20170243 심재윤

#1.

#1-a.

$$\eta_{ij} = \begin{cases} 1, & \text{if } x_i \text{ and } x_j \text{ are same label pair.} \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{il} = \begin{cases} 1, & \text{if } x_i \text{ and } x_l \text{ are same label pair} \\ 0, & \text{otherwise.} \end{cases}$$

이 둘은 indicator variable로 label이 같은지와 관련한 variable이다.

#1-b.

first term : positive pair의 거리 합을 minimization 하고 싶다.

x_i와 x_j가 같은 class이면 pull 한다.

second term : pair data에 관한 것이 아니고 triplet data에 관한 것이다. (same/different)

x_l을 x_i로 부터 push 한다 (만약 다른 class 라면)

negative pair을 멀리 보내고 싶은 것이다.

#1-c.

hinge function은 triplet 관계에 대해서 이용할수 있다. 그래서 negative pair의 거리가 동적으로 결정이 될수 있다. 이는 positive pair와 margin 에 의해서 영향을 받고, 결국 large margin nearest neighbor는 hinge function 사용한 것이다. 이는 결국 more flexible 하다.

20170243 심재윤

#2.

#2-a.

가정 ① Color consistancy - I_{t-1}과 I_t에서 점은 같다.

가정 ② small motion. - 점들이 그리 크게 멀리 이동하지 않는다.

#2-b. ①번을 사용한다. Brightness constancy 가정을 사용하여 유도할수 있다.

$$I(x,y,t-1) = I(x+u, y+v, t) \approx I(x,y,t) + u \cdot \nabla_x \cdot I(x,y,t) + v \cdot \nabla_y \cdot I(x,y,t)$$

$$u \cdot \nabla_x \cdot I(x,y,t) + v \cdot \nabla_y \cdot I(x,y,t) \approx I(x,y,t-1) - I(x,y,t) = -\nabla_t \cdot I(x,y,t)$$

$$\therefore \begin{bmatrix} \nabla_x \cdot I(x,y,t) & \nabla_y \cdot I(x,y,t) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\nabla_t I(x,y,t)$$

#2-c.

$$\underbrace{\begin{bmatrix} \nabla_x I(x_1,y_1,t) & \nabla_y I(x_1,y_1,t) \\ \nabla_x I(x_2,y_2,t) & \nabla_y I(x_2,y_2,t) \\ \vdots & \vdots \\ \nabla_x I(x_9,y_9,t) & \nabla_y I(x_9,y_9,t) \end{bmatrix}}_{\boxed{A}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\boxed{v}} = \underbrace{\begin{bmatrix} \nabla_t I(x_1,y_1,t) \\ \nabla_t I(x_2,y_2,t) \\ \vdots \\ \nabla_t I(x_9,y_9,t) \end{bmatrix}}_{\boxed{b}} \Leftrightarrow Av = b$$

$$\downarrow$$

$$A^TA\,v = A^Tb$$

#2-d.

① A^TA는 invertible 해야 한다. (= A^TA는 full-rank 이다. = A^TA의 eigenvalue는 non-zero이다)

② λ_1 / λ_2는 너무 크지 않아야 한다. ($\lambda_1 > \lambda_2$ 가정) → 둘의 차이가 거의 없다.

③ A^TA의 eigen value는 작으면 안된다.

20170243 심재윤

#3.

#3-a.

probabilistic – particle filtering (generative)
(resampling → state transition → observation)

deterministic – tracking by detection (discriminative)
(online learning manner)

#3-b.

1st order Markov assumption

→ 현재 state인 x_t는 이전 state인 x_{t-1}에만 의존한다. 그리고 현재 observation z_t는 현재 state인 x_t에만 의존한다.

#3-c

pros – 일단 모든 video frame을 다 보고서 거기서 detection 했다 가정한다. 그래서 occlusion이나 이런한 문제를 잘 처리할수 있다.

cons – object detection model이 전체 video frame에 필요하다. → costly.
offline 방식이다.

20170243 심재윤

3-d. Soccer ball.  speed, direction. w/ illumination change. real time system.

- state space : 축구공의 location을 time t에서 vector로 정의한다.

  $X_t = [x, y, s]^T \in R^3$

- appearance model : sample candidate들을 찾아 축구공과 비슷한 것을 선택한다.
  선택된 Bounding box는 타겟으로 하는 축구공과 매우근접해야 할것이다.

- motion model : particle filtering 등과 같이 probabilistic tracking 방식을
  사용하고자 축구공의 sample들을 random하게 sampling 해준다.
  미리 여러 가설들을 가지고 있고 계속해서 weight 따라 update가
  가능하다.

2017○243 심재윤

#4.

#4-a.

$$\min \sum_{i=1}^{n} w_i (ax_i + b - y_i)^2 = \min \| W(Xa - y) \|^2$$

$$\frac{d \| W(Xa-y) \|^2}{da} = 2(WX)^T W(a^T X - y) = 0.$$

$$\therefore a = (X^T W^T W X)^{-1} X^T W^T y$$

#4-b.

① vertical distance 경우 rotation invariant 하다.

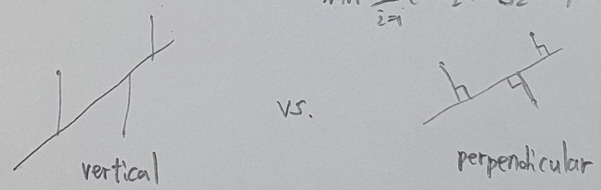
② vertical distance 는 infinit로 가게 되는데 이러면 numerical unstable 하다. (기울기가 심할시).

ex>

#4-c.

ordinary least square는 vertical distance 이지만, total linear least square는 perpendicular distance를 사용한다.

$$\min \sum_{i=1}^{n} (ax_i + by_i + c)^2 = \min \| Xa \|^2$$



#4-d.

위의 objective function을 그대로 optimization 하면 trivial solution 인 $a = [0,0,0]^T$가 존재한다. 이는 적절하지 않고, 우리가 원하는 결과도 아니다. 그래서 Constraint가 필요하다.

($kax + kby + kc = 0 \Leftrightarrow Xa = 0$ 을 풀어야 한다.)

#4-e

가정> $a^* = \min \| Xa \|^2$ subject to $\| a \| = 1$ to avoid trivial solution

→ eigen decomposition 사용. $\| Xa \|^2 = (Xa)^T (Xa) = a^T X^T X a = a^T (V^T D V) a = (Va)^T D (Va)$

→ new objective function 사용: $\min \| Xa \|^2 = \min k^T D k$ subject to $\| k \| = 1$

& D = diagonal matrix with eigenvalue, $k = Va$.

→ $a = V^T k$ 이고 $k = [0, 0, \cdots, 0, 1]^T$.

$X^T X$에 eigen decomposition을 하면 a^*는 가장 작은 eigen value를 갖게하는 eigen vector 이다.

20170243 심재윤

#5.-a.

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$ → location change

dof = 2

t_x : x축 이동
t_y : y축 이동.

rotation.

$$E = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$ → translation.

dof = 3

θ : 각도

$$S = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$ s: scale change

dof = 4.

s : 스케일 값.

non-singular transform

$$A = \begin{bmatrix} a_1 & a_2 & t_x \\ a_3 & a_4 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

dof = 6.

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

dof = 8.

→ homography.

2017024 3 심재윤

#5-b. DoF(M) = 11. & 적어도 6개의 world image point correspondence 필요

#5-c.

$$m^* = \underset{m}{\operatorname{argmin}}\ m^T A^T A m \qquad \|m\| = 1.$$

$$m = [m_{11}, m_{12}, \cdots, m_{34}]^T$$

$$A = \begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -\bar{u}_1 x_1 & -\bar{u}_1 y_1 & -\bar{u}_1 z_1 & -\bar{u}_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -\bar{v}_1 x_1 & -\bar{v}_1 y_1 & -\bar{v}_1 z_1 & -\bar{v}_1 \\ & & & & & & & \vdots & & & & \\ x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -\bar{u}_n x_n & -\bar{u}_n y_n & -\bar{u}_n z_n & -\bar{u}_n \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -\bar{v}_n x_n & -\bar{v}_n y_n & -\bar{v}_n z_n & -\bar{v}_n \end{bmatrix}$$

#5-d.

$$m^* = \underset{m}{\operatorname{argmin}}\ m^T A^T A m \quad \text{subject to } \|m\| = 1.$$

→ e_i 를 A^TA의 eigen vector 라 하고, λ_i를 A^TA의 eigen value로 한다.

→ unit vector x를 놓고 이를 e_i들의 linear combination으로 적으면

$$x = u_1 e_1 + \cdots + u_{12} e_{12} \quad \& \quad \sum_{i=1}^{12} u_i^2 = 1$$

→ $x^T A^T A x - e_1^T A^T A e_1 = (u_1 \lambda_1)^2 + \cdots + (\lambda_{12} u_{12})^2 - \lambda_1^2 \geq 0.$

→ $x^T A^T A x - e_1^T A^T A e_1 \geq 0.$

→ $m^* = e_1$