functiei reziduale *e*

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \left(I(x,y) - J(x+d_x, y+d_y)\right)^2 .$$

Calcul flux optic prin metoda Lucas – Kanade printr-un algoritm iterativ pe o piramida de imagini

LKT piramidal - initializari

Initializari:

- Contsruieste reprezentarea piramidala a imaginilor I si J:

$$\{I^L\}_{L=0,\dots,L_m} \text{ and } \{J^L\}_{L=0,\dots,L_m}$$

m – nivelul piramidei (L=0 – rezolutia initiala)

Lm =2,**3**,4

- Imaginea de la nivelul L se obtine din cea de la nivelul L-1 prin decimare ponderata (gaussiana)

- Coordonatele lui u in piramida:

$$\mathbf{u}^L = \frac{\mathbf{u}}{2^L}.$$

$$\begin{aligned} I^L(x,y) = & \frac{1}{4} I^{L-1}(2x,2y) + \\ & \frac{1}{8}\left(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)\right) + \\ & \frac{1}{16}\left(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)\right). \end{aligned}$$

LKT piramidal – notatii

Functia reziduala pentru calculul lui d la nivelul L

$$\epsilon^L(\mathbf{d}^L) = \epsilon^L(d_x^L, d_y^L) = \sum_{x=u_x^L-\omega_x}^{u_x^L+\omega_x} \sum_{y=u_y^L-\omega_y}^{u_y^L+\omega_y} \left(I^L(x,y) - J^L(x+g_x^L+d_x^L, y+g_y^L+d_y^L)\right)^2.$$

Obs: fereastra de integrare are dimensiune constanta pt orice nivel L al piramidei: $(2\omega_x+1)\times(2\omega_y+1)$

→ se pot calcula valori mari ale fluxului optic cu efort de calcul mic (constant)

Calculul fluxului optic (rezidual):

$$\mathbf{g}^{L-1} = 2\,(\mathbf{g}^{\mathbf{L}} + \mathbf{d}^L).$$

$$\mathbf{d} = \sum_{L=0}^{L_m} 2^L\,\mathbf{d}^L.$$

$$\mathbf{g}^{L_m} = [0 \;\; 0]^T.$$

.....

$$\mathbf{d} = \mathbf{g}^{\mathbf{0}} + \mathbf{d}^0.$$

g – estimarea initiala la nivelul L
d – masuratoarea de la nivelul L

LKT piramidal – algoritm

For $L = L_m$ **down to** *0* (step -1) // pentru fiecare nivel L din piramida

Punctul u(x,y) va avea in imaginea I^L coordonatele: $\mathbf{u}^L = [p_x \;\; p_y]^T = \mathbf{u}/2^L$

Se calculeaza derivatele imaginii I^L:

$$I_x(x,y) = \frac{I^L(x+1,y) - I^L(x-1,y)}{2}$$

$$I_y(x,y) = \frac{I^L(x,y+1) - I^L(x,y-1)}{2}$$

Matricea de autocorelatie:

$$G = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)\, I_y(x,y) \\ I_x(x,y)\, I_y(x,y) & I_y^2(x,y) \end{bmatrix}$$

Initializare algoritm LK iterativ (fluxul optic):

$$\overline{\nu}^0 = [0 \;\; 0]^T$$

LKT piramidal – algoritm OF / piramida L

Algoritm LK iterativ (calculul fluxului optic pt. fiecare L)

For *k=1* **to** *K* (step 1) (sau || η^k || < th) // ex: th = 0.03 sau K=20

Imaginea diferenta:

$$\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + \nu_x^{k-1}, y + g_y^L + \nu_y^{k-1})$$

Vector eroare imagini:

$$\bar{b}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y)\, I_x(x,y) \\ \delta I_k(x,y)\, I_y(x,y) \end{bmatrix}$$

Corectia pt. fluxul optic:

$$\bar{\eta}^k = G^{-1}\, \bar{b}_k$$

Fluxul optic la pasul k:

$$\bar{\nu}^k = \bar{\nu}^{k-1} + \bar{\eta}^k$$

End (for k)

LKT piramidal – algoritm

Fluxul optic final la nivelul L:

$$\mathbf{d}^L = \overline{\nu}^K$$

Estimarea pentru nivelul L-1 (urmator):

$$\mathbf{g}^{L-1} = [g_x^{L-1} \quad g_y^{L-1}]^T = 2\,(\mathbf{g^L} + \mathbf{d}^L)$$

End (for L …)

Vectorul de fluxul optic final:

$$\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$$

Locatia corespondenta in imaginea J:

$$\mathbf{v} = \mathbf{u} + \mathbf{d}$$

Implementare OpenCV: calcOpticalFlowPyrLK

Input: set de trasaturi (colturi) – ex. goodFeaturesToTrack

Potrivirea trasaturilor

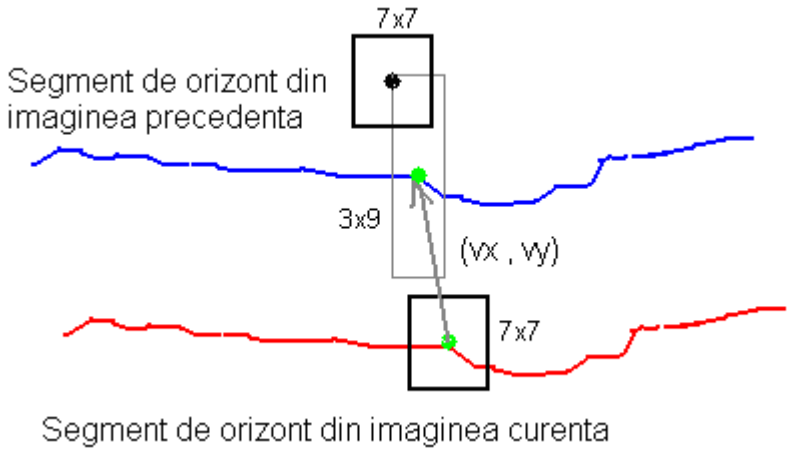
Exemplu de aplicatie pentru asistenta condunducerii autovehiculelor

- Estimarea inclinarii dinamice a unei camere montate pe un autovehicul in miscare
- Urmarirea unor trasaturi (puncte) aflate la distanta mare (distanta se poate aproxima la infinit in raport cu deplasamentul (trasnlatia) camerei intre cadre succesive
- Puncte aflate la infinit – puncte de pe linia orizontului intr-un scenariu de drum (oricare altul decat urban)

Estimare inclinarii dinamice (pitch) a unei camere mobile

Principiul metodei

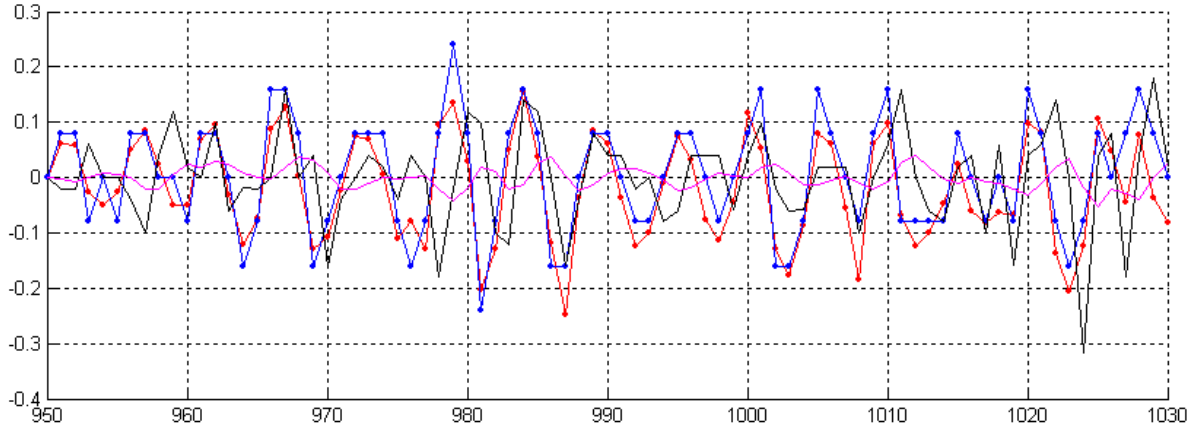
1. Detectia liniei orizontului
2. Potrivirea in imagini succesive a unor segmente de pe linia orizintului
3. Potirvirea in imagini succesive unor puncte individuale de pe linia orizontului
4. Calculul campului de miscare al acestor puncte (matching)
5. Estimarea vitezelor unghiulare al e camerei



$$\begin{cases} v_X = \dfrac{-T_X f + T_Z X}{Z} + \varpi_X \dfrac{xy}{f} - \varpi_Y \left(\dfrac{x^2}{f} + f\right) + \varpi_Z y \\ v_Y = \underbrace{\dfrac{-T_Y f + T_Z Y}{Z}}_{Translatie \to 0} + \underbrace{\varpi_X \left(\dfrac{y^2}{f} + f\right) - \varpi_Y \dfrac{xy}{f} - \varpi_Z x}_{Rotatie} \end{cases}$$

Estimare inclinarii dinamice (pitch) a camerei

Rezultate

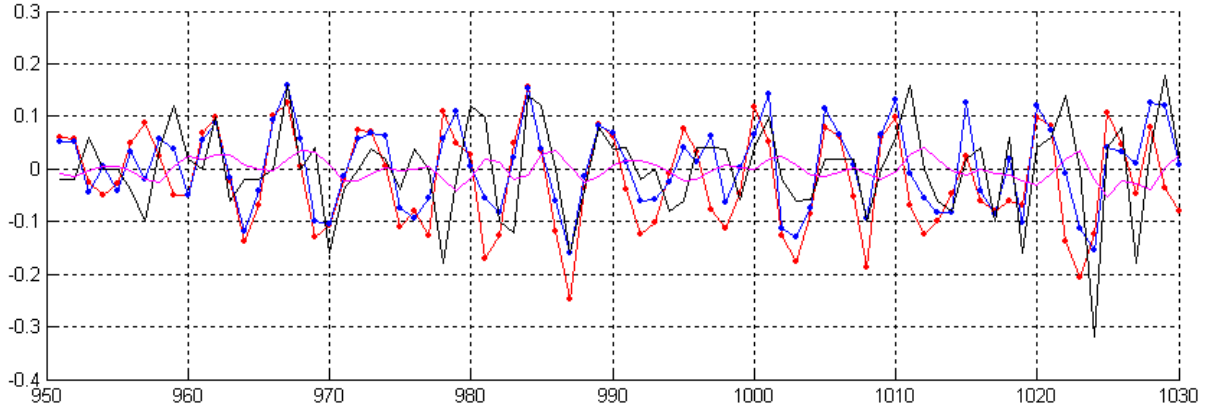


Roşu - metoda liniei orizontului

Albastru - modulul LD

Negru - senzorii de înălţime ai suspensiilor

Variaţia unghiului de pitch [grd/cadru] în funcţie de numărul cadrului

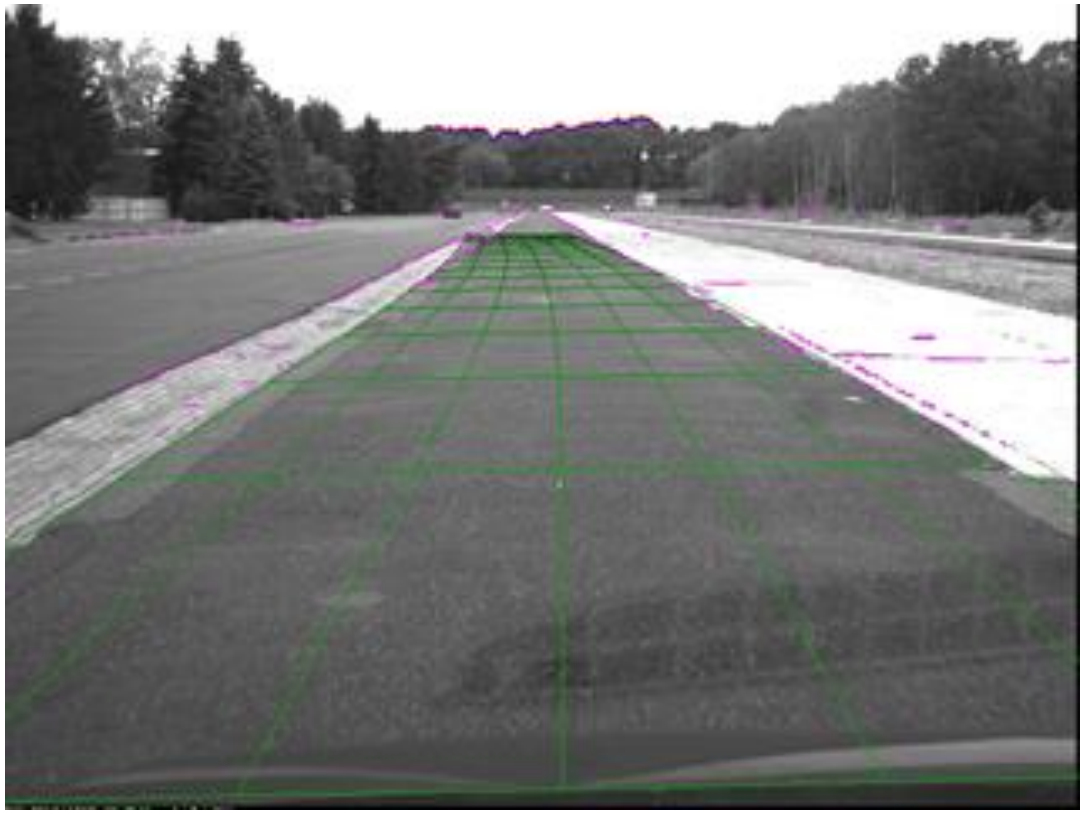


Variaţia unghiului de pitch [grd/cadru] în funcţie de numărul cadrului.

In estimarea dată de modulul LD s-a folosit ca predicţie măsurătoarea dată de metoda liniei orizontului

Estimare inclinarii dinamice (pitch) a autovehiculului

Rezultate



Tracking

Tracking := potrivirea (matching) / urmarirea trasaturilor in imagini consecutive multiple

Filtru Kalman – metoda de estimare optimala

$\Rightarrow$ Algoritm recursiv ce permite estimarea pozitiei si incertitudinii in imaginea urmatoare a unui punct / trasaturi aflate in miscare

pozitie := unde sa cautam

incertitudine := cat de mare trebuie sa fie regiunea in care sa cautam in jurul pozitiei prezise pentru a fi siguri ca gasim trasatura cu un anumit grad de confidenta

Filtru Kalman liniar

- model al sistemului

- model al masuratorii

Exemple aplicatii tracking

Exemple aplicatii Tracking:

http://www.youtube.com/watch?v=vfgIXdbEyUw&feature=related (Kalman – ball)

http://www.youtube.com/watch?NR=1&v=lvmEE_LWPUc Kalman – ball + oclusions)

http://www.youtube.com/watch?NR=1&v=hxTVPzeLHiA (Kalman – pendulum + ocl)

http://www.youtube.com/watch?v=fRowYlxKt7s (Kalman – humans)

http://www.youtube.com/watch?NR=1&v=GC8O3G2uwE4 (Mean Shift – human)

http://www.youtube.com/watch?NR=1&v=6G07zXQDV2c (tennis ball)

http://en.wikipedia.org/wiki/Hawk-Eye#Tennis (patent)

http://www.youtube.com/watch?v=-q5aXTg0pVE&feature=related (finger tracking)

http://www.youtube.com/watch?NR=1&v=iXPkXqU4WRE (finger tracking)

http://www.youtube.com/watch?NR=1&v=91tYEgpmN4M (head tracking)

http://www.youtube.com/watch?v=NCtYdUEMotg&feature=endscreen (eye gaze tracking (paper)

http://www.youtube.com/watch?NR=1&v=JmxDIuCIIcg (car tracking)

http://www.youtube.com/watch?NR=1&v=LNaL58GRou4&feature=endscreen

Modelul sistemului

Un **sistem** este modelat prin:

- Vector de stare: $\mathbf{x}(t) = [x_1, x_2, .., x_n]^T$ - variabilele stemului

- Set de ecuatii: **model al sistemului**

$\Rightarrow$ evolutia in timp a vectorului de stare

$$t_k = t_0 + k\Delta T$$
$$\mathbf{x}_k = \mathbf{x}(t_k)$$

$\Delta T \to 0 \Rightarrow$ model liniar (vectorul de stare nu se schimba semnificativ la momente de timp consecutive):

$$x_k = \Phi_{k-1} x_{k-1} + \xi_{k-1}$$

ξ_{k-1} – vector aleator ce modeleaza zgomotul aditiv

Φ - matricea de tranzitie a starilor (nxn)

Modelul masuratorii

Se presupune ca la fiecare moment t_k se face o masuratoare (cu zgomot) a vectorului de stare (sau cel putin a catorva componente):

$$z_k = H_k x_k + \mu_k$$

$\mathbf{z}_k$ – vectorul de masuratori de la momentul t_k *(mx1)*

H_k – matricea de masuratori *(mxn)*

μ_k – vector aleator care modeleaza zgomotul aditiv *(mx1)*

Zgomotul:

$\Rightarrow$ se presupune: *zgomot alb, aditiv, medie 0, distributie gaussiana*

$\Rightarrow$ *matrici de covarianta:* sistem $\xi_k \rightarrow \boldsymbol{Q}_k$, masuratoare $\mu_k \rightarrow \boldsymbol{R}_k$

http://en.wikipedia.org/wiki/Covariance_matrix,
http://www.itl.nist.gov/div898/handbook/pmc/section5/pmc541.htm

Matricea de covarianta

Exemplu: set de 5 masuratori / pentru un vector de 3 coordonate [x_1,x_2,x_3]

$$\mathbf{X} = \begin{array}{c} \\ \text{mas1} \\ \text{mas2} \\ \text{mas3} \\ \text{mas4} \\ \text{mas5} \end{array} \begin{array}{c} \begin{array}{ccc} x1 & x2 & x3 \end{array} \\ \begin{bmatrix} 4.0 & 2.0 & .60 \\ 4.2 & 2.1 & .59 \\ 3.9 & 2.0 & .58 \\ 4.3 & 2.1 & .62 \\ 4.1 & 2.2 & .63 \end{bmatrix} \end{array}$$

Vectorul mediu:

$$\bar{\mathbf{x}} = [4.10 \quad 2.08 \quad .604]$$

Covarianta dintre coordonatele x_i si x_j :

$$S_{ij} = \frac{1}{m}\sum_{k=1}^{m}\left(x_i^k - \bar{x}_i\right)\left(x_j^k - \bar{x}_j\right)$$

Matricea de varianta/covarianta:

$$\mathbf{S} = \begin{bmatrix} 0.025 & 0.0075 & 0.00175 \\ 0.0075 & 0.0070 & 0.00135 \\ 0.00175 & 0.00135 & 0.00043 \end{bmatrix}$$

Varianta coordonatei x_i

$$S_{ii} = \frac{1}{m}\sum_{k=1}^{m}\left(x_i^k - \bar{x}_i\right)\left(x_i^k - \bar{x}_i\right)$$

Kalman Filter

Problema:= calculul celei mai bune aproximari a starii sistemului , $\hat{\mathbf{x}}_k$, la momentul t_k ,luand in considerare estimarea starii prezise de model la momentul t_{k-1} si masuratorile $\mathbf{z}_k$ de la momentul t_k.

Algoritmul "Kalman Filter" [Trucco]:

$$P_k' = \Phi_{k-1} P_{k-1} \Phi_{k-1}^\top + Q_{k-1} \qquad \text{(A.11)}$$

$$K_k = P_k' H_k^\top (H_k P_k' H_k^\top + R_k)^{-1} \qquad \text{(A.12)}$$

covarianta inovatiei

$$\hat{\mathbf{x}}_k = \Phi_{k-1}\hat{\mathbf{x}}_{k-1} + K_k(\mathbf{z}_k - H_k \Phi_{k-1}\hat{\mathbf{x}}_{k-1}) \qquad \text{(A.13)}$$

inovatie

$$P_k = (I - K_k) P_k' (I - K_k)^\top + K_k R_k K_k^\top \qquad \text{(A.14)}$$

$\mathbf{K}_k$ - matricea de castig (gain)

$\mathbf{P}_k$, $\mathbf{P}_k$' – matricile de covarianta ale starilor $\hat{\mathbf{x}}_k$

Modele cantitative ale incertitudinilor starilor $\mathbf{X}_k$' si $\mathbf{X}_k$

$\mathbf{P}_k$' – covarianta estimari starii la t_k, $\hat{\mathbf{x}}_k' = \Phi_{k-1}\hat{\mathbf{x}}_{k-1}$ (peredictie) in momentul imediat anterior masuratorii $\mathbf{z}_k$

$\mathbf{P}_k$ – covarianta estimari starii la t_k, $\hat{\mathbf{x}}_k$ dupa integrarea masuratorii $\mathbf{z}_k$, cu predictia $\mathbf{x}_k$'

Kalman tracking

Urmarirea miscarii unui punct/trasaturi in imagine

Modelul sistem

Pozitia: $\mathbf{p}_k = [x_k, y_k]^\top$

Viteza: $\mathbf{v}_k = [v_{x,k}, v_{y,k}]^\top$

Vect. stare: $\mathbf{x} = [x_k, y_k, v_{x,k}, v_{y,k}]^\top$

$t_k = t_0 + k$ miscare cu viteza constanta $\Rightarrow$ model:

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \mathbf{v}_{k-1} + \boldsymbol{\xi}_{k-1}$$
$$\mathbf{v}_k = \mathbf{v}_{k-1} + \boldsymbol{\eta}_{k-1},$$

$$\mathbf{x}_k = \Phi_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}$$

$$\Phi_{k-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ matricea de tranzitie a starilor

$$\mathbf{w}_{k-1} = \begin{bmatrix} \boldsymbol{\xi}_{k-1} \\ \boldsymbol{\eta}_{k-1} \end{bmatrix}$$ zgomotul sistem – model aditiv, gausian cu medie 0, alb

Kalman tracking

Modelul masuratorii

$$\mathbf{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{bmatrix} + \boldsymbol{\mu}_k$$

μ_k – zgomotul masuratorii

Exemplu implementare OpenCV:
http://opencvexamples.blogspot.com/2014/01/kalman-filter-implementation-tracking.html

Tutorial Kalman filter (1D):
https://www.kalmanfilter.net/default.aspx

Tutorial Kalman filter pt. urmarirea miscarii unui punct / obiect in spatiul 2D (miscare cu acceleratie constanta)

http://www.youtube.com/watch?v=GBYW1j9lC1I

http://www.youtube.com/watch?v=Jq8HcIar68Y

Aplicatii tracking

Beneficii Tracking:

- Rezolavarea problemei datelor/ masuratorilor lipsa, ocluziilor etc.
- Reducerea timpului de cautare/procesare
- Estimarea parametrilor dinamici ai miscarii (viteza, traiectorie etc.)

Alte metode/filtre folosite in tracking:

- particle filtering, mean-shift

Exemple aplicatii Tracking:

- Urmarire trasaturi (colturi etc.) sau obiecte in imagini succesive
- Preprocesare in recunosterea scrisului de mana (determinarea randurilor cu text)
- ……..

Bibliografie

[1] Trucco E., Verri A, Introductory techniques for 3D Computer Vision, Prentice Hall, 1998, pp. 198 – 203 si 328 – 333

[2] Pyramidal Implementation of the Lucas Kanade Feature Tracker - Description of the algorithm, Jean-Yves Bouguet, Intel Corporation Microprocessor Research Labs, jean-yves.bouguet@intel.com
http://robots.stanford.edu/cs223b04/algo_tracking.pdf

[3] B. Kisacanin, V. Pavlovic, T.S. Huang, Real-Time Vision for Human-Computer Interaction, Springer 2005, pp.27-29