Tính toán Homography

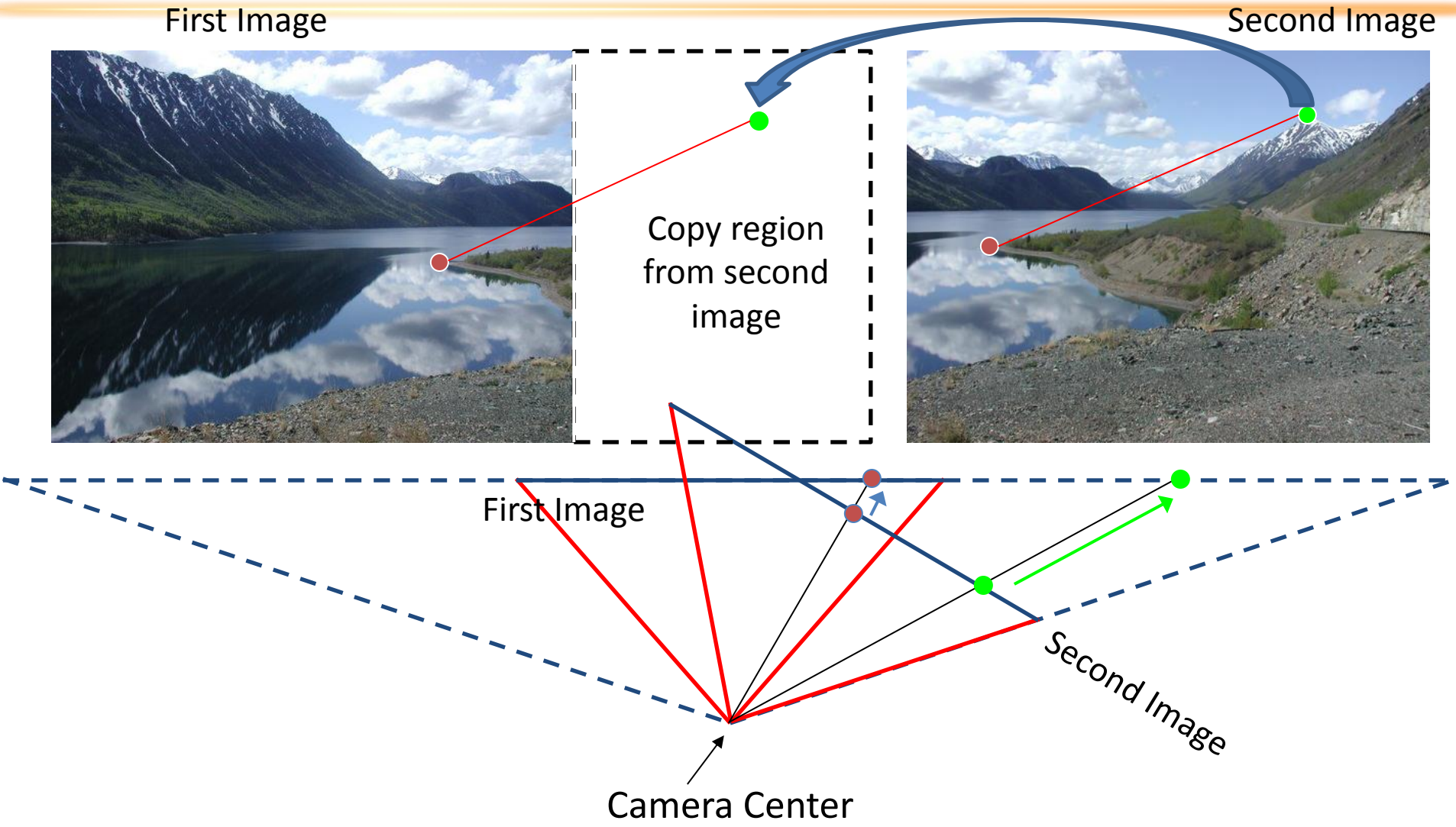
NHẬP MÔN THỊ GIÁC MÁY TÍNH

Trình bày: TS Trần Thái Sơn; Email: ttson@fit.hcmus.edu.vn

Phép biến đổi Homography

- Cách chuyển đổi ngược từ một mặt phẳng chiếu đến mặt khác mà đảm bảo từ đường thẳng đến đường thẳng.
- Bất kỳ 2 hình ảnh nào trong mặt phẳng 2 chiều đều liên quan nhau bằng 1 phép chuyển đổi homography.

Các góc nhìn khi xoay camera



Giải thuật

- 1. Tìm điểm trọng yếu: sử dụng đặc trưng SIFT.
- 2. So khớp keypoint: dựa vào thuật toán láng giềng gần nhất (kNN) dựa trên độ đo khoảng cách Euclid. Để tăng độ mạnh, thuật toán sẽ loại bỏ các điểm có tỉ lệ khoảng cách đến điểm gần nhất và gần thứ 2 lớn hơn 0.8.
- 3. Ước lượng homography với 4 điểm tìm được ở bước 2 (sử dụng thuật toán RANSAC).
- 4. Chiếu lên một mặt phẳng.

Tính toán homography

- Ta có 4 điểm khớp từ bước 2. Cách tính:
- Direct Linear Transform (DLT):

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \qquad \mathbf{x}' = \begin{bmatrix} w'u' \\ w'v' \\ w' \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

$$\begin{bmatrix} -u & -v & -1 & 0 & 0 & 0 & uu' & vu' & u' \\ 0 & 0 & 0 & -u & -v & -1 & uv' & vv' & v' \end{bmatrix} \mathbf{h} = \mathbf{0} \qquad \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}$$

Tính toán homography

- Direct Linear Transform

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1u_1' & v_1u_1' & u_1' \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v_1' & v_1v_1' & v_1' \\ & & & & \mathrm{M} & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv_n' & v_nv_n' & v_n' \end{bmatrix} \mathbf{h} = \mathbf{0} \Rightarrow \mathbf{Ah} = \mathbf{0}$$

- Sử dụng SVD: $\boldsymbol{UDV^T = A}$
- $\boldsymbol{h = V}_{\text{nhỏnhất}}$ (cột của V có trị riêng nhỏ nhất)

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \mathrm{M} \\ h_9 \end{bmatrix} \Rightarrow \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

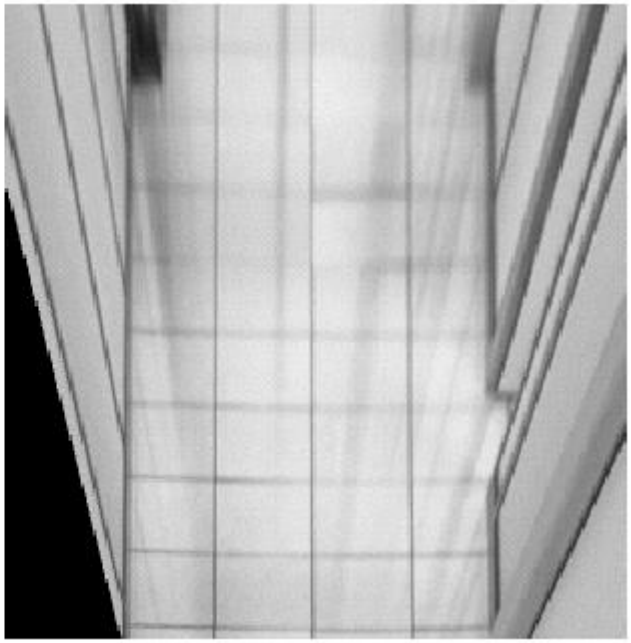
RANSAC: RANdom SAmple Consensus

- Có thể sử dụng nhiều 4 điểm để tìm ra tham số mô hình, nhưng không chắc chắn nghiệm nào là đúng.
- Thuật toán RANSAC:
- Lặp N lần:
  1. Chọn mẫu ngẫu nhiên.
  - Chỉ chọn số điểm vừa đủ để tìm được các tham số.
  2. Xây dựng mô hình với mẫu nhiên.
  3. Đếm bao nhiêu điểm phù hợp với mô hình.
- Ước lượng tốt nhất khi số điểm phù hợp nhiều nhất.
  - Có thể sử dụng những điểm đã phù hợp rồi để chắc chắn ước lượng.

Tính toán homography

- Tính ma trận homography H:
- Tự động ước lượng homography với RANSAC:
  1. Chọn số lượng điểm lấy mẫu N.
  2. Chọn ngẫu nhiên 4 điểm trùng khớp tiềm năng.
  3. Tính **H** sử dụng DLT.
  4. Chiếu các điểm từ **x** sang **x'** cho mỗi cặp trùng khớp : $\mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$
  5. Đếm số điểm với khoảng cách chiếu < t
     1. Ví dụ: t = 3 pixels.
  6. Lặp các bước từ 2 đến 5 N lần.
     1. Chọn **H** với nhiều điểm phù hợp mô hình nhất (most inliers).

Ví dụ



Ví dụ

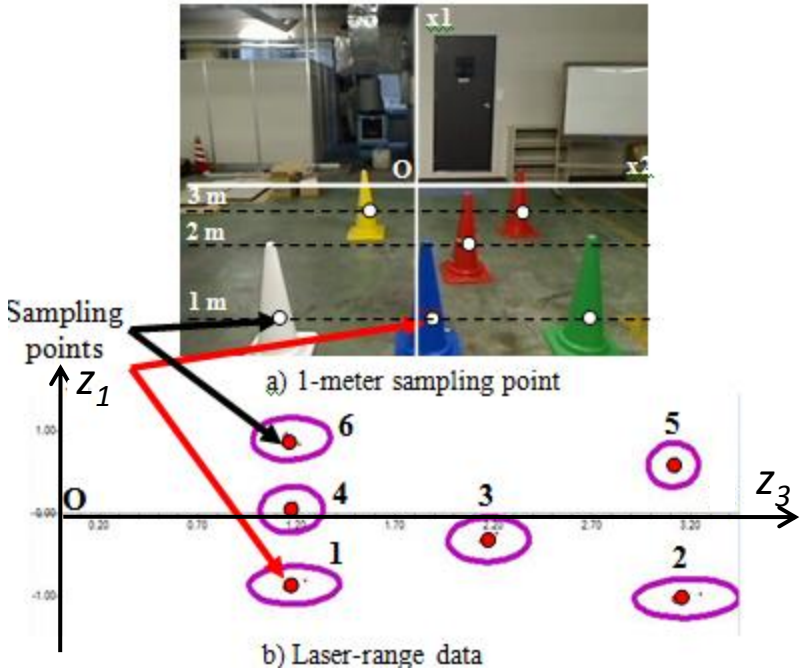


Table 2: Coordinates of landmark points

Point order	Image (pixels)		Laser (mm)	
	x1	x2	z3	z1
1 (green)	-89	-114	1189	-855
2 (far red)	-18	-70	3181	-1016
3 (near red)	-41	-34	2157	-300
4 (blue)	-89	-8	1174	57
5 (yellow)	-18	33	3114	593
6 (white)	-89	94	1175	912

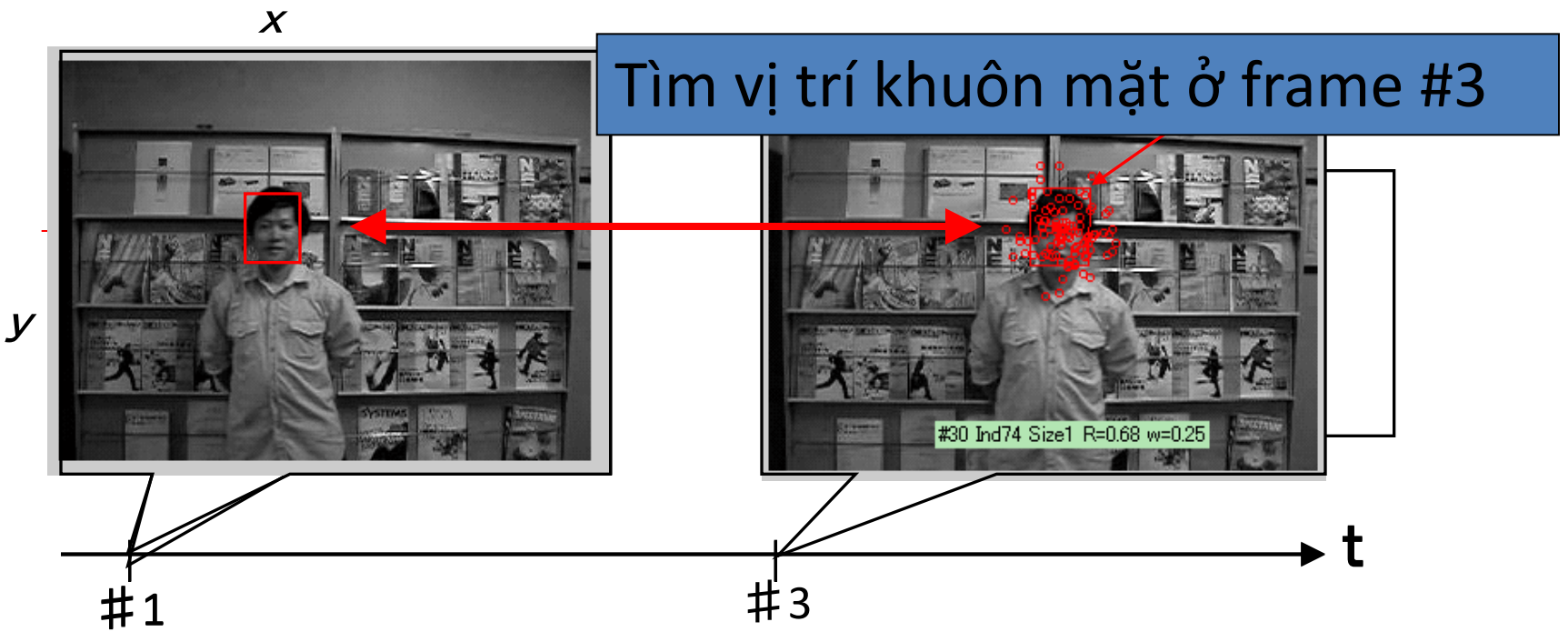
$$\begin{pmatrix} z_3 \\ z_1 \\ 1 \end{pmatrix} = H \begin{pmatrix} x1 \\ x2 \\ 1 \end{pmatrix}$$

$$rad = \arctan(\frac{z_1}{z_3}) \text{ or } \arctan(\frac{-z_1}{z_3}) + \pi$$

z_2 : distance from the surface to laser device

Các vấn đề trong Tracking

- Theo vết khuôn mặt ở khung hình tiếp theo theo thời gian thực sử dụng **Particle Filter**.



Particle Filters

- Biểu diễn đối tượng bằng cách lấy mẫu ngẫu nhiên.
- Thuộc dạng phương pháp non-Gaussian, nonlinear.
- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter

- Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
- Computer vision: [Isard and Blake 96, 98]
- Dynamic Bayesian Networks: [Kanazawa et al., 95]

Thuật toán Particle Filter

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \varnothing, \quad \eta = 0$
3. **For** $i = 1 \text{K } n$
4. Lấy lại tập mẫu $j(i)$ từ phân phối rời rạc được cho bởi w_{t-1}
5. Lấy mẫu x_t^i từ $p(x_t \mid x_{t-1}, u_{t-1})$ sử dụng $x_{t-1}^{j(i)}$ và u_{t-1}
6. $w_t^i = p(z_t \mid x_t^i)$ ***Tính hệ số quan trọng***
7. $\eta = \eta + w_t^i$ ***Cập nhật hệ số chuẩn hóa***
8. $S_t = S_t \cup \{[x_t^i, w_t^i]\}$ ***Thao tác chèn***
9. **For** $i = 1 \text{K } n$
10. $w_t^i = w_t^i / \eta$ ***Chuẩn hóa trọng số.***

Bước khởi tạo: Xây dựng tập Particle

Sử dụng N mẫu (N=500)

X1, Y1	X2, Y2	X3, Y3	X4, Y4	----------	X500, Y500

$$\begin{pmatrix} X_1 & Y_1 \\ \dots & \dots \\ X_{500} & Y_{500} \end{pmatrix} = U_{(500\times 2)} \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} + \begin{pmatrix} x_1 & y_1 \\ \dots & \dots \\ x_1 & y_1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Ngẫu nhiên từ phân phối đều [-1, 1]

Trọng số (N=500) $Wi = \frac{1}{N}$

w1	w2	w3	w4	----------	w500

Tính toán trên Particle

z_k: Khoảng cách trung bình tính theo không gian Euclid.

P(x(k)|x(k-1)): Propagation of position from t=k-1 to t=k.

Particle Calculation (Sequential Importance Sampling):

$$\{x_k^i, w_k^i\}_{i=1}^{N_s} = SIS(\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k)$$

For $i = 1:N$

- Draw x_k^i from $\{x_{k-1}\}$ by distribution u_{t-1}

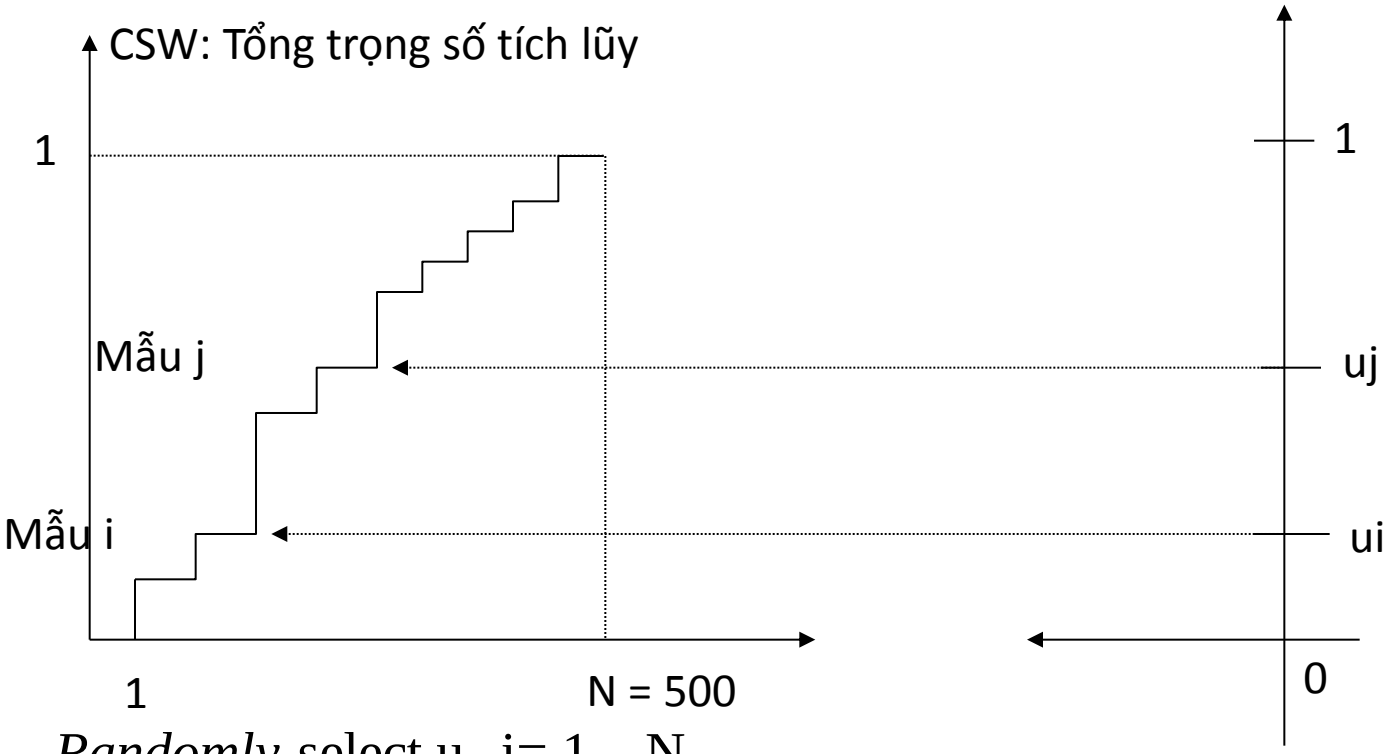
- Update weight

$$w_k^i \approx w_{k-1}^i p(z_k \mid x_k^i) \text{ or } w_k^i \approx p(z_k \mid x_k^i)$$

End For

Normalize weight $w_k^i = \dfrac{w_k^i}{\sum_i w_k^i}$

Lấy lại tập Particle



$Randomly$ select $u_i, i = 1,..,N$

$$NewSample_i = Sample_t \ \ with \ \ t = \arg\min_k \{CSW(k)\}_{k=1,..,N} > u_i)$$

$$NewWeight_i = \frac{1}{N},$$

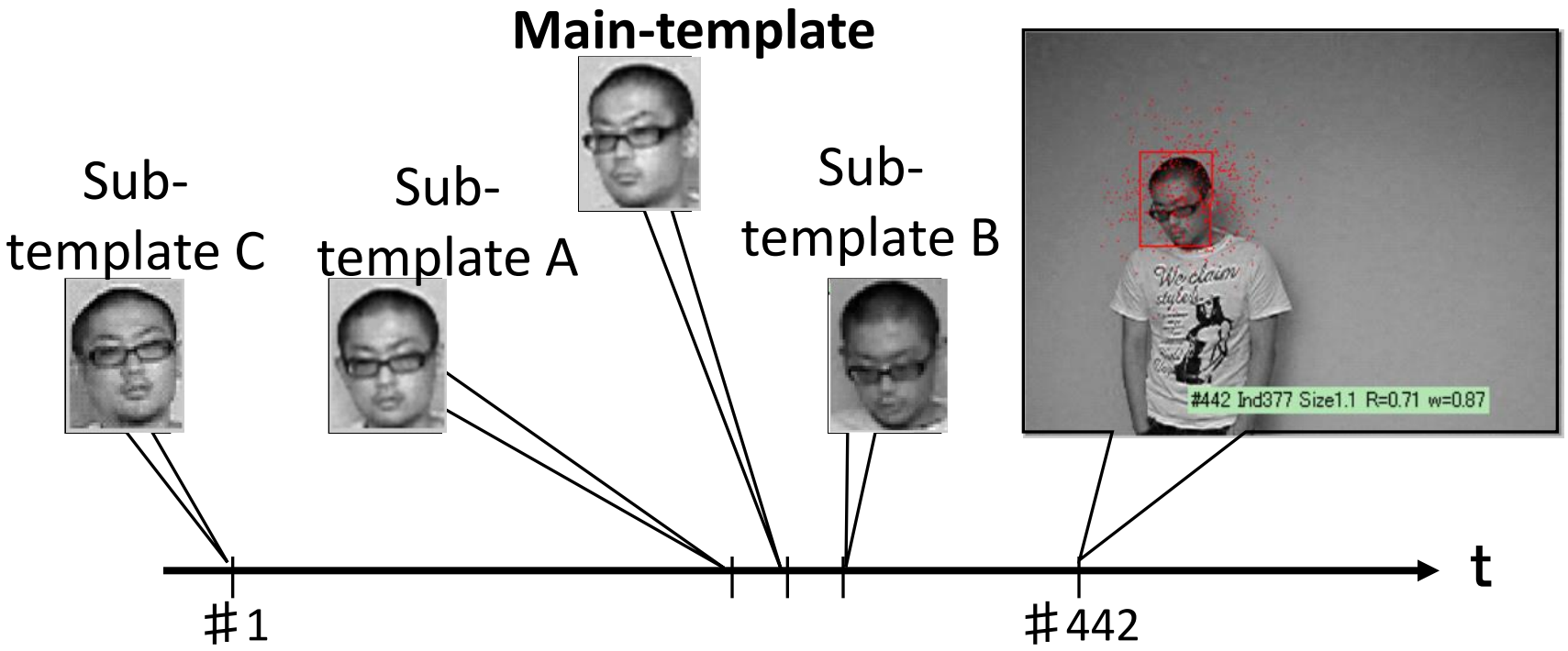
Các chiến thuật

- ISI (Important Sampling Sequence)
- **Trích nền**
  - Xác định vị trí nền trong ảnh.
  - Cập nhật trạng thái các vị trí nền.

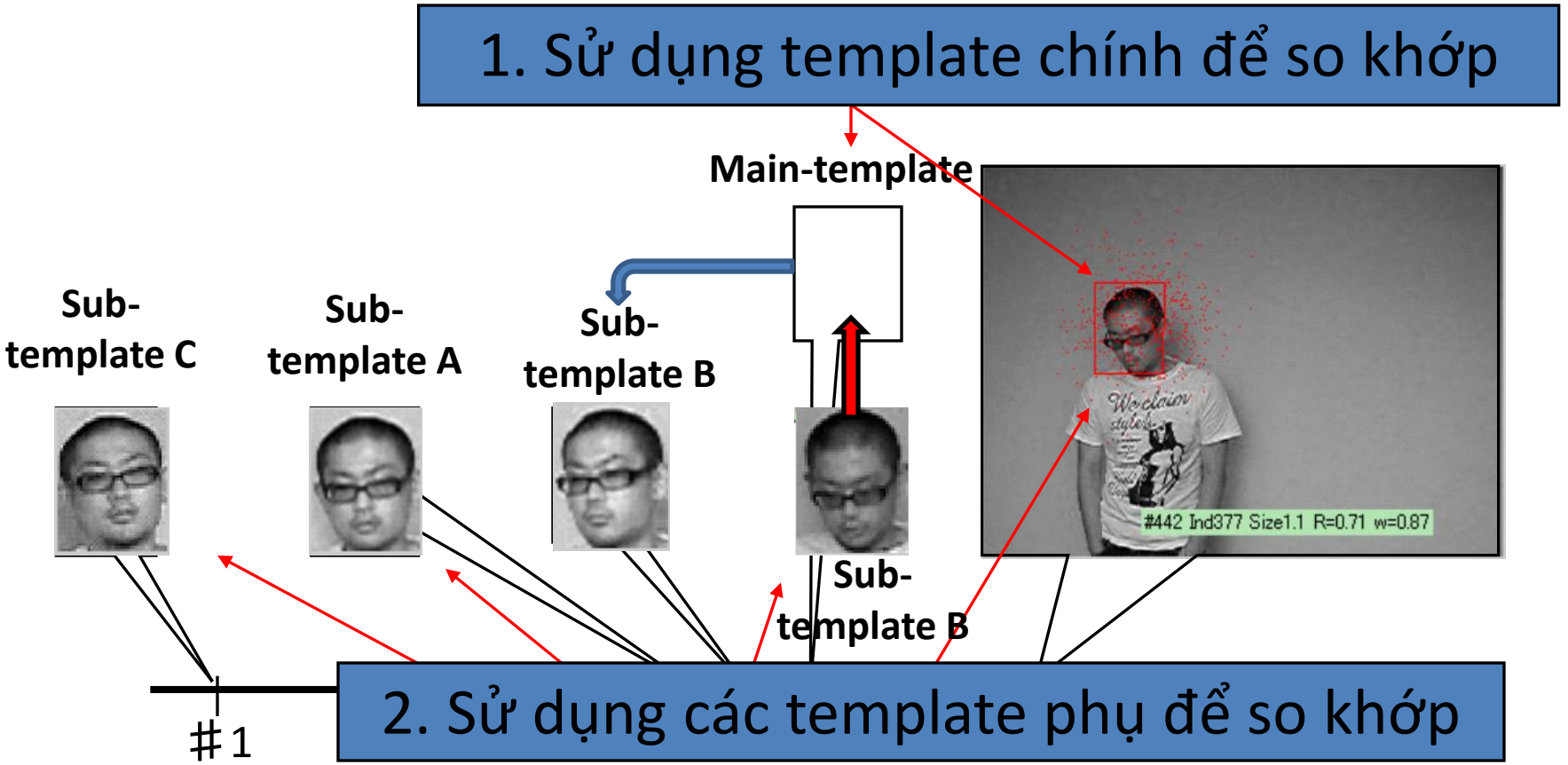
  Giữ các điều kiện cho vùng tìm kiếm.
- **Sử dụng mẫu thay đổi thích nghi (**Adaptive Template)
  - Sử dụng các template phụ.
  - Sử dụng template cùng với template chính.
- Tiếp tục lấy mẫu (resample).

Quá trình lặp

Adaptive Template



Adaptive Template



Sử dụng Adaptive Template

23 degree

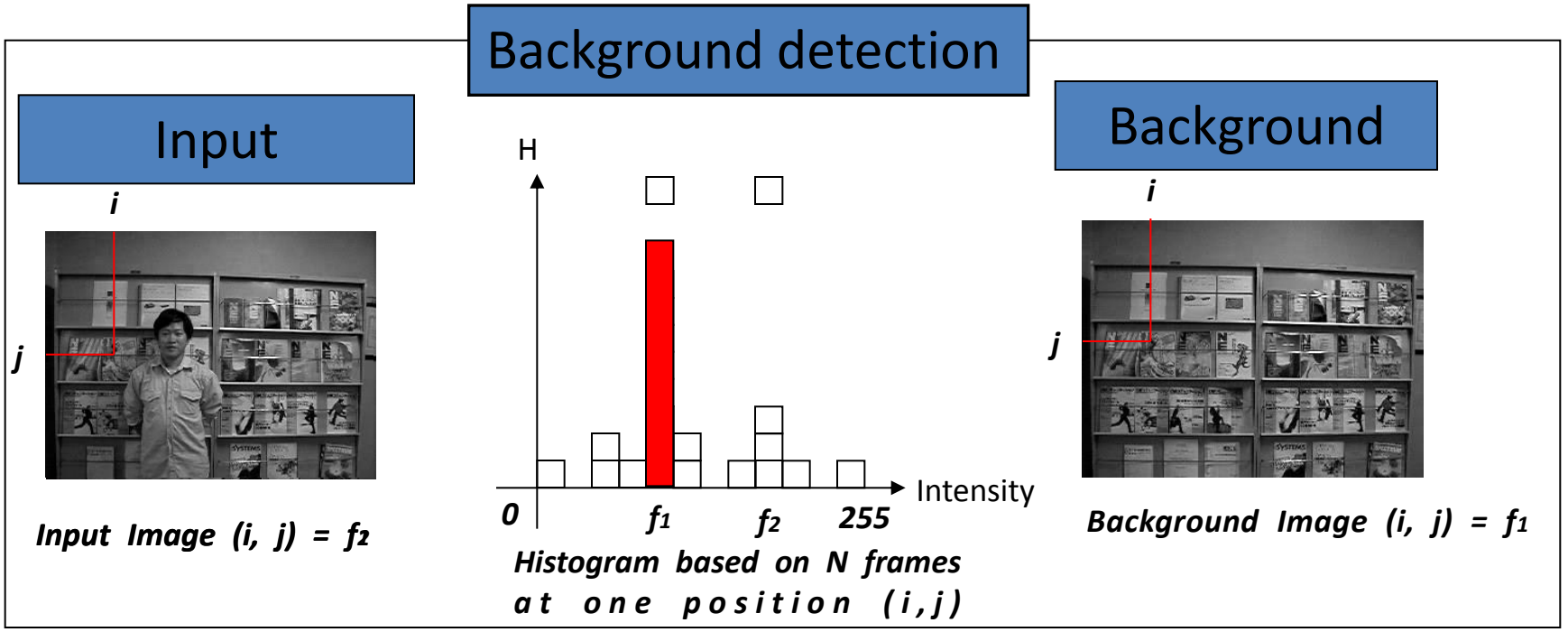
Các truyền thống

45 degree

Adaptive template

Phát hiện nền

- Thao tác phát hiện nền được sử dụng khi máy ảnh luôn đặt ở một vị trí cố định trong khi đối tượng di chuyển.



Phát hiện nền

Có vân ảnh trong nền

So khớp → Theo vết bị lỗi

Cách truyền thống

Lỗi truyền thống

Phát hiện nền

Có vân ảnh trong nền

So khớp → Tracking error

Sử dụng thông tin nền

Truyền thống

Thông tin nền có thể giúp sửa đổi lỗi nếu có

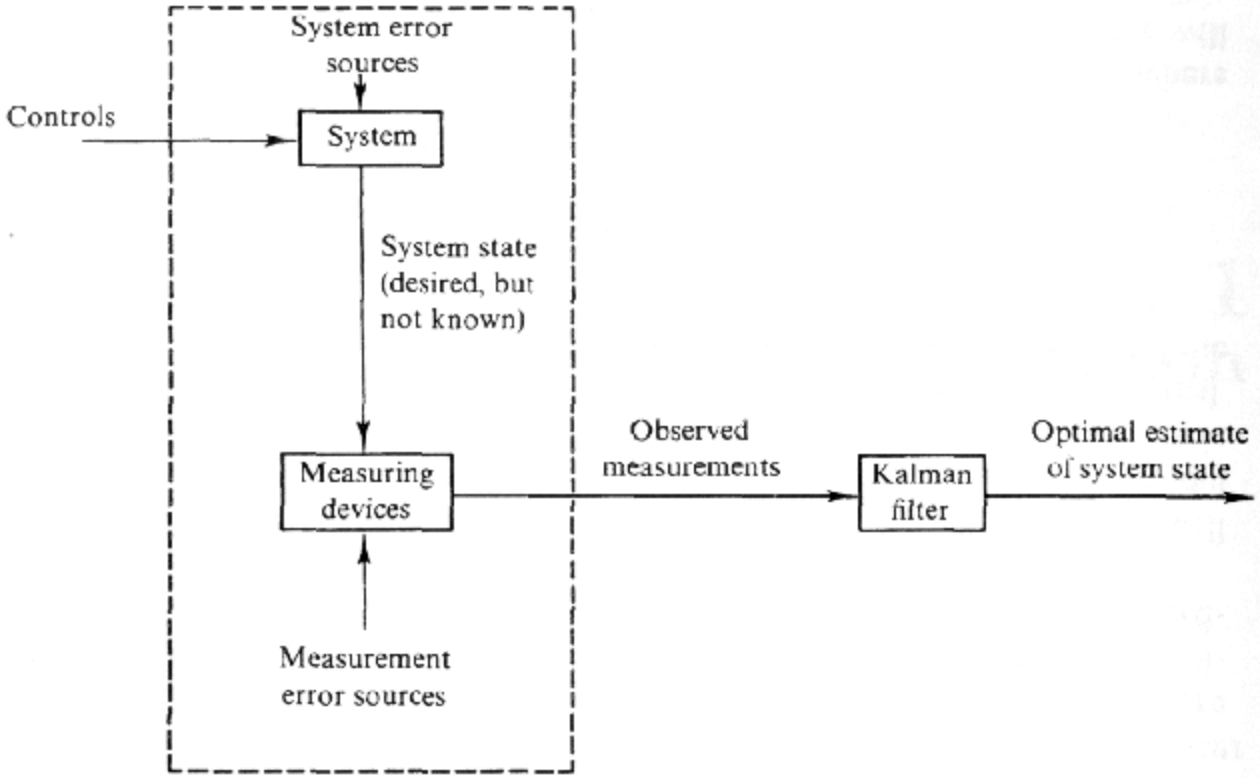
Bộ lọc Kalman

- Nếu xuất hiện nhiễu trong hệ thống và trong sự đo lường.
  - Luôn luôn có lỗi trong các thiết bị đo lường.
    - Lỗi trong các vi mạch.
    - Tín hiệu không rõ ràng.
    - Độ phân giải hữu hạn của hệ thống.
  - Mô hình của hệ thống g()
    - Mô hình hệ thống không chính xác.
    - Sự không chắc chắn ở bước tính toán trạng thái trước dẫn đến độ đo không chính xác.

Bộ lọc Kalman

- Bộ lọc Kalman tối ưu vì bao gồm:
  - Tri thức của hệ thống và các thiết bị đo lường.
  - Biểu diễn theo thống kê cho nhiễu hệ thống, lỗi đo lường, và các mô hình động không chắc chắn.
  - Chứa bất kỳ thông tin nào về điều kiện khởi đầu của các biến quan tâm.

Bộ lọc Kalman



Ứng dụng bộ lọc Kalman điển hình

Bộ lọc Kalman

- Công thức bộ lọc Kalman tuyến tính:
- Giả sử ta có công thức trạng thái và độ đo.

$$a_i = Aa_{i-1} + Bu_{i-1} + w_{i-1}$$

$$x_i = Ha_i + v_i$$

Cả hai w_{i-1} and v_i là lỗi (hay nhiễu) và
giả sử là biến ngẫu nhiên thuộc phân phối chuẩn

Bộ lọc Kalman

- Định nghĩa R, Q:

  R : ma trận hiệp phương độ lỗi đo lường cov(vi)
  Q : ma trận hiệp phương sai độ lỗi trạng thái mô hình cov(wi)

  Cả R và Q đều được tính toán bổ sung trên từng bước.

- Định nghĩa P_i:

  P_i : Ma trận hiệp phương sai của lỗi ước lượng trạng thái.

  P_i có thể được tính bằng công thức $E[(a_i - \hat{a}_i)(a_i - \hat{a}_i)^T]$

Bộ lọc Kalman

- Bộ lọc Kalman tuyến tính với giá trị Pi tối ưu:
- Bước dự đoán:

$$\bar{a}_i = A\hat{a}_{i-1} + C \quad \text{with } C = Bu_{i-1} = \text{const}$$

$$\bar{P}_i = AP_{i-1}A^T + Q$$

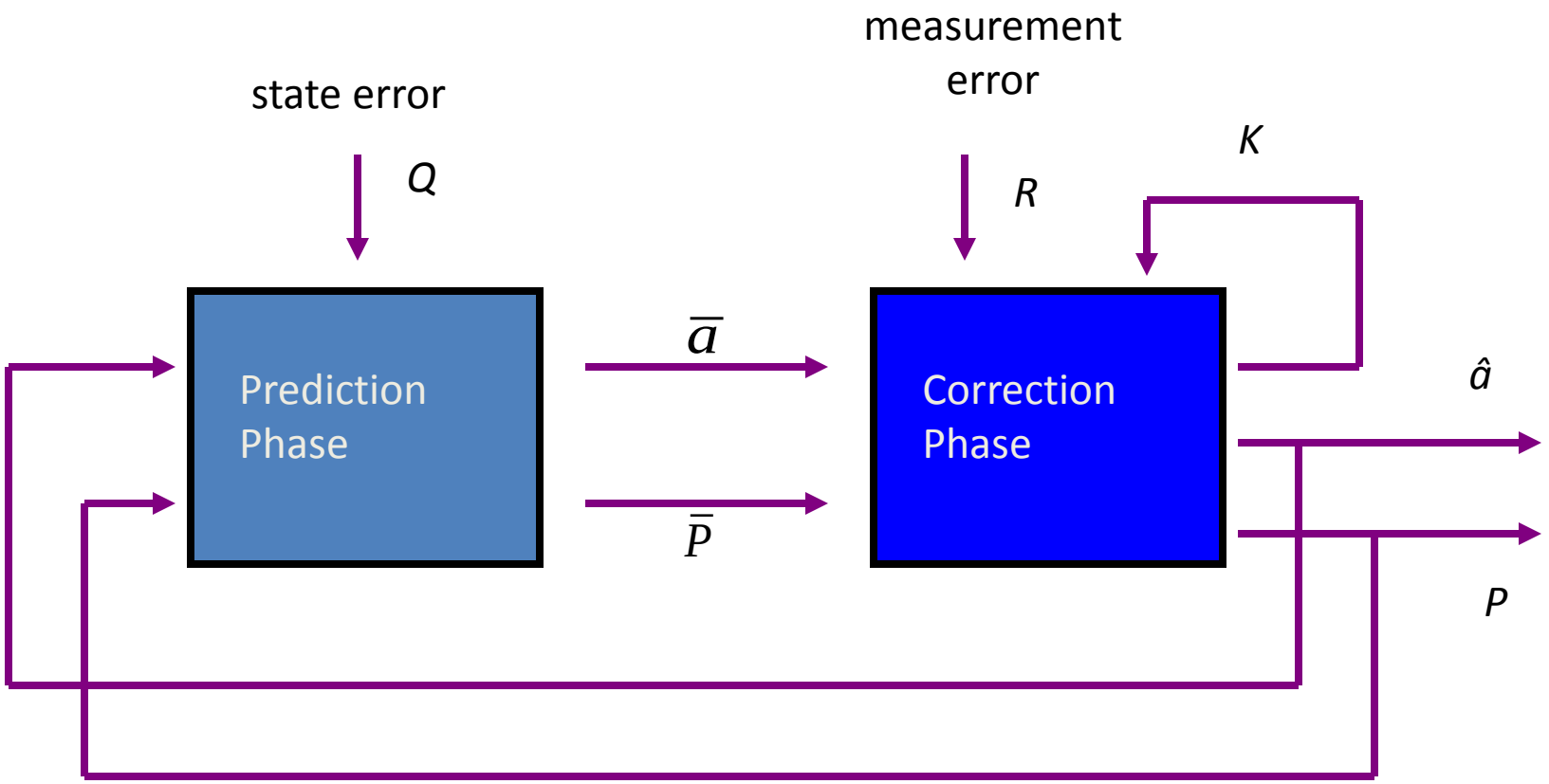
- Bước sửa lỗi:

$$K_i = \bar{P}_i H^T (H\bar{P}_i H_i^T + R)^{-1}$$

$$\hat{a}_i = \bar{a}_i + K_i(x_i - H\bar{a}_i)$$

$$P_i = \bar{P}_i - K_i H\bar{P}_i$$

Bộ lọc Kalman



Bộ lọc Kalman

- Điều kiện ở bước khởi đầu:

1. Định nghĩa trạng thái A,H,C.
2. Khởi tạo P là P_0.
3. Khởi tạo $\bar{a}_0$
4. Định nghĩa hiệp phương sai lỗi R và Q.

- Bộ lọc Kalman bắt đầu bằng phép tính:

$$\bar{P}_1 = AP_0A^T + Q$$

Ví dụ: theo vết bóng 2D

- Đầu vào: 1 dãy các ảnh với thứ tự thời gian biết rõ.
- Mục đích: cho 1 trái bóng ở một ảnh, theo vết chuyển động của nó trên dãy ảnh.
- Giả sử:
  - Bóng tồn tại trong ảnh.
  - Luôn luôn di chuyển theo hướng ngang với một hướng nhất định.

Ví dụ: theo vết bóng 2D

- Miêu tả vấn đề:
  - Xác định "trạng thái hệ thống"
    - Tìm một biểu diễn cho chuyển động bóng.
    - Tập các thông tin: vị trí bóng, bán kính, gia tốc tối đa.
    - Cho phép có lỗi trong hệ thống.
  - Tìm phương trình độ đo:
    - Được tính bằng sự so khớp của vị trí bóng thực tế so với dự đoán.

Ví dụ: theo vết bóng 2D

Biểu diễn bóng với [tâm-x, *bán kính, tốc độ, bán kính*]
Nên phương trình di chuyển có thể viết như sau:

$$a_i = Aa_{i-1} + C, \text{ where A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$a_i = \begin{bmatrix} x \\ r \\ x^{/} \\ x^{//} \end{bmatrix}, C = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 6 \end{bmatrix}$$

Ví dụ: theo vết bóng 2D

Hàm độ đo có thể được viết như sau:

$$x_i = Ha_i + v_i$$

Để hoàn chỉnh mô hình ta cần thêm ma trận hiệp phương sai Q và R

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, P_0 = 100 \times I_{4\times4}, Q = 0.01 \times I_{4\times4}$$

$$R = \begin{bmatrix} 0.28 & 0.0045 \\ 0.0045 & 0.0045 \end{bmatrix}$$

Thuật toán Mean-Shift

- Mean Shift [Che98, FH75, Sil86].
  - Một thuật toán liên tục dịch chuyển một điểm dữ liệu đến điểm trung bình của tập các điểm lân cận.
  - Mục đích giống như gom cụm.
  - Sử dụng tốt cho các mục đích: gom cụm, tìm kiếm mô hình, ước lượng mật độ xác suất, theo vết ...

Thuật toán Mean-Shift

- Xét tập **n** điểm dữ liệu **S**: $\mathbf{x_i}$ trong không gian Euclid **X**.
- Đặt **K(x)** là hàm kernel để tính sự đóng góp của **x** vào việc ước lượng giá trị trung bình **m** của tập.
- Công tính tính giá trị trung bình **m** tại **x** với kernek **K**:

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)\,\mathbf{x}_i}{\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)}$$

- Khác biệt: m(x) –x được gọi là mean shift (dịch chuyển trung bình).
- TT mean shift: lặp lại việc di chuyển điểm đến trung bình của nó.
- Mỗi vòng lặp: x ← m(x).
- Thuật toán dừng khi m(x) = x.
- Chuỗi: x , m(x), m(m(x)) .... được gọi quỹ đạo của x.
- Nếu trung bình được tính từ nhiều điểm, mỗi lần lặp thì cập nhật vị trị cho tất cả các điểm.

PP theo vết Mean Tracking

- Ý tưởng cơ bản [CRM00]:
  - Mô hình đối tượng sử dụng mật độ xác suất giá trị điểm ảnh (màu sắc).
  - Theo vết đối tượng trong video sử dụng so khớp xác suất màu sắc.
  - Sử dụng mean shift để ước lượng mật độ màu sắc và điểm đích.

Mô hình đối tượng

- Với xi, i = 1, ..., n: ký hiệu vị trí điểm ảnh của mô hình với tâm 0.
- Biểu diễn phân phối màu sắc bởi histogram rời rạc m-bin.
- Với b(xi) ký hiệu bin màu tại xi.
- Giả sử kích cỡ của mô hình được chuẩn hóa, bán kính kernel h = 1.
- Xác suất q của màu u trong mô hình:

$$q_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i\|^2)\, \delta(b(\mathbf{x}_i) - u)$$

- C là hằng số chuẩn hóa

$$C = \left[\sum_{i=1}^{n} k(\|\mathbf{x}_i\|^2) \right]^{-1}$$

- Kernel k tính trọng số ảnh hưởng bởi khoảng cách đến tâm điểm.

Mô hình đối tượng

- Với δ là hàm Kronecker delta.

$$\delta(a) = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$$

- Tức là màu **u** sẽ đóng góp lượng $k(\|\mathbf{x}_i\|^2)$ vào q_u nếu $b(\mathbf{x}_i) = u$

Các điểm ứng viên

- Tập $\mathbf{y_i}$: i = 1, ..., n_h, ký hiệu vị trí điểm ảnh của các điểm đích ứng viên **y**.
- Xác suất **p** của màu **u** ở điểm đích là:

$$p_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{y}_i}{h}\right\|^2\right) \delta(b(\mathbf{y}_i)-u)$$

- $\mathbf{C_h}$ là hằng số chuẩn hóa:

$$C_h = \left[\sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{y}_i}{h}\right\|^2\right)\right]^{-1}$$

Các điểm ứng viên

- Hệ số Bhattacharyya ρ

$$\rho(p(\mathbf{y}), q) = \sum_{u=1}^{m} \sqrt{p_u(\mathbf{y})\, q_u}$$

- ρ là góc cosin giữa 2 vector $(\sqrt{p_1}, \ldots, \sqrt{p_m})^\top$ &$(\sqrt{q_1}, \ldots, \sqrt{q_m})^\top$
- ρ lớn có nghĩa màu sắc trùng nhau lớn.
- Với mỗi khung ảnh, tìm **y** để tối đa ρ
- Điểm **y** là tọa độ của điểm đích.

Thuật toán theo vết

- Cho $\{q_u\}$ của mô hình và vị trí y của điểm đích trong khung hình trước.

1. Khởi đầu vị trí của điểm đích của frame đang xét là **y**.
2. Tính $\{p_u(\mathbf{y})\}$ và $\rho(p(\mathbf{y}), q)$
3. Áp dụng MeanShift: Tính vi trí đích mới **z** :

$$\mathbf{z} = \frac{\sum_{i=1}^{n_h} g\left(\left\|\frac{\mathbf{y}-\mathbf{y}_i}{h}\right\|^2\right)\mathbf{y}_i}{\sum_{i=1}^{n_h} g\left(\left\|\frac{\mathbf{y}-\mathbf{y}_i}{h}\right\|^2\right)}$$

4. Tính $\{p_u(\mathbf{z})\}$ và $\rho(p(\mathbf{z}), q)$
5. Trong khi $\rho(p(\mathbf{z}), q) < \rho(p(\mathbf{y}), q)$, thực hiện $\mathbf{z} \leftarrow \frac{1}{2}(\mathbf{y}+\mathbf{z})$
6. Nếu ||z-y|| đủ nhỏ thì dừng. Ngược lại: đặt y = z trở lại bước 1.

Thuật toán theo vết

- Bước 3: Thực tế là một cửa sổ của điểm ảnh $_{yi}$ được xét đến. Kích cỡ cửa sổ liên quan đến **h**.
- Bước 5: được sử dụng để xác nhận vị trí điểm đích mới.
  - Có thể dừng nếu **y** và **z** gần như cùng 1 vị trí.
- Thực tế tính toán thì bước 5 chỉ cần gần như 0.1% tổng thời gian tính.
- Bước 6: có thể dừng thuật toán nếu **y** và **z** gần như cùng 1 vị trí.
- Để theo vết đối tượng có thay đổi kích cỡ, thì thay đổi nhiều giá trị bán kính **h**.

Kernel

- Thường thì kernel **K** là một hàm số của $\|\mathbf{x}\|^2$

$$K(\mathbf{x}) = k(\|\mathbf{x}\|^2)$$

- k được gọi là profile của K.

- Tính chất:
  - k không âm.
  - k không tăng: $k(x) \geq k(y) \text{ if } x < y$
  - k liên tục từng phần và

$$\int_0^\infty k(x)dx < \infty$$

Kernel

Nhân phẳng:

$$K(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Lỗi bình phương trung bình được tối tiểu bởi nhân Epanechnikov:

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2C_d}(d+2)(1-\|\mathbf{x}\|^2) & \text{if } \|\mathbf{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Với C_d là thể tích của sphere đơn vị d-D:

$$k_E(x) = \begin{cases} \frac{1}{2C_d}(d+2)(1-x) & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x > 1 \end{cases}$$

Kernel

- Được sử dụng nhiều là nhân Gaussian:

$$K(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d}} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$$

- Với profile:

$$k(x) = \frac{1}{\sqrt{(2\pi)^d}} \exp\left(-\frac{1}{2}x\right)$$

- Biểu diễn khác của kernel $G(\mathbf{x}) = g(\|\mathbf{x}\|^2)$ là:

$$g(x) = -\frac{dk(x)}{dx}$$

Tham khảo

[1] Multiple View Geometry in Computer Vision, Richard Hartley, Andrew Zisserman, Cambridge Unv. Press, 2003.

[2] M.S. Arulampalam, et.al., A tutorial on particle filters for online nonlinear/non-GaussianBayesian tracking, Trans. IEEE Signal Processing, Vol. 50(2), 2002, pp. 174-188.

[3] Computer Vision: Algorithms and Applications, Richard Szeliski, Springer, 2011.

[4] L.W. Kheng, Course slide, CS4243, National University of Singapore.