Computer Vision II

Speed is Important

Tracking: estimate the number & state of objects in a region of interest

- number:
  - 1: single target tracking
  - 0 or 1: detection & tracking
  - N: Multi-target detection & tracking
- state:
  - [x, y]: location
  - [x, y, dx, dy]: location, velocity
  - [x, y, appearance params]: ...
- objects: humans...

- Articulated Tracking: estimate the motion of objects with multiple, coordinate parts
  Ex: face expression reconstruction, human-posed estimation
- Active Tracking: moving sensor in response to motion of target
  need to be real time

+) Elements of Tracking:
- Detection: Find object(s) of interest
- Association: which observations come from same object
- Prediction: predict future motion based on the observed motion pattern
  use this prediction to improve detection & association

+) Visual Odometry: - track camera motion
- also involves a data association problem
- is prone to drift due to local views

+) Visual SLAM (Simultaneous Localization & Mapping)

- Global & local optimization methods

+) Lecture Overview:

⊗ Exercise:

1) Simple Background Models
   Statistical Background Models
   Generalized Lukas-Kanade Tracking

2) Kalman Filter, Extended KF Tracking
   Particle Filter Tracking

Single-Object Tracking

2) Background Modelling

- General algorithm for all kind of objects
- Real time ⇒ need to be simple calculation
- Assume: static camera, moving objects are important

1) Simple Background Models:

+) Simple Background Subtraction: $|I(t) - B|$ ⟹ abs → threshold λ → $M(t)$

- Problems:
  - objects enter scene, then stop ⇒ still detected
  - assumed-static object starts moving → 2 objects
  - illumination
  - global threshold is often suboptimal

+) Single Frame Differencing: $|I(t) - B(t-1)|$ → abs → $T(\lambda)$ → $M(t)$

(+) quick to adapt to change of lightning, camera motion

(−) few pixels of object are labeled when whole object body has similar color..
object moving away/toward camera

⇒ +) Temporal Scale: $D(N) = \|I(t) - I(t+N)\|$

⇒ +) Three-Frame Differencing: $D(+N)$, $D(-N)$ → AND → object

+) Adaptive Background Subtraction:

$I(t)$ → ⊖ → abs → $\hat{T}$ → $M(t)$

$B(t-1)$ ← delay ← $B(t)$

$B(t) = \alpha \cdot I(t) + (1-\alpha) B(t-1)$

- trails of pixels
- ghosting effects
- center of large object fades

Comments

Simple background models are very heuristic, hard to find good params
optimal temporal scale depends on object (size & speed)
global threshold is often suboptimal for parts of the image

⇒ Need better founded approach

⇒ 2, Statistical Background Models:

+) Single Gaussian: $\hat{\mu} = \frac{1}{N}\sum^{N} x_n \quad ; \quad \hat{\sigma}^2 = \frac{1}{N-1}\sum^{N}(x_n - \hat{\mu})^2$

- Online Adaptation: μ & σ for each pixel

  - Running Estimate: $\hat{\mu}^{(t+1)} = \hat{\mu}^{(t)} + \frac{1}{N}x^{(t+1)} - \frac{1}{N}x^{(t+1-T)}$ (add new one, remove oldest one)

$$(\hat{\sigma}^2)^{(t+1)} = (\hat{\sigma}^2)^{t} + \frac{1}{N-1}\left(x^{(t+1)} - \hat{\mu}^{(t+1)}\right)^2 - \frac{1}{N-1}\left(x^{(t+1-T)} - \hat{\mu}^{(t+1-T)}\right)^2$$

  - Exponential Moving Average Filter: (since new data are usually more valuable)

$$\hat{\mu}^{(t+1)} = (1-\alpha)\,\hat{\mu}^{(t)} + \alpha \cdot x^{(t+1)}$$

$$(\hat{\sigma}^2)^{t+1} = (1-\alpha)(\hat{\sigma}^2)^{(t)} + \alpha\left(x^{(t+1)} - \hat{\mu}^{(t+1)}\right)^2$$

α as fixed learning rate

- Problems: A single Gaussian is clearly insufficient
  Adaptation speed vs Sensitivity

+) MoG (Mixture of Gaussians):

- EM Algorithm: $p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k)$

+) Stauffer - Grimson: K Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \text{ where } \Sigma_k = \sigma_k^2 \cdot I$$

- Check every new pixel value, within $2,5\,\sigma_k$ of μ_k ⇒ matched
- If matched ⇒ adapt the corresponding component
  else, replace least probable component with new value as μ_k
  and an initial high σ_k and small π_k

- order components by π_k/σ_k, select B best components as background

$$B = \underset{b}{\operatorname{argmin}} \left(\sum_{k=1}^{b} \frac{\pi_k}{\sigma_k} > c \right)$$

- Online adapting for matching components:

$M_{k,t} = 1$ iff component k matched, else 0

$$\pi_k^{(t+1)} = (1-\alpha)\,\pi_k^{(t)} + \alpha \cdot M_{k,t}$$

$$\mu_k^{(t+1)} = (1-\rho)\,\mu_k^{(t)} + \rho \cdot x^{(t+1)}$$

$$\Sigma_k^{(t+1)} = (1-\rho)\,\Sigma_k^{(t)} + \rho \left(x^{(t+1)} - \mu^{(t+1)}\right)\left(x^{(t+1)} - \mu^{(t+1)}\right)^T$$

where $\rho = \alpha \cdot \mathcal{N}(x_n \mid \mu_k, \Sigma_k)$ update weighted by component likelihood

Comments MoG:
- static foreground objects can be integrated
- ordering by π_k/σ_k: components have more evidence $\pi_k \uparrow$ smaller variance $\sigma_k \downarrow$
- can select only one component if unimodel
- not flexible with dynamic areas (wind + trees rippling water ..)

+) Kernel background Modelling:

$$p(x^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{\left[-\frac{1}{2} \frac{(x_j^{(t)} - x_j^{(i)})^2}{\sigma_j^2}\right]}$$

current pixel value $x^{(t)}$
with N other value at other time $x^{(i)}$

$$= \frac{1}{N} \sum_{i=1}^{N} K\left(x^{(t)} - x^{(i)}\right)$$ with K as Gaussian $\mathcal{N}(0, \Sigma)$, $\Sigma = \operatorname{diag}(\sigma_j)$

⇒ foreground if $p(x^{(t)}) < c$

[can be speed up with look-up table..
[can be very flexible, deal with large amounts of background motion

+) Update:
- FIFO: discard oldest
- Selective update: uncontaminated or go to deadlock situation
- Blind update: no deadlock, but more false negatives

⇒ Solution: combine Short-term model (Selective update) with Long-term model (Blind update)
most recent window; larger time window

Summary: Background Modelling: fast & simple, if applicable, use it
Stauffer-Grimson model } perform well
Kernel background model } used extensively

7

⊛ Template based tracking:

L3/

1, Lucas - Kanade Optical Flow:

Try to measure motion of each pixel in the scene

+) Key assumptions:
- Brightness constancy
- Small motion
- Spatial coherence (points move like their neighbors)

- Brightness constancy:

$$f(x,y,t) = f(x+dx, y+dy, t+dt)$$

⇒ Taylor expansion for approximation

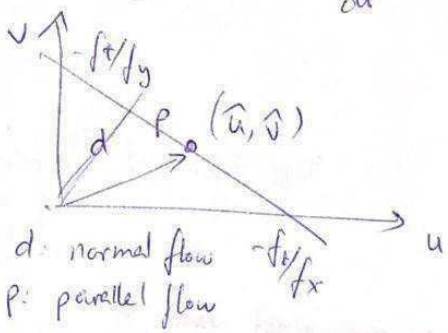
$$f(x,y,t) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt$$

$$\Leftrightarrow \quad f_x \cdot dx + f_y \cdot dy + f_t \cdot dt = 0$$

$$\Leftrightarrow \quad \boxed{f_x \cdot u + f_y \cdot v + f_t = 0}$$

$u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$

$$\Rightarrow \quad v = -\frac{f_x}{f_y}u - \frac{f_t}{f_y} \quad \text{equation of a line}$$



d: normal flow
p: parallel flow

⇒ The Aperture problem
(The Barber Pole Illusion)

can't detect velocity that perpendicular to the gradient

+) Horn & Schunk:

$$\text{minimize} \iint \{ (f_x u + f_y v + f_t)^2 + \lambda (u_x^2 + u_y^2 + v_x^2 + v_y^2) \} dx\,dy$$

brightness constancy — smoothness constraint

$$\Leftrightarrow \begin{cases} (f_x u + f_y v + f_t) f_x + \lambda(\nabla^2 u) = 0 \\ (f_x u + f_y v + f_t) f_y + \lambda(\nabla^2 v) = 0 \end{cases}$$

$$\text{discretize} \cdot \begin{cases} (f_x u + f_y v + f_t) f_x + \lambda(u - u_{av}) = 0 \\ (f_x u + f_y v + f_t) f_y + \lambda(v - v_{av}) = 0 \end{cases} \Leftrightarrow \begin{cases} u = u_{av} - f_x \frac{P}{D} \\ v = v_{av} - f_y \frac{P}{D} \end{cases}$$

$$P = f_x u_{av} + f_y v_{av} + f_t$$

$$D = \lambda + f_x^2 + f_y^2$$

Iterative algorithm till u, v converge

+) Lukas - Kanade (Least - squares)

- Pretend neighber pixels have the same (u, v)

→ Use 5×5 window ⇒ 25 equations of brightness constancy.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

$$A \quad \cdot d = b$$

Since A is not square, can not invert normally

$$\Rightarrow A^T A . d = A^T b \quad \Leftrightarrow \begin{bmatrix} \Sigma I_x I_x & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{bmatrix} \Sigma I_x I_t \\ \Sigma I_y I_t \end{bmatrix}$$

$$\Rightarrow d = \underbrace{(A^T A)^{-1} A^T}_{\text{pseudo-inverse}} b$$

- Remember that we assume all pixels in window have same (u, v)

actually, they don't, thus $f_{xi} u + f_{yi} v + f_t \neq 0$

⇒ Minimize $\min \sum_i (f_{xi} u + f_{yi} v + f_t)^2$

$$\Leftrightarrow \begin{cases} \Sigma (f_{xi} u + f_{yi} v + f_t) f_{xi} = 0 \\ \Sigma (f_{xi} u + f_{yi} v + f_t) f_{yi} = 0 \end{cases} \Leftrightarrow \begin{cases} \Sigma f_{xi}^2 u + \Sigma f_{xi} f_{yi} v = -\Sigma f_{xi} f_{ti} \\ \Sigma f_{xi} f_{yi} u + \Sigma f_{yi}^2 v = -\Sigma f_{yi} f_{ti} \end{cases}$$

+) Problem: 1 Errors from non-linear

⇒ Iterative LK Refinement

- Estimate velocity at each pixel by solving LK equations
- Warp $I_t \Rightarrow I_{t+1}$ used (u, v), but we only get I'_{t+1}

$I'_{t+1} \neq I_{t+1}$

- Repeat (estimate (u, v) from I'_{t+1} and I_{t+1} ⇒ warp ⇒ . .)

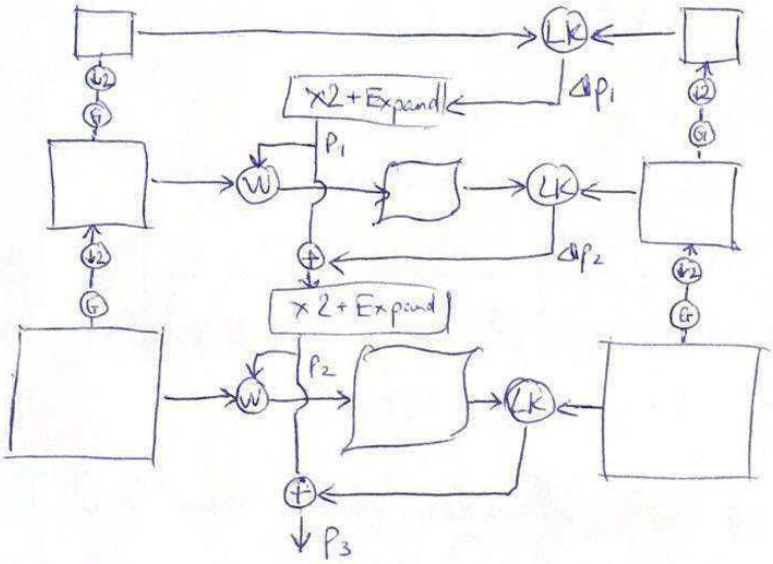
until convergence

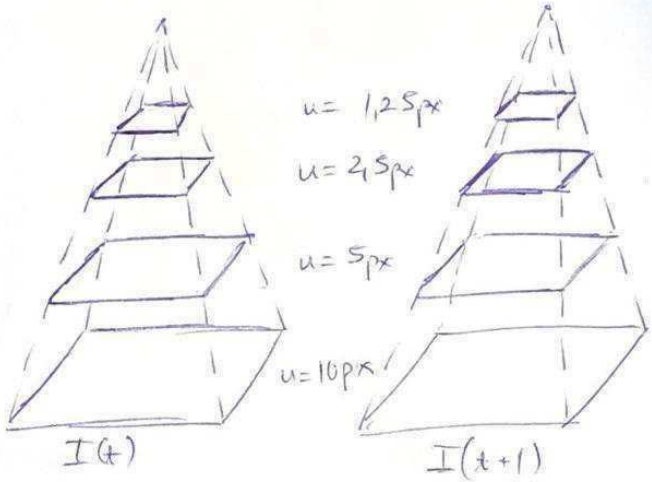
Forward warping
+
Backward warping
(with bilinear interpolation)

+) Problem 2 : Large Motions (more than 1 - 2 pixels)

⇒ Coarse-to-Fine Optical Flow

⇒ Hierarchical LK:





1) Image Pyramid with Gaussian blur and downsampling
2) LK on level k
3) x2 and Upsampling
4) Warp image at level k-1
5) Refinement ⇒ repeat

2) Feature Tracking:

Sparse LK · Shi-Tomasi Feature Tracker (KLT)

Only find optical flow of key feature points

For points in middle of object, we can't estimate (u,v) accurately

In terms of math, A has small eigen values

- Comparing to the first frame helps minimize the drift

KLT : first step of SfM / SLAM

LK Template Tracking

Lg)

+) Idea: In KLT, we track corner-like feature (small patches: 5×5px) → compute optical flow

Now, LK Template Tracking, use larger window

+) $$E(u,v) = \sum_x \left[I(x+u, y+v) - T(x,y) \right]^2$$

- Problematic Assumption: a constant flow (pure translation)

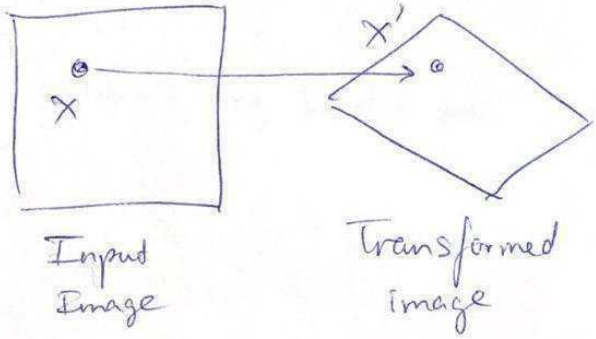
(-) In practice, objects might rotate...

⇒ Use "warping" function W with parameters p

$$E(p) = \sum_x \left[I(W([x,y];p)) - T([x,y]) \right]^2$$

+) Geometric Image Warping:

$$x' = W(x;p)$$



- Translation:

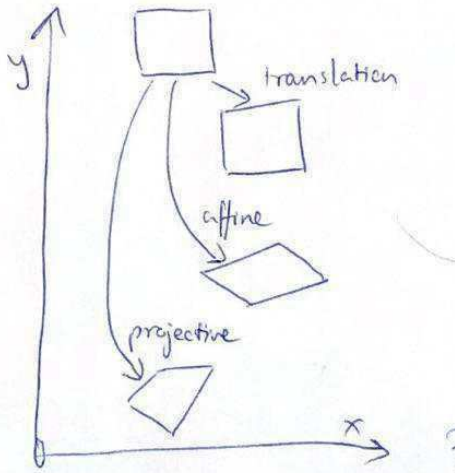
2 unknowns $$W([x,y];p) = \begin{bmatrix} x+p_1 \\ y+p_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Affine:

6 unknowns $$W([x,y];p) = \begin{bmatrix} x+p_1x+p_3y+p_5 \\ y+p_2x+p_4y+p_6 \end{bmatrix} = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Perspective

8 unknowns $$W([x,y];p) = \frac{1}{p_7x+p_8y+1} \begin{bmatrix} x+p_1x+p_3y+p_5 \\ y+p_2x+p_4y+p_6 \end{bmatrix}$$



- With KLT, we kind of already consider warping, but only translation warping, thus $p = (u, v)$

⇒ Now, we consider affine warping. p has 6 dof

$$I(W(x; p+\Delta p)) \approx I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p$$

- As $\frac{\partial W}{\partial p} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{bmatrix}$

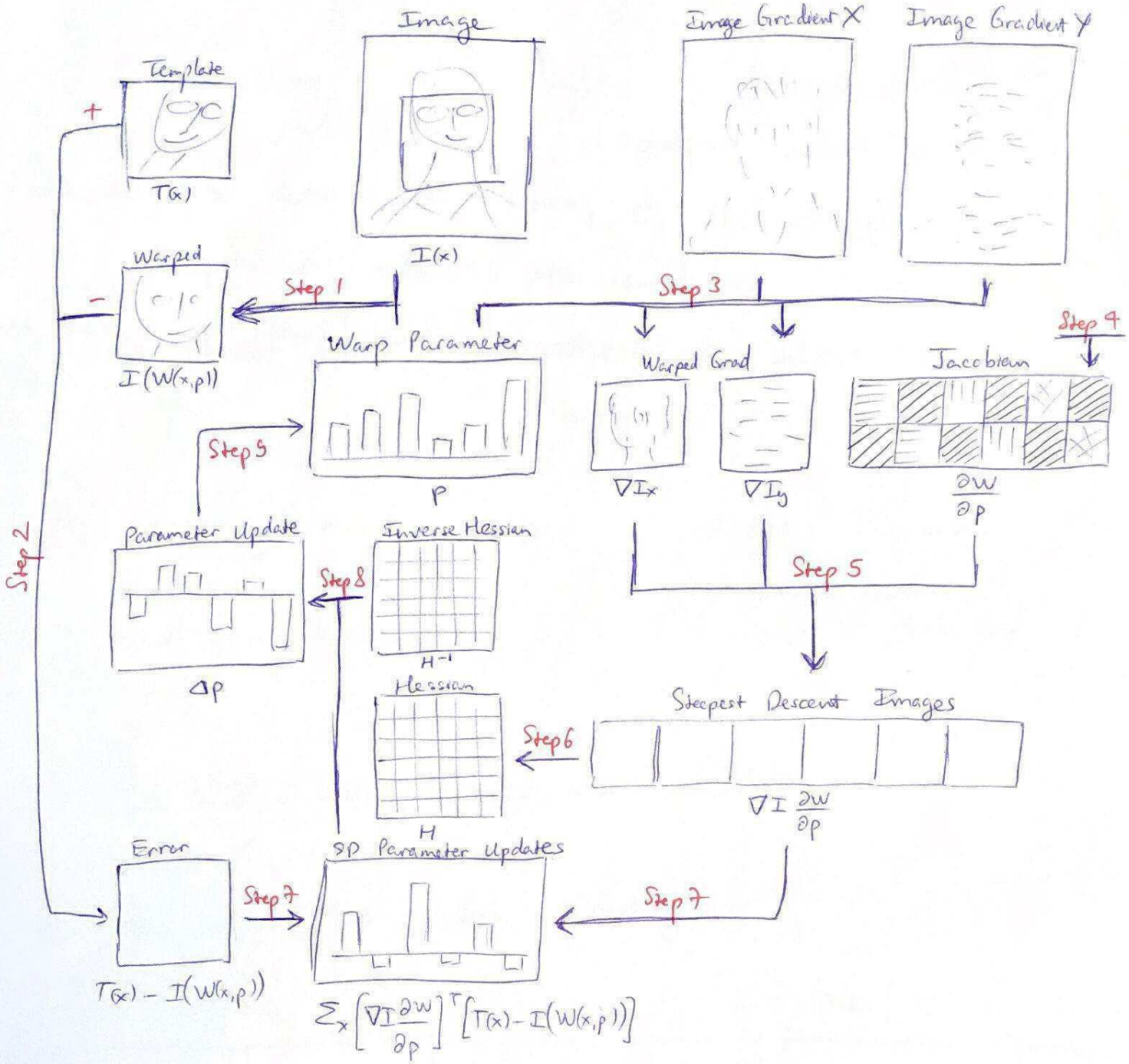
And that: for affine tf $W([x,y];p) = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + p_1 x + p_3 y + p_5 \\ y + p_2 x + p_4 y + p_6 \end{bmatrix}$

$$\Rightarrow \frac{\partial W}{\partial p} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

⇒ Problem: $\arg\min_{\Delta p} \sum_x \left[I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right]^2$

⇒ Closed form solution: $\boxed{\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p} \right]^T \left[I(x) - I(W(x;p)) \right]}$

Hessian $H = \sum_x \left[\nabla I \frac{\partial W}{\partial p} \right]^T \left[\nabla I \frac{\partial W}{\partial p} \right]$



⊘, Discussion:

(+) Fast & Simple algorithm

Applicable to Optical flow estimation
stereo disparity estimation
parametric motion tracking

(−) Prone to local minima

Relatively small movement

⇒ Good init neccessary

* Note:

- When tracking templates..

  Typically use the previous frame's result as initialization

  Higher the frame rate, smaller the warp

  ⇒ Tracking becomes easier (& faster) with higher frame rates

Tracking by Online Classification

L5)

Idea:

- As object moves, new objects | scenes will appear background

- Object Detector ⟹ Object Tracker

Fixed training set
General object detector

On-line update
Object vs Background

1) Adaboost in Viola-Jones Detector

(*) From offline ⇒ Online Boosting

Offline	Online
Given: - Set of labeled samples $X = \{ \langle x_1, y_1 \rangle, \dots, \langle x_L, y_L \rangle \mid y_i = \pm 1 \}$	- One labeled sample $\langle x, y \rangle \mid y = \pm 1$
- Weight distribution of samples $D_0 = 1/L$	- Strong classifiers to update
	- Initial importance $\lambda = 1$ of sample
Loop: for $n = 1$ to N	Loop:
- Train a new weak classifier $h_n^{weak}(x) = \mathcal{L}(X, D_{n-1})$	- Update weak classifiers using {samples and their import. $h_n^{weak}(x) = \mathcal{L}(h_n^{weak}, \langle x, y \rangle, \lambda)$
- Calculate error e_n	- Update error estimation $\hat{e}_n$
- Calculate weight for classifier $\alpha_n = f(e_n)$	- Update weight of classifier: $\alpha_n = f(\hat{e}_n)$
- Update weight dist. of samples D_n	- Update importance weights of samples: λ
$\uparrow_{next}$	$\uparrow_{next}$
$h^{strong}(x) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n h_n^{weak}(x) \right)$	$h^{strong}(x) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n h_n^{weak}(x) \right)$

- Compare:

	Offline Boosting	Online Boosting
At each iteration	Pass all samples	Pass 1 samples
	Add a new classifier	Update all classifier

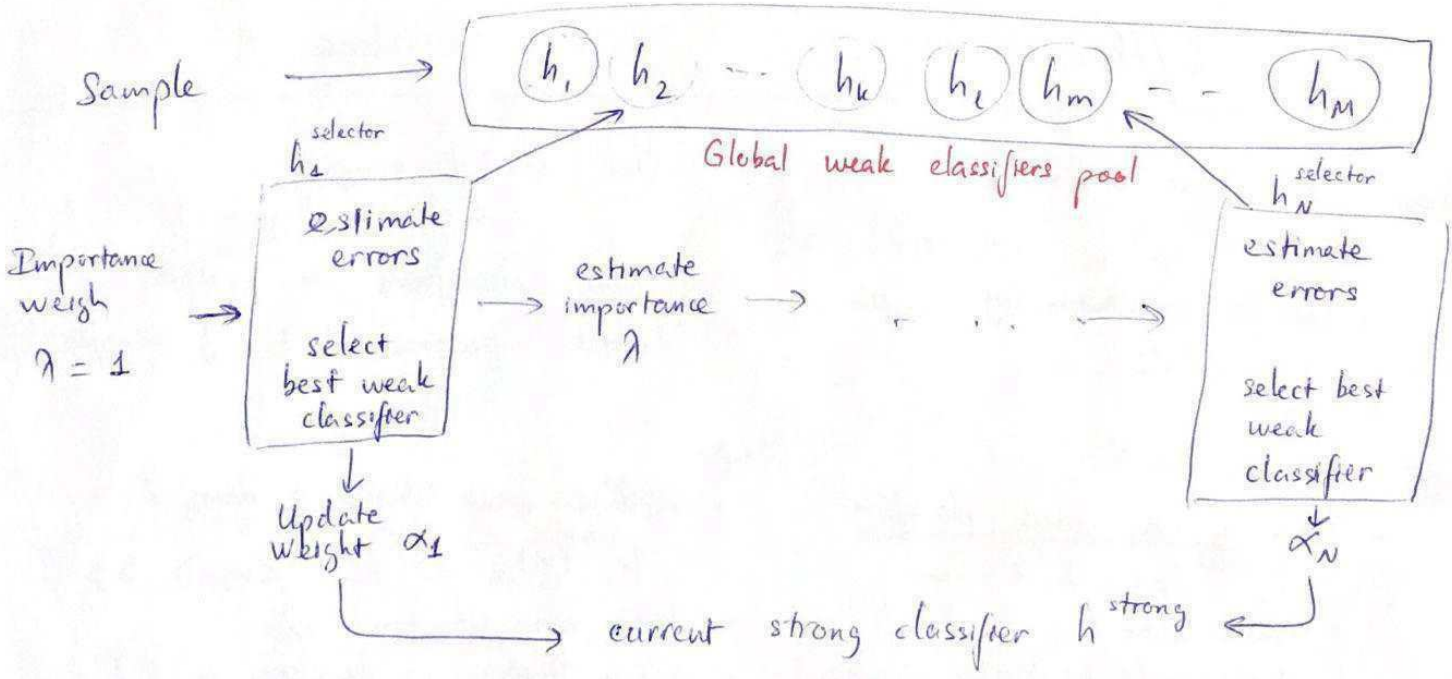
= Given same training set,
Online Boosting will converges to same weak classifiers as Offline Boosting in limit of $N \to \infty$ iterations (Oza, 2001)

+) Features: - Haar-like Wavelets
- HoG
- Locally binary pattern (LBP)

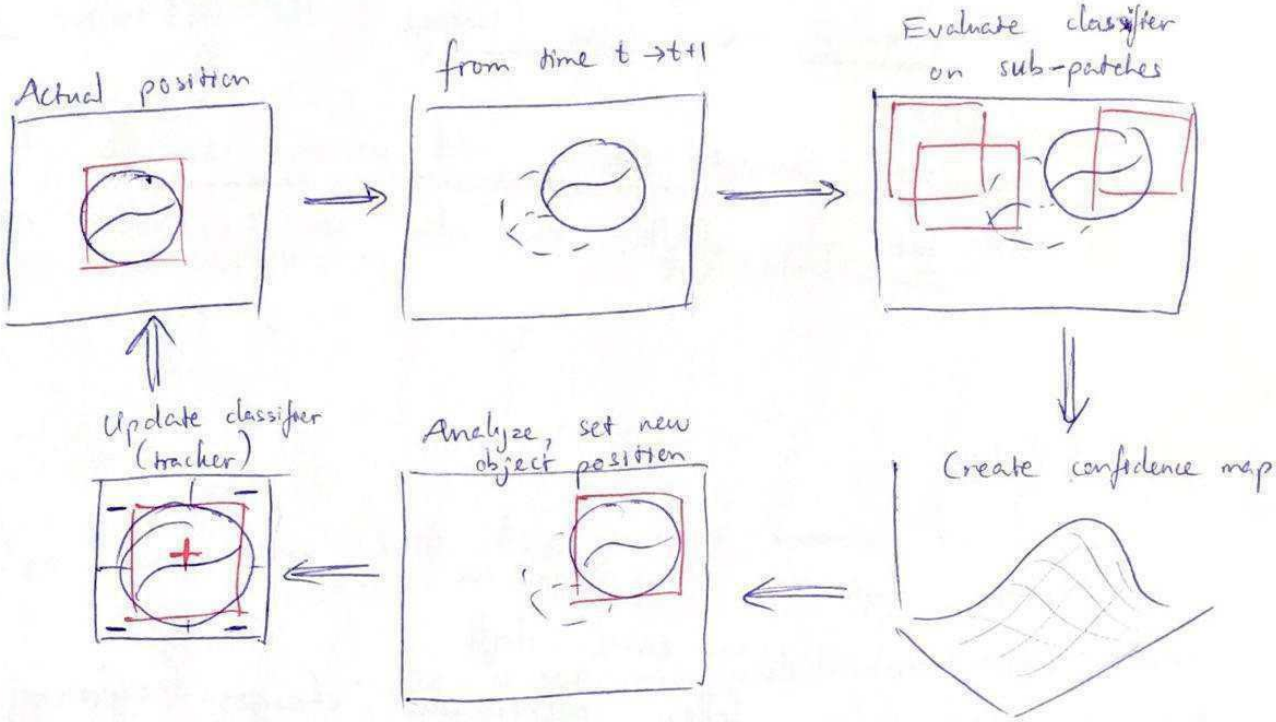
Fast computation using efficient data structures: image integral
integral histograms

+) Selector — choose 1 feature for each weak classifier (because there are many feature choices)

⊗ Direct Feature Selection

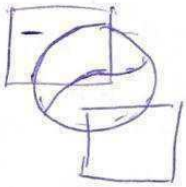


⊗ Tracking by Online Classification



- Add partial overlap samples as negative samples

  ⇒ to penalize if misalign object center



- Online feature exchange:
  Combine difference feature type Haar, LBP ..

⊗ Summary

- Performs Boosting on Selectors, not weak classifiers
  (what features go with what classifiers)
- Use shared feature pool accross classifiers
  (all N classifiers have same choice from M feature)
  not each classifier has different M feature choices

Problem: Drift

- If the change is slow enough, the classifier adapt to new objects
- Because we might slowly add wrong sample
- It will not only drift, but also stay confident about it

⇒ Solution:

1) Match against initialization
   (+) robust to catch drift
   (-) can't follow appearance changes (lightning, rotation ..)

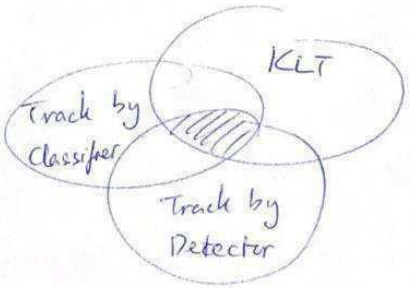
2) Semi supervised learning

   Object Detector ⟷ Object Tracker

   Fixed Prior for updating an adaptive on-line classifier (should still be in same class)

3) Using additional cues:

+) Tracking - Learning - Detection (TLD)
- Combine with informations from [KLT
  [Tracking by Detection ..



Tracking by Detection

L6,

currently the basis for [Multi Object tracking
surveilance

+) Idea:

Detection ⟹ Data Association ⟹ Prediction

- Apply generic object detector
⇒ Objects

- Use appearance models of those objects
- Even for/back-ground segmentations for better appearance model

- Predict position of objects in next time step

1) SVM based Detectors:

- Sliding-window appreach
- HoG, SVM, Image Pyramid, Non-Maximum Suppression

+) Deformable Part-based Model (DPM)

- Idea: Some objects are vary in form, pose ...
- Response of i-th part filter, in l-th level:

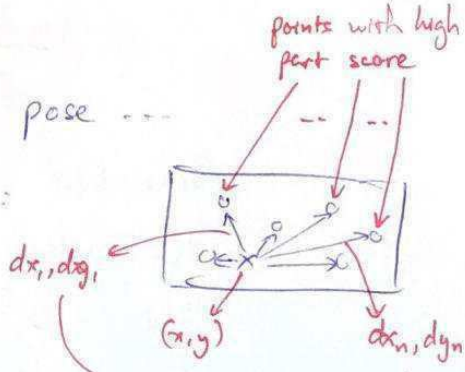
$$R_{i,l}(x,y) = F_i \cdot \phi(H,(x,y,l))$$

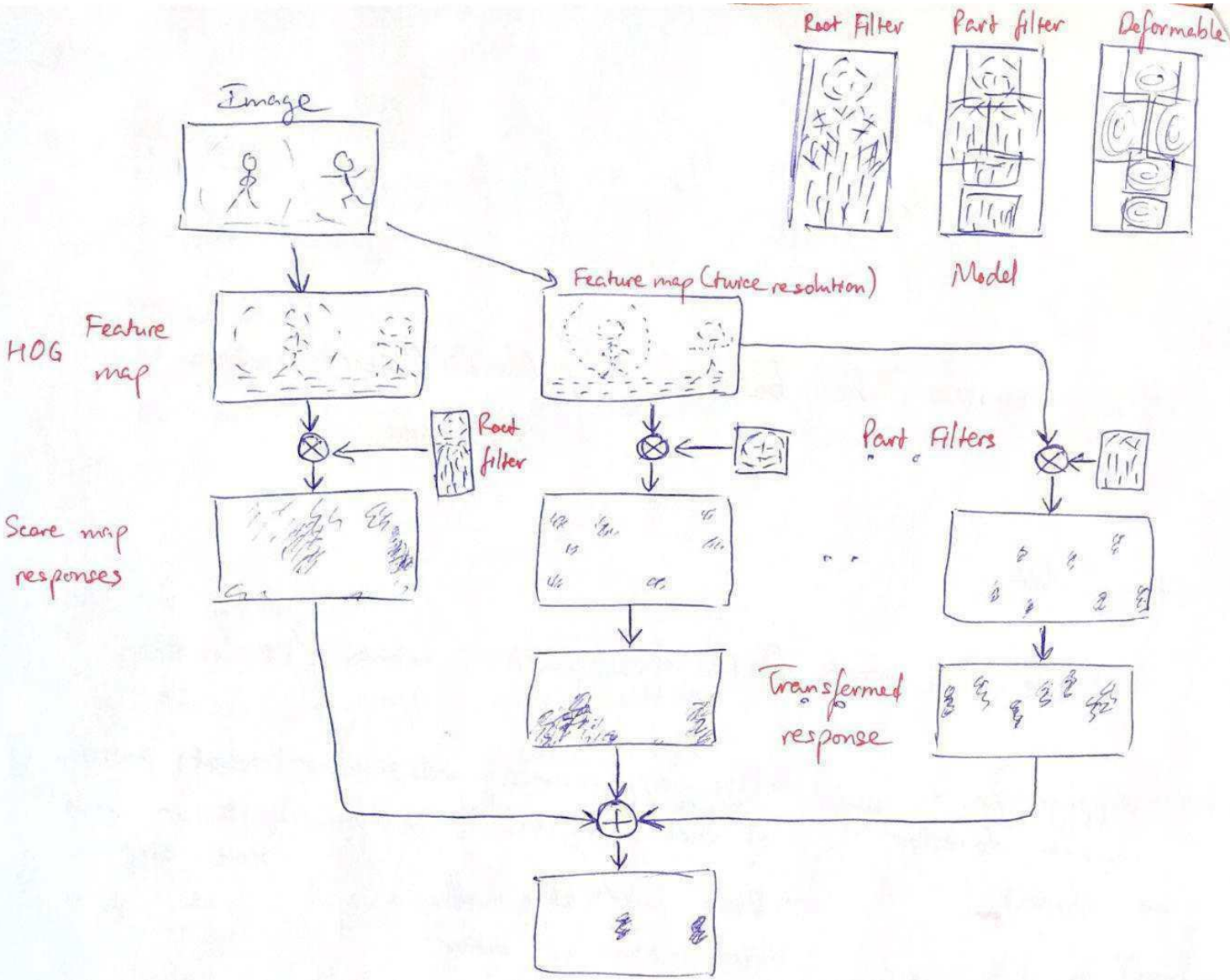
From part score ↔ to root score

⊛ Transformed response

$$D_{i,l}(x,y) = \max_{dx,dy} \left[R_{i,l}(x+dx, y+dy) - d_i \cdot \phi_d(dx,dy)\right] = \max \begin{cases} \dots dx_1, dy_1 \\ \dots dx_n, dy_n \end{cases}$$

dx, dy (the displacement between root & part) has to make sense for the deformable loss to be small





2) AdaBoost-based Detectors

- Viola-Jones: Integral Images + Cascading Classifiers

⇒ Integral Channel Features

- Idea: Generalization of Haar Wavelet
  Combine other feature channels (HoG, color, texture)
  6 orientation bins + Grad magnitude + 3 LUV color channels
- Generalize also block computation
  1st order features: sum of pixels in rectangular region
  2nd ____: Haar like difference over blocks
  Generalized Haar: sum with weights

1) Very Fast / Roerei

- { 1 model, 50 image scales } ⇒ { 50 models, 1 image scale } ⇒ { 5 models, 1 image scale }

Then by using feature interpolation

⇒ Transfer test time computation to train time

- Ensemble of short trees, learned by Adaboost

3) CNN-based Detectors

- Faster RCNN
- Mask RCNN
- YOLO, SSD

Bayesian Filtering

1) Tracking with Dynamics

+) Let state be X, measurement be Y

⇒ Prediction: past measurements ⇒ current state

$$P(X_t \mid Y_0 = y_0, \dots, Y_{t-1} = y_{t-1})$$

Correction: Updated state from prediction & new measurement

$$P(X_t \mid Y_0 = y_0 \dots Y_{t-1} = y_{t-1}, Y_t = y_t)$$

+) Simplify assumptions:

- Only immediate past matters:

$$P(X_t \mid X_0 \dots X_{t-1}) = P(X_t \mid X_{t-1})$$

- Measurements depend only on current state

$$P(Y_t \mid X_0 \dots X_t) = P(Y_t \mid X_t)$$

$$P(A) = \int P(A,B)\,dB$$

$$\int P(X_t, X_{t-1} \mid y_0 \dots y_{t-1})\,dX_{t-1}$$

+) Formulation:

Prediction $P(X_t \mid y_0, \dots y_{t-1}) = \int P(X_t \mid X_{t-1})\, P(X_{t-1} \mid y_0, \dots y_{t-1})\, dX_{t-1}$ // (Dynamics model; corrected estimate)

Correction $P(X_t \mid y_0 \dots y_t) = \dfrac{P(y_t \mid X_t, y_0 \dots y_{t-1})\, P(X_t \mid y_0 \dots y_{t-1})}{P(y_t \mid y_0 \dots y_{t-1})}$ ⟸ $P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$

$$= \frac{P(y_t \mid X_t)\cdot P(X_t \mid y_0 \dots y_{t-1})}{\int P(y_t \mid X_t)\, P(X_t \mid y_0 \dots y_{t-1})\, dX_t}$$

(measurement model; predicted state)

$$= \eta \cdot P(y_t \mid X_t) \cdot P(X_t \mid y_{0:} y_{t-1})$$

2) Linear Dynamics Models.

- Dynamics Model $x_t \sim \mathcal{N}(D_t x_{t-1}, \Sigma_{d_t})$ — x_t: $n\times 1$, D_t: $n\times n$, x_{t-1}: $n\times 1$
- Observation Model $y_t \sim \mathcal{N}(M_t x_t, \Sigma_{m_t})$ — y_t: $m\times 1$, M_t: $m\times n$, x_t: $n\times 1$
- Ex: Constant Velocity (1D points)

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} = D_t x_{t-1} + \Sigma_{d_t} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \cdot x_{t-1} \quad ; \quad y_t = M_t x_t = [1\ 0] \cdot \begin{bmatrix} p_t \\ v_t \end{bmatrix} = p_t$$

Ex Constant Acceleration (1D points)

$$x_t = \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} = D_t x_{t-1} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ a_{t-1} \end{bmatrix} \quad ; \quad y_t = M_t x_t = [1\ 0\ 0] \cdot \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} = p_t$$

3) Kalman Filter

only linear $\begin{cases} \text{transition model} \\ \text{observation} \end{cases}$

$x_t \sim \mathcal{N}(D_t x_{t-1}, \Sigma_{d_t})$

$y_t \sim \mathcal{N}(M_t x_t, \Sigma_{m_t})$

Prediction

$$x_t^- = D_t \cdot x_{t-1}^+$$

$$\Sigma_t^- = D_t \cdot \Sigma_{t-1}^+ \cdot D_t^T + \Sigma_{d_t}$$

Correction

Kalman gain

$$K_t = \Sigma_t^- M_t^T \left(M_t \Sigma_t^- M_t^T + \Sigma_{m_t}\right)^{-1}$$

$$x_t^+ = x_t^- + K_t \underbrace{(y_t - M_t x_t^-)}_{\text{residual}}$$

$$\Sigma_t^+ = (I - K_t M_t)\,\Sigma_t^-$$

Cons: restricted to linear model class of motions

$\Rightarrow$ EKF

4, EKF (Extended Kalman filter)

Ls, + Nonlinear model:

$$x_t = g(x_{t-1}) + \varepsilon_t$$
$$y_t = h(x_t) + \delta_t$$

+, Prediction step:

$$\bar{x}_t = g(x_{t-1}^+)$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1}^+ G_t^T + \Sigma_{d_t}$$

+, Correction step:

Kalman gain

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + \Sigma_{m_t})^{-1}$$
$$x_t^+ = \bar{x}_t + K_t (y_t - h(\bar{x}_t))$$
$$\Sigma_t^+ = (I - K_t H_t) \bar{\Sigma}_t$$

The Jacobians

$$G_t = \left. \frac{\partial g(x)}{\partial x} \right|_{x = x_{t-1}^+}$$

$$H_t = \left. \frac{\partial h(x)}{\partial x} \right|_{x = \bar{x}_t}$$

Note:
- EKF essentially linearizes arround current estimation
- Generally not an optimal estimator
- If initial estimate is wrong ⇒ the filter will quickly diverge
- Still, is the de-facto standard in many applications

Lg, Particle Filter

To represent posterior pdf by randomly chosen weighted samples (particles)

+, Monte Carlo Sampling:

$$\mathbb{E}[f] = \int f(z)\, p(z)\, dz$$

Draw L samples $z^{(l)}$ from $p(z)$

→ Can approximate $\mathbb{E}[\hat{f}] = \mathbb{E}[f]$

with $\hat{f} = \frac{1}{L} \sum_{l}^{L} f(z^{(l)})$

⇒ Monte Carlo integration: $F = \int_D f(x)\,dx$

$$= \int_D \frac{f(x)}{p(x)} p(x)\,dx = \mathbb{E}_{x\sim p}\left[\frac{f(x)}{p(x)}\right]$$

Again, choose N i.i.d samples from p: $x_1 \dots x_N$

$$\Rightarrow \quad F_N = \frac{1}{N}\sum_i^N \frac{f(x_i)}{p(i)} \quad \text{as} \quad \lim_{N\to\infty} F_N = F$$

⇒ Idea: for Importance Sampling:

$$\mathbb{E}[f] = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)} q(z)\,dz$$

Approximate f with a given distribution $p(z)$

$$\simeq \frac{1}{L}\sum_l^L \underbrace{\frac{p(z^l)}{q(z^l)}}_{\text{Importance weight}} f(z^l) \quad \text{with } z^l \sim q(z)$$

by finite samples from a new distribution $q(z)$

+) Sequential Importance Sampling Algorithm (SIS)

function $\left[\{x_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{x_{t-1}^i, w_{t-1}^i\}_{i-1}^N, y_t\right]$

$\eta = 0$ (normalize factor)

for $i = 1:N$ (N particles)

sample from proposal ~~distribution~~ pdf

$x_t^i \sim q(x_t | x_{t-1}^i, y_t)$

$$w_t^i = w_{t-1}^i \frac{p(y_t|x_t^i)\,p(x_t^i|x_{t-1}^i)}{q(x_t | x_{t-1}^i, y_t)}$$

$\eta = \eta + w_t^i$

end

for $i = 1:N$

$w_t^i = w_t^i / N$ (normalize the weights)

end

$$w_t^i = w_{t-1}^i \cdot p(y_t|x_t^i)$$

+) choose prior pdf $q(x_t | x_{t-1}^i, y_t)$ as $p(x_t | x_{t-1})$ (transitional prior)

⇒ SIS Algorithm with Transitional Prior

- Sampling step: $x_t^i = g(x_{t-1}^i) + \varepsilon_t^i$

+ The Degeneracy Phenomenon

After few iterations, most particles have negligible weights

⇒ Waste computational effort to update these particles

→ Need to measure degeneracy : $N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^i)^2}$

[If uniform: $N_{eff} = N$
[Severe degeneracy: $N_{eff} = 1$

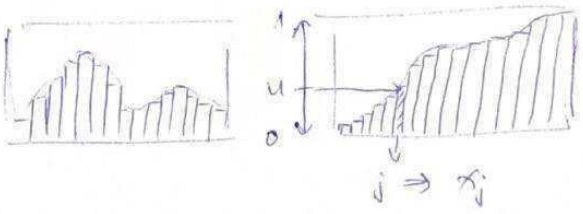
⇒ Resampling according to prior weights

$$\{x_t^i, w_t^i\}_{i=1}^N \rightarrow \left\{x_t^{i*}, \frac{1}{N}\right\}_{i=1}^N$$

+ Inverse Transform Sampling:

(CDF) – Cumulative distribution function:

$$C(k) = \sum_{i=1}^{k} w_i \Big/ \sum_{i=1}^{N} w_i$$



⊗ Arulampalam's Resampling Algorithm

$$\left[\{x_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}\right] = \text{RESAMPLE}\left[\{x_k^i, w_k^i\}_{i=1}^{N_s}\right]$$

Given the weights w of the values x

How to sample the values?

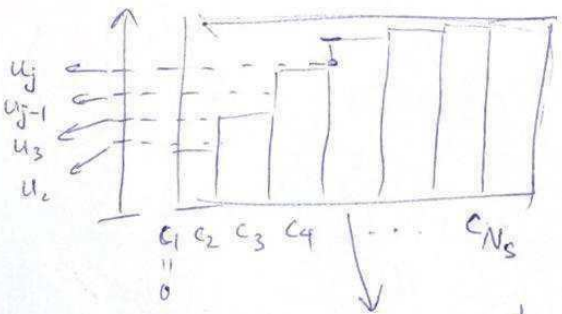
Init CDF : $c_1 = 0$

for $i = 2 : N_s$

| $c_i = c_{i-1} + w_k^i$

end

(Input weights are different to each others)



$i = 1$

$u_1 \sim U[0, N_s^{-1}]$

for $j = 1 : N_s$

| $u_j = u_1 + N_s^{-1}(j-1)$

| while $u_j > c_i$

| | $i++$

| end

| $x_k^{j*} = x_k^i$

| $w_k^j = N_s^{-1}$

| $i^j = i$

end

output weights are all equal

Generic Particle Filter:

$$\text{function}\left[\{x_t^i, w_t^i\}_{i=1}^N\right] = PF\left[\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N, y_t\right]$$

$$\left[\{x_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N, y_t\right]$$

Calculate N_{eff}

if $N_{eff} < N_{thr}$ (threshold)

$\left[\{x_t^i, w_t^i\}_{i=1}^N\right] = RESAMPLE\left[\{x_t^i, w_t^i\}_{i=1}^N\right]$

end

only resample when needed

avoid drift

Another variant: Sampling Importance Resampling (SIR)

$$\text{function}\left[X_t\right] = SIR\left[X_{t-1}, Y_t\right]$$

$\bar{X}_t = X_t = \emptyset$

for $i = 1:N$

  sample $x_t^i \sim p(x_t | x_{t-1}^i)$

  $w_t^i = p(y_t | x_t^i)$

end

for $i = 1:N$

  draw i with probability $\propto w_t^i$

  add x_t^i to X_t

end

Sampling every iteration

prone to drift

but reflect change of particles..

Application:

can track head, body parts

but as single object tracking problem

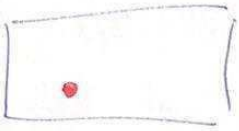
worst case complexity grows exponentially with dimensions

Multi Object Tracking

L16,

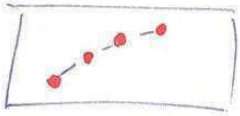
Idea: Elements of Tracking

Detection



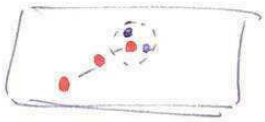
Background modelling
Template based Tracking
Tracking by Online Classification
Pure Detectors (with heatmap)

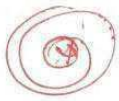
Prediction



Bayesian Filters
KF, EKF
PF, SIS, SIR

Data Association

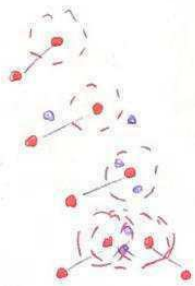




Focus here

⇒ Motion correspondence problem:

+) Ambiguities:
- disappear objects, or obscluded
- unexpected measurements
- one prediction - many measurements
- many predictions - one measurement



1) Simple Approaches

+) Formalization:

- State vector: $\underline{x} = [x, y, v_x, v_y]^T$
  - ⇒ at time t: $\underline{x}^{(k)} = [x^{(k)}, y^{(k)}, v_x^{(k)}, v_y^{(k)}]^T$
- Observation at each time step: $\underline{Y}^{(k)} = \{\underline{y}_1^{(k)}, \dots \underline{y}_{M_k}^{(k)}\}$

⇒ We need to make association between:

tracks: $\{\underline{x}_1^{(k)}, \dots, \underline{x}_{N_k}^{(k)}\}$ and observation $\{\underline{y}_1^{(k)}, \dots, \underline{y}_{M_k}^{(k)}\}$

$z_\ell^{(k)} = j$ iff $\underline{y}_j^{(k)}$ is associated with $\underline{x}_\ell^{(k)}$

+) Gating: only consider measurements within a certain area around predicted location

+) Nearest Neighbor filter: closest to the prediction $\underline{x}_p$

- $z_\ell^{(k)} = \underset{j}{\text{argmin}} \left(\underline{x}_{p,\ell}^{(k)} - \underline{y}_j^{(k)} \right)^T \left(\underline{x}_{p,\ell}^{(k)} - \underline{y}_j^{(k)} \right)$
- Or better: $z_\ell^{(k)} = \underset{j}{\text{argmax}}\ \mathcal{N}\left(\underline{y}_j^{(k)};\ \underline{x}_{p,\ell}^{(k)},\ \underline{\underline{S}}_{p,\ell}^{(k)} \right)$

equivalent to: $z_\ell = \underset{j}{\text{arg min}} \left(\underline{x}_{p,\ell}^{(k)} - \underline{y}_j^{(k)} \right)^T \left(\underline{\underline{S}}_{p,\ell}^{(k)} \right)^{-1} \left(\underline{x}_{p,\ell}^{(k)} - \underline{y}_i^{(k)} \right)$

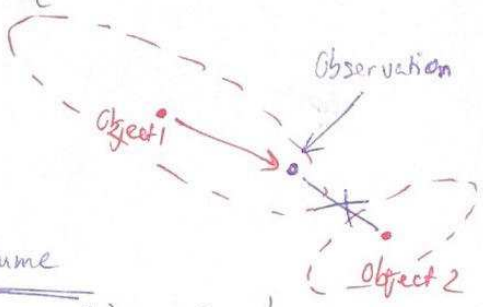
(Mahalanobis distance – makes sense with Kalman filter)

not Euclidean distance

+) Extra notation:

- The innovation that measurement y_j brings to track x_ℓ at time k

$$\underline{\nu}_{j,\ell}^{(k)} = \left(\underline{y}_j^{(k)} - \underline{x}_{p,\ell}^{(k)} \right)$$



$$\Rightarrow \quad p\left(\underline{y}_j^{(k)} \mid \underline{x}_\ell^{(k)} \right) \sim \exp\left\{ -\frac{1}{2}\, \underline{\nu}_{j,\ell}^{(k)T}\, \underline{\underline{S}}_{p,\ell}^{(k)-1}\, \underline{\nu}_{j,\ell}^{(k)} \right\}$$

and the ellipsoidal gating / validation volume

$$V^{(k)}(\gamma) = \left\{ \underline{y} \mid \left(\underline{y} - \underline{x}_{p,\ell}^{(k)} \right)^T \underline{\underline{S}}_{p,\ell}^{(k)-1} \left(\underline{y} - \underline{x}_{p,\ell}^{(k)} \right) \leq \gamma \right\}$$

⊗ Problems: hard assignment – finite chance that the association is incorrect (of nearest neighbor)

⇒ 2) Track-Splitting Filter: as soft/alternate assignments

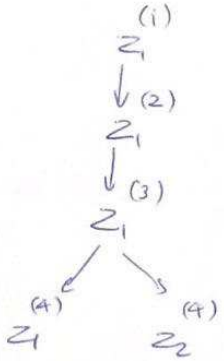
→ postpone decision until additional measurements have been gathered

+) Track likelihood:

Given track ℓ, denotes by $\theta_{k,\ell}$ the event that the sequence of assignments from time step $1 \to k$ belongs to same object

$$Z_{k,\ell} = \left\{ z_{i_1,\ell}^{(1)}, \dots, z_{i_k,\ell}^{(k)} \right\}$$

$$L(\theta_{k,\ell}) = \prod_{j=1}^{k} p\left(z_{i_j,\ell}^{(k)} \mid Z_{(j-1),\ell}, \theta_{k,\ell} \right)$$



- If assume Gaussian obs. likelihood

$$\Rightarrow L(\theta_{k,\ell}) = \prod_{j=1}^{k} \frac{1}{(2\pi)^{d/2} |\Sigma_\ell^{(j)}|^{1/2}} \exp\left[-\frac{1}{2} \sum_{j=1}^{k} v_{ij,\ell}^{(j)\,T} \Sigma_\ell^{(j)-1} v_{ij,\ell}^{(j)}\right]$$

⇒ The modified log-likelihood λ_ℓ of track ℓ

$$\lambda_\ell(k) = -2 \log \left[\frac{L(\theta_{k,\ell})}{\sum_{j=1}^{k} (2\pi)^{-d/2} |\Sigma_\ell^{(j)}|^{-1/2}} \right]$$

$$= \sum_{j=1}^{k} v_{ij,\ell}^{(j)\,T} \Sigma_\ell^{(j)-1} v_{ij,\ell}^{(j)}$$

$$= \lambda_\ell(k-1) + v_{ik,\ell}^{(k)\,T} \Sigma_\ell^{(k)-1} v_{ik,\ell}^{(k)}$$

⇒ Recursive calculation, sum of Mahalanobis distances
(the track with shortest "Mahalanobis" trajectories)

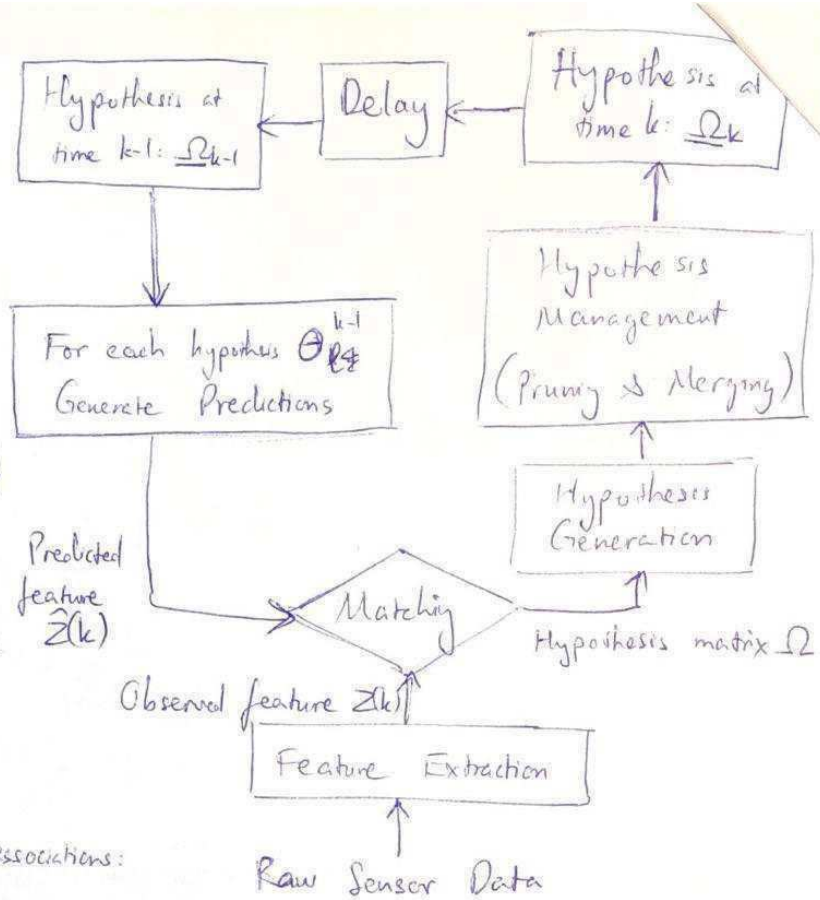
+) Pruning strategies:

- Deleting unlikely tracks : compare $\lambda(k)$ with threshold
- Use sliding window or exponential decay term to deal with long tracks (reduce the effect of very old observation that might overwhelm new observation)
- Merging track nodes: If state estimates are similar → prediction
  both tracks validate identical subsequent measurements → measurement
- Only keeping the most likely N tracks
  rank the tracks based on $\lambda(k)$

3) Multi-Hypothesis Tracking (MHT)

- Target-oriented approaches		- Measurement oriented approaches	
evaluate probabilities of	measurement belongs to established objects	established object / new target	belongs to a measurement sequence / give rise
	$p(Y \mid X)$	$P(X \mid Y)$	

(MHT)



+) Hypothesis generation:

- Set of hypotheses at time k:

$$\underline{\Omega}^{(k)} = \{\Omega_j^{(k)}\}$$

- Latest set of measurements

$$\underline{Y}^{(k)} = \{\underline{y}_1^{(k)}, \underline{y}_2^{(k)}, \dots \underline{y}_{M_k}^{(k)}\}$$

- Feasible associations:
  - A continuation of previous track
  - A false alarm
  - A new target

⇒ Hypotheses matrix of feasible associations:

$$\ominus = \begin{bmatrix} \underline{x}_1 & \underline{x}_2 & \dots & x_{fa} & x_{nt} \\ 1 & 0 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 1 \end{bmatrix} \begin{matrix} y_1 \\ y_2 \\ \vdots \\ y_{M_k} \end{matrix}$$

⇒ $\ominus(i,j) = 1$ if y_i is contained in validation region of $\underline{x}_j$

⇒ Hypothesis $\Omega_j^{(k)}$ =

Assignment z_j:

	$\underline{x}_1$	$\underline{x}_2$	..	x_{fa}	x_{nt}
y_1	0	1		0	0
y_2	1	0		0	0
	0	0		1	0
y_{M_k}	0	0		1	0

each row, only one "1"
each column x_n, only one "1"
except x_{fa} & x_{nt}

$\Omega_{p(j)}^{(k-1)}$: is parent hypothesis of $\Omega_j^{(k)}$

z_j : the set of assignments $\Omega_j^{(k)}$

+) Hypothesis probabilities:

$$p(\Omega_j^{(k)} \mid \underline{Y}^{(k)}) = p(z_j^{(k)}, \Omega_{p(j)}^{(k-1)} \mid \underline{Y}_k^{(k)})$$

$$\overset{\text{Bayes}}{=} \eta \cdot p(\underline{Y}^{(k)} \mid z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) \cdot p(z_j^{(k)}, \Omega_{p(j)}^{(k-1)})$$

$$= \eta \cdot p(\underline{Y}^{(k)} \mid z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) \cdot p(z_j^{(k)} \mid \Omega_{p(j)}^{(k-1)}) \cdot p(\Omega_{p(j)}^{(k-1)})$$

normalization factor | measurement likelihood | prob. of assignment set | prob of parent

+) $M_k = N_{det} + N_{fal} + N_{new}$

+) Measurement likelihood: { assume Gaussian dist of $y_i^{(k)}$ around $\hat{x}_j^{(k)}$ with $\hat{\Sigma}_j^{(k)}$
probabilities of { new track / false alarm } over volume W: W^{-1}

$$\Rightarrow p\left(\underline{Y}^{(k)} \mid \underline{\chi}_j^{(k)}, \Omega_{p(j)}^{(k-1)}\right) = W^{-(N_{fal}+N_{new})} \cdot \prod_{i=1}^{N_{det}} \mathcal{N}\left(y_i^{(k)}; \hat{x}_j, \hat{\Sigma}_j^{(k)}\right)$$

→ Probabilities of an Assignment set.

① Probabilities of number of tracks N_{det}, N_{fal}, N_{new}

N_{det} : Binomial Dist. while N_{fal}, N_{new}: Poisson dist

N is no. of tracks in the parent hypothesis

$$\Rightarrow p\left(N_{det} \mid \Omega_{p(j)}^{(k-1)}\right) = \binom{N}{N_{det}} \cdot p_{det}^{N_{det}} \cdot (1-p_{det})^{(N-N_{det})}$$

$$\Rightarrow p\left(N_{det}, N_{fal}, N_{new} \mid \Omega_{p(j)}^{(k-1)}\right) = \binom{N}{N_{det}} \cdot p_{det}^{N_{det}} \cdot (1-p_{det})^{(N-N_{det})} \cdot \mu(N_{fal}; \lambda_{fal} W), \mu(N_{new} \mid \lambda_{new} W)$$

⊗ $\binom{N}{N_{det}} = {}^{N}C_{N_{det}} = \frac{N!}{N_{det}!(N-N_{det})!}$ No. of combinations, order doesn't matter

$$\mu(x;\lambda) = \frac{\lambda^x \cdot e^{-\lambda}}{x!}$$

② Probability of specific assignments of measurements

$$1 \Big/ \binom{M_k}{N_{det}} \binom{M_k - N_{det}}{N_{fal}} \underbrace{\binom{M_k - N_{det} - N_{fal}}{N_{new}}}_{=1} = \frac{1}{\binom{M_k}{N_{det}} \cdot \binom{M_k - N_{det}}{N_{fal}}}$$

③ Probability of a specific assignment of tracks

$$1 \Big/ \frac{N!}{(N-N_{det})!} \binom{N-N_{det}}{N_{det}} = \frac{(N-N_{det})!}{N!}$$

Example: a lots in military scenarios
detect, track unauthorized vehicles

⊗ - Also works for object with similar appearance model (ants, plane)
observation input (from radar..)

- Fit well with particle filter

$$P\left(Z_j^{(k)} \mid \Omega_{p(j)}^{(k-1)}\right) = \frac{N_{fal}!\, N_{new}!}{M_k!} \cdot P_{det}^{N_{det}} \cdot (1 - P_{det})^{(N - N_{det})} \cdot \mu(N_{fal}; \lambda_{fal} W) \cdot \mu(N_{new}; \lambda_{new} W)$$

+) Practical Issues
- Exponential Complexity
- Need to apply heuristics pruning

+) Pros: when combine with current online-learning appearance model
(initially just popular with radar target tracking)

+) Tricks: Use MWIS (Maximum Weight Independent Set) as efficient way to organize MHT's hypotheses

From here down, consider Multi-Object Tracking purely as Data Association Prob.

4) Data Association as Linear Assignment Problem

- Similarity scores based from [motion prediction (Mahalanobis dist.)
  appearance
  both]

⇒ matrix of pairwise similarity scores

⇒ +) Linear Assignment Problem: (LAP)

- maximize $\sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} z_{ij}$ (weight, assignment)

N tracks, M observation such $\sum_{j=1}^{N} z_{ij} = 1$, $\sum_{i=1}^{M} z_{ij} = 1$

(1 obs per track, 1 track per obs)

- Also could instead minimize cost:

$$\underset{z_{ij}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} z_{ij}$$

+) Greedy algorithm: [find largest
remove scores in same row & column

pretty good, but not optimal solution

+) Hungarian algorithm Use this, not Greedy Algorithm

- The Munkres Assignment algorithm:

① Row reduction: minus each row with the min of that row

② Column reduction: ____ column ____ column

③ Draw lines (minimum) to cover all zeros

If the no. of lines = no. of rows / columns

⇒ optimal solution exists

⇒ Find 0 in way each row & column has only one 0

④ If not, shift zero:

Find Smallest un cover value, subtract from all uncover values

add to intersections of lines

⑤ Make final assignments..

If no. rows ≠ no. columns ⇒ add rows/ columns of 0

But remove solution in these rows / columns

- The overall complexity $O(n^3)$

frame t	frame t+1		
	1	2	3
a	0,11	0,95	0,23
b	0,35	0,25	0,89
c	0,90	0,12	0,81

L12/

② LAP as 2-frame problems

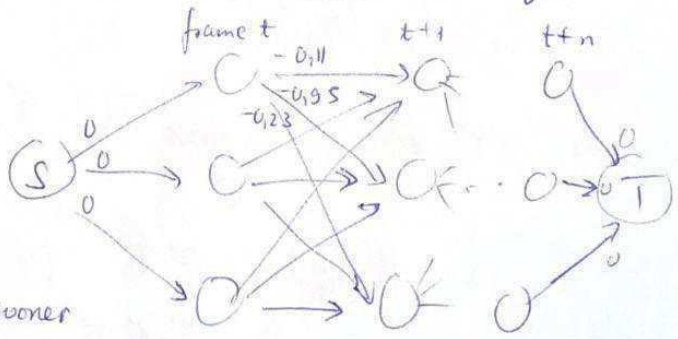
for multi-frame problems

↳ 5) Tracking as Network Flow Optimization

+) Network flow formulation:

- Min-cost flow

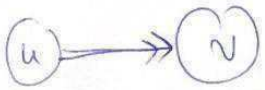
with weights transform to costs



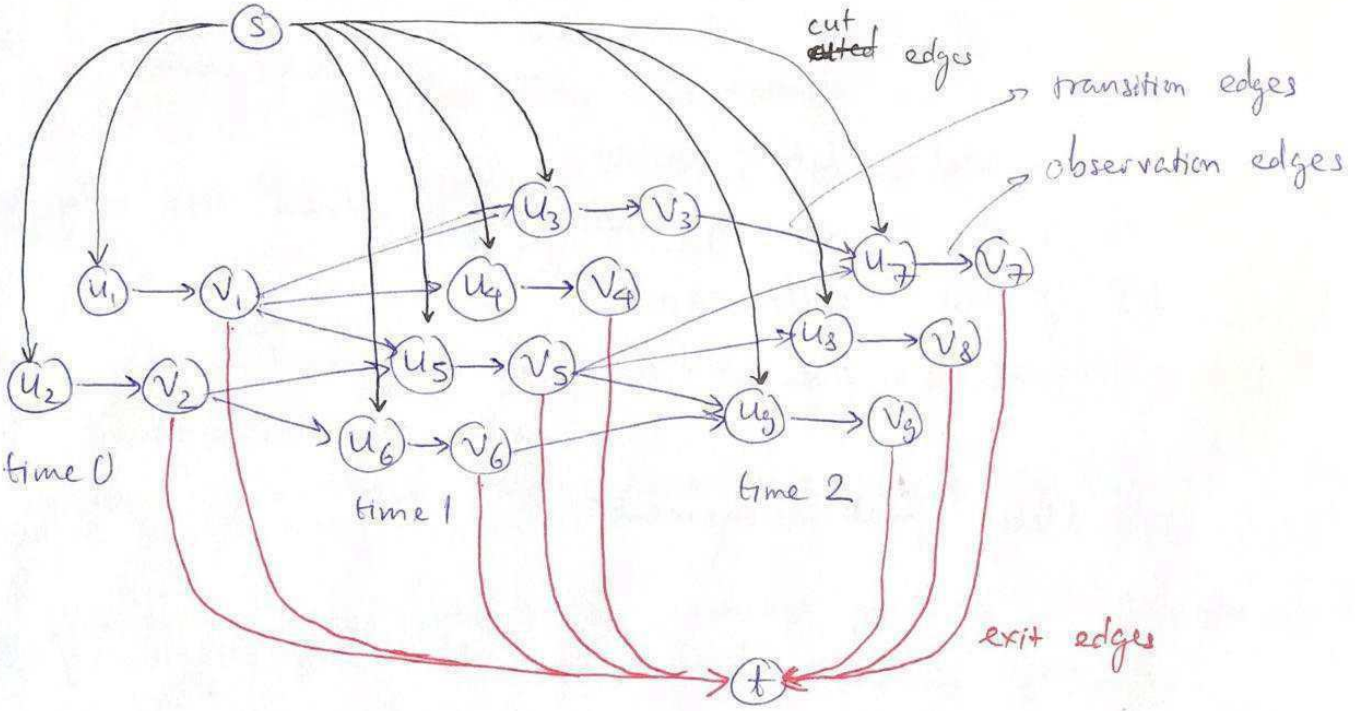
Complication: object appear later, end sooner

⇒ source, sink connect to all intermediate nodes

+). Solution Divide each detection into 2 nodes



detection edge : $c_i = \log \frac{\beta_i}{1-\beta_i}$ ← prob. that detection i is a false alarm



+). Objective function

$$\mathcal{C}^* = \underset{\mathcal{C}}{\operatorname{argmin}} \sum_i c_{in,i} \cdot f_{in,i} + \sum_i c_{i,out} \cdot f_{i,out} + \sum_{i,j} c_{i,j} \cdot f_{i,j} + \sum_i c_i f_i$$

source · sink · association cost · detection edges

$$\text{subject to: } \begin{cases} f_{in,i} + \sum_j f_{j,i} = f_i = f_{out,i} + \sum_j f_{i,j} & \forall i \\ f_i \leq 1 \end{cases}$$

$$c_i = -\log(p_i)$$

$$\Leftrightarrow \quad \mathcal{C}^* = \underset{\mathcal{C}}{\operatorname{argmax}} \prod_i P(o_i \mid \mathcal{C}) \cdot P(\mathcal{C})$$

$$P(\mathcal{C}) = \prod_{T_k \in \mathcal{C}} P(T_k) \quad \text{(Markov)}$$

Comments

- Pros:
  - Clear framework, equivalence to probabilistic formulation
  - Well understood LP optimization prob
  - Globally optimal solution
- Cons
  - not possible to encode exclusion constraints
  - only with zero-velocity model, not constant-velocity model
  - C_{in}, C_{out} are fiddly to set in practice
    - too low ⇒ fragmentations
    - too high ⇒ ID switches

Visual Odometry

43)

- Track motion of camera from its images
- Also involves a data association problem
- Complement other motion sensors
  - GPS
  - Inertia Measurement Units (IMU)
  - Wheel odometry
  - etc
- More accurate than wheel odometry, not prone to slippage
- Important in GPS-denied environments (indoors, close to buildings, Mars)

1) Geometric Background:

+) Geometric Point Primitives

- Point: $x = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$ $\quad [x \; y \; z]^T \in \mathbb{R}^3$
- Augmented vector: $\bar{x} = [x \; y \; 1]^T \in \mathbb{R}^3$ $\quad [x \; y \; z \; 1] \in \mathbb{R}^4$
- Homogeneous coordinates: $\tilde{x} = [\tilde{x} \; \tilde{y} \; \tilde{w}]^T \in \mathbb{P}^2$ $\quad [\tilde{x} \; \tilde{y} \; \tilde{z} \; \tilde{w}] \in \mathbb{P}^3$

$$\tilde{x} = \tilde{w} \cdot \bar{x}$$

+) Euclidean transformations:

$$x' = Rx + t \qquad \bar{x}' = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \cdot \bar{x}$$

+) Rotation:

- Special orthogonal Group $SO(n)$: if R is orthonormal matrix

$$R \in SO(n) \subset \mathbb{R}^{n \times n}$$

$$\det(R) = 1, \quad RR^T = I$$

associative
but not commutative
inverse & neutral element exist

+) 3D Rotation Representations

- $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}_{3\times3} \in \mathbb{R}^{3\times3}$, orthonormal matrix

  ⇒ 9 parameters for 3 DoF

→ - Euler Angles : minimal with 3 parameters

  But with singularities (gimbal lock)

→ - Axis & Angle: rotate along axis $\underline{n} \in \mathbb{R}^3$ by angle $\theta \in \mathbb{R}$

$$R(\underline{n}, \theta) = I + \sin(\theta)\hat{n} + (1-\cos\theta)\hat{n}^2 \qquad \|\underline{n}\|_2 = 1$$

(unit norm constraint)

$$\hat{x} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \qquad \hat{x}y = \underline{x} \times y$$

(n, θ): 4 params

$\omega = \theta n$: 3 params

+) Quarternions

$$q = [q_x \; q_y \; q_z \; q_w]^T \in \mathbb{R}^4, \quad \|q\|_2 = 1$$

- Relation with Axis & Angle

  - $q(n, \theta) = \left[n^T \sin(\theta/2), \; \cos(\theta/2) \right]$ axis angle ⇒ quarternion

  - $\underline{n}(q) = \begin{cases} (q_x \; q_y \; q_z)^T / \sin(\theta/2) & \theta \neq 0 \\ \underline{0} & \theta = 0 \end{cases}$ quarternion ⇒ axis & angle

$$\theta = 2 \arccos(q_w)$$

+) Special Euclidean Group SE(3)

$$T = \begin{bmatrix} R & * \\ 0 & 1 \end{bmatrix} \in SE(3) \subset \mathbb{R}^{4\times4}$$

$SE(3) \times SE(3) \to SE(3)$

$T_B^A \cdot T_C^B \mapsto T_C^A$

+) Sensor types for Visual Odometry:

- Monocular cameras

Input $I_{0:t} = \{I_0, \dots I_t\}$

(+) low power, light weight, low-cost, simple

(−) require motion parallax & texture

scale not observable

- Stereo cameras: (left + right)

$I^l_{0:t} = \{I^l_0 \dots I^l_t\}$

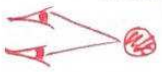
$I^r_{0:t} = \{I^r_0 \dots I^r_t\}$

(+) depth without motion, less power than active sensor

(−) require texture

accuracy depends on baseline, calibration..

have to invest in computation hardware



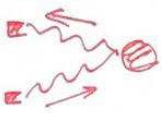
- Active RGB-D sensors

$I_{0:t} = \{I_0 \dots I_t\}$

$Z_{0:t} = \{Z_0 \dots Z_t\}$

(+) No texture needed

(−) consume power, blackbox depth estimation



Mostly indoor, because outdoor, the sun can overwhelm the infrared from our sensor,

⇒ Outputs of Visual odometry:

relative transformation estimates $T_t^{t-1} \in SE(3)$

(pose at t to t-1)

- $T_t \in SE(3)$ is the camera pose at time t in world frame

$$T_t = T_0 T_1^0 \dots T_t^{t-1}$$

+) Direct vs Indirect Methods:

Direct Methods

photometric error

$$p(I_2 \mid I_1, \xi) \Rightarrow E(\xi) = \int_{u \in \Omega} |I_1(u) - I_2(\omega(u,\xi)|\, du$$

of the whole image

like Lukas-Kanade, but with 6 dof in 3D

Indirect Methods

Reprojection error of geometric primitives (points, lines..)

$$p(Y_2 \mid Y_1, \xi) \Rightarrow E(\xi) = \sum_i |y_{1,i} - \omega(y_{2,i}, \xi)|$$

L14,

2, Point-based Visual Odometry as indirect method

1) 2D to 2D Motion Estimation

match points from 2D image to 2D points of previous image

- Given corresponding image point observations

$$Y_t = \{ y_{t,1}, y_{t,2}, \dots y_{t,N} \} \;\&\; Y_{t-1} = \{ y_{t-1,1}, y_{t-1,2}, \dots y_{t-1,N} \}$$

of unknown 3D points $X = \{x_1, x_2, \dots x_N\}$

- Goal: determine relative motion T_t^{t-1} between frames

- Reprojection error: nonlinear, non-convex

$$\underset{T_t^{t-1}}{\operatorname{argmin}} \; E(T_t^{t-1}, X) = \sum_{i=1}^{N} \| \bar{y}_{t,i} - \pi(x_i) \|_2^2 + \| \bar{y}_{t-1,i} - \pi\left(T_t^{t-1} \bar{x}_i\right) \|_2^2$$

⇒ not convex
⇒ thus leads to multiple solution
⇒ thus use good initialization first

- Essential matrix: $E = \hat{t} R = [t]_\times . R$

$$\tilde{y}^T . E . \tilde{y}' = 0$$

⊛ Eight-point Algorithm for Essential Matrix

As a linear approach to estimate T_t^{t-1}

① Rewrite epipolar constraints as a linear system of equations

- For each pair of correspondence $\tilde{y}_i$ & $\tilde{y}_i'$: $\tilde{y}_i^T E \tilde{y}_i' = 0 = a_i E_s$

⇒ All pairs: $A . E_s = 0$ with $E_s = [e_{11}, e_{12}, \dots, e_{33}]^T$

$$A = [a_1^T, a_2^T, \dots a_N^T]$$

② Apply SVD to A: $A = U_A \Sigma_A V_A^T$

$A . E_s = 0 \Rightarrow E_s$ is the last column of V_A (the 9^{th} column)

③ Project approximate $\bar{E}$ into the normalized essential space

$$\bar{E} = U . \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3) . V^T \quad \text{with } U, V \in SO(3)$$

Replace $\sigma_1, \sigma_2, \sigma_3$ with $1, 1, 0$ to enforce rank 2

$$E = U . \operatorname{diag}(1, 1, 0) . V^T$$

④ Decomposition of E to R and T

there will be 2 ~~points~~ solutions
2R, 2T ⇒ 4 sols. choose 1 that the point is in front of both cam

$$R = U . R_Z^T\left(\pm \tfrac{\pi}{2}\right) . V^T \;; \quad \hat{t} = U . R_Z\left(\pm \tfrac{\pi}{2}\right) . \operatorname{diag}(1, 1, 0) . U^T$$

⇒ the depths ≥ 0: $\lambda \bar{y}_i = \lambda' R \bar{y}_i' + \gamma t$

$\Rightarrow \lambda, \lambda' > 0$

$R_Z(\varphi)$: as the rotation matrix around z-axis to angle φ

⊛ Remark: - Eight-Point Algorithm is used as initialization for non-linear least-square reprojection error.

- Normalized essential matrix $\|E\| = 1$
  (stack to column vector, then normalized with L_2)
- There is also 5-points algorithm (non linear)
- Will have degenerated solution if all are on same plane
- Normalized Eight-Point Algorithm: better accuracy

⊛ Triangulation (Repeated)

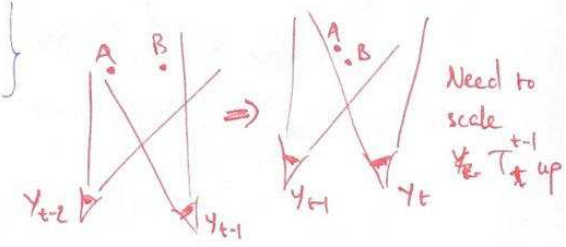
- Goal: reconstruct 3D points: $\tilde{x} = [x \; y \; z \; w]^T \in \mathbb{R}^3$
  from 2D observation $\{y_1 \dots y_N\}$ and known camera pose $\{T_1 \dots T_N\}$
- Reprojection: $y'_i = \pi(T_i \tilde{x}) = \begin{bmatrix} \frac{r_{11}x + r_{12}y + r_{13}z + t_x w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \\ \frac{r_{21}x + r_{22}y + r_{23}z + t_y w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \end{bmatrix}$

+) Linear solution: each image: $y_i - y'_i = 0$

$\Rightarrow A \cdot \tilde{x} = 0$

$\Rightarrow$ [Set $w = 1$ $\Rightarrow$ solve non-homogeneous system
Find null space of A with SVD

+) Non-linear solution: $\min_x \left\{ \sum_{i=1}^{N} \|y_i - y'_i\|_2^2 \right\}$



+) Relative Scale Recovery:

- Triangulation overlapping points Y_{t-2}, Y_{t-1}, Y_t
  $\Rightarrow$ points: $X_{t-2,t-1}$, $X_{t-1,t}$

$\Rightarrow$ Rescale the translation of current relative pose estimate to match reconstruction scale with the distance ratio between corresponding pairs

$$r_{i,j} = \frac{\|X_{t-2,t-1,i} - X_{t-2,t-1,j}\|_2}{\|X_{t-1,t,i} - X_{t-1,t,j}\|_2} \quad \text{(of 1 pair)}$$

$\Rightarrow$ Use mean / robust median over all available pairs

+) 2D to 3D Motion Estimation

- Given a local set of 3D points: $X = \{X_1, X_2, \dots X_N\}$ and corresponding image observations $Y_t = \{y_{t,1}, y_{t,2}, \dots y_{t,N}\}$

  Goal: determine T_t within local map

- Minimize geometric reprojection error

$$E(T_t) = \sum_{i=1}^{N} \| y_{t,i} - \pi(T_t X_i) \|_2^2 \qquad \text{(PnP)}$$

⊛ Direct Linear Transform (DLT) one method of Perspective n Point pose

- Goal: determine reprojection matrix $P = [R\ t] \in \mathbb{R}^{3\times 4} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$

2D: $\tilde{y}_i = [x_i'\ y_i'\ w_i']^T \in \mathbb{P}^2$ → 3D: $\tilde{X}_i = [x_i\ y_i\ z_i\ w_i]^T \in \mathbb{P}^3$

⇒ 2 constraints: $\begin{bmatrix} 0 & -w_i' \tilde{X}_i^T & y_i' \tilde{X}_i^T \\ w_i' \tilde{X}_i^T & 0 & -x_i' \tilde{X}_i^T \end{bmatrix} \cdot \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0$ from $\tilde{y}_i \times (P \tilde{X}_i) = 0$

⇒ Linear system of equations: $Ap = 0$ with $p = \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} \in \mathbb{R}^9$ (?)

from $N \geq 6$ correspondences

+) 3D to 3D Motion Estimation (L15)

- Given 3D point coordinates of corresponding points in 2 frames $X_{t-1} = \{x_{t-1,1}, \dots x_{t-1,N}\}$ & $X_t = \{x_{t,1}, \dots, x_{t,N}\}$

- Goal: determine relative camera pose T_t^{t-1}

- Geometric least square error:

$$E(T_t^{t-1}) = \sum_{i=1}^{N} \| \bar{x}_{t-1,i} - T_t^{t-1} \bar{x}_{t,i} \|_2^2 \qquad N \geq 3$$

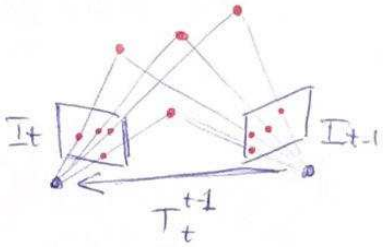
Closed form solution available

- Means of 3D points: $\mu_{t-1} = \frac{1}{N} \sum_{i=1}^{N} x_{t-1,i}$, $\mu_t = \frac{1}{N} \sum_{i=1}^{N} x_{t,i}$

⇒ Set $A = \sum_{i=1}^{N} (x_{t-1} - \mu_{t-1})(x_t - \mu_t)^T$, $A = U \cdot \Sigma \cdot V^T$

⇒ $R_{t-1}^t = V \cdot U^T$ and $t_{t-1}^t = \mu_t - R_{t-1}^t \mu_{t-1}$

Algorithm: 2D to 2D Visual Odometry:

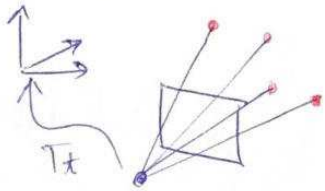


- Input: image sequence $I_{0:t}$
- Output: aggregated camera pose $T_{0:t}$

- For each I_k:
  1) Extract & match key points between I_{k-1} & I_k
  2) Compute T_k^{k-1} from E between I_{k-1}, I_k (8 pts algorithm)
  3) Compute relative scale and rescale translation of T_k^{k-1}
  4) Aggregate $T_k = T_{k-1} T_k^{k-1}$

$$E(T_t^{t-1}, X) = \sum_{i=1}^{N} \|\bar{y}_{t,i} - \pi(\bar{x}_i)\|_2^2 + \|\bar{y}_{t-1,i} - \pi(T_t^{t-1}\bar{x}_t)\|_2^2$$

Algorithm: 2D to 3D Visual Odometry:



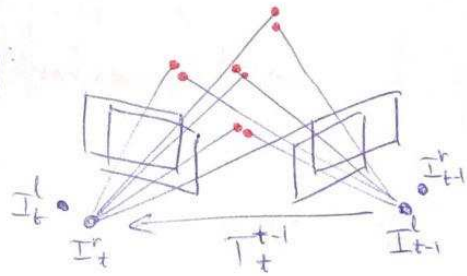
- Input: image sequence $I_{0:t}$
- Output: aggregated camera pose $T_{0:t}$

- Initialize:
  1) Extract & match key points I_0 and I_1
  2) Determine camera pose E and triangulate 3D point X_1
- For each current I_k:
  1) Extract & match key point between I_{k-1} & I_k
  2) Compute T_k using PnP from 2D-to-3D matches using $\tilde{x}_k$ from X_{k-1} (DLT PnP)
  3) Triangulate all new key point matches between I_{k-1} & I_k and add to local map X_k

X_k is not really accurate here. In SLAM, we will reestimate position of points

$$E(T_t) = \sum_{i=1}^{N} \|y_{t,i} - \pi(T_t \bar{x}_i)\|_2^2$$

Algorithm 3D to 3D Visual Odometry:



- Input: stereo image sequence: $I_{0:t}^l$, $I_{0:t}^r$
- Output: aggregated camera pose $T_{0:t}$

- For each current stereo image I_k^l, I_k^r
  1) Extract and match key points between I_k^l and I_{k-1}^l
  2) Triangulate 3D points X_k between I_k^l and I_k^r
  3) Compute camera pose T_k^{k-1} from 3D to 3D point matches X_k to X_{k-1} (Arun's Method)
  4) Aggregate $T_k = T_{k-1} \cdot T_k^{k-1}$

$$E(T_t^{t-1}) = \sum_{i=1}^{N} \|\bar{x}_{t-1,i} - T_t^{t-1}\bar{x}_{t,i}\|_2^2$$

+) Further Consideration

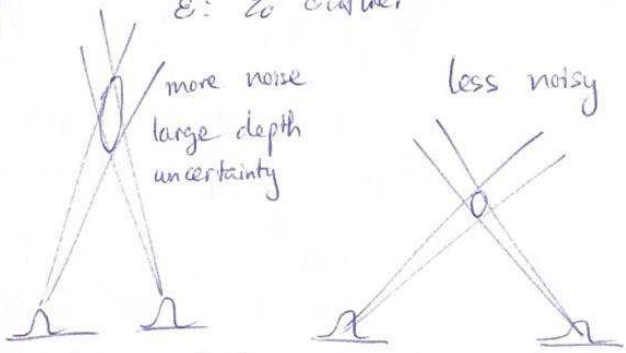
- Keypoint Detectors : Harris, FAST, LoG, DoG, SIFT, SURF
- Data association:
  - match by reprojection error
  - detect-then-track KLT Tracker
  - matching by descriptor
  - robustness through RANSAC
- RANSAC

$$\#\text{iteration } N = \frac{\log(1-p)}{\log\left(1-(1-\varepsilon)^s\right)}$$

p : success prob
s : required data points
ε : % outlier

- Probabilistic Modelling:

$$p(X,\xi|Y) \propto p(Y|X,\xi)\cdot p(X,\xi)$$
$$= p(X,\xi)\prod_{i=1}^{N} p(y_i|x_i,\xi)$$

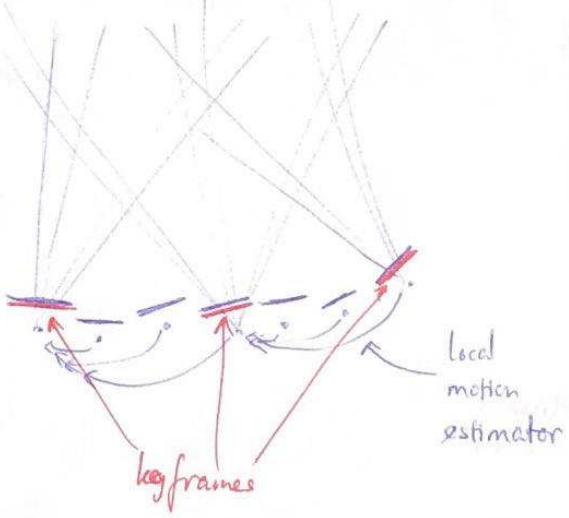


$$E(X,\xi) = \text{const} + (\xi-\mu_{\xi,0})^T \Sigma_{\xi,0}^{-1} (\xi-\mu_{\xi,0}) + \sum_{i=1}^{N} (x_i-\mu_{x_i,0})^T \Sigma_{x_i,0}^{-1} (x_i-\mu_{x_i,0}) + (y_i-\pi(T(\xi)x_i))^T \Sigma_{y_i}^{-1} (y_i-\pi(T(\xi)x_i))$$

Note: 3D-to-3D use stereo camera ⇒ obviously baseline ≪ depth
⇒ actually less accurate than 2D-to-3D VO
⇒ Thus, even though 3D-to-3D is simpler, we prefer 2D-to-3D more

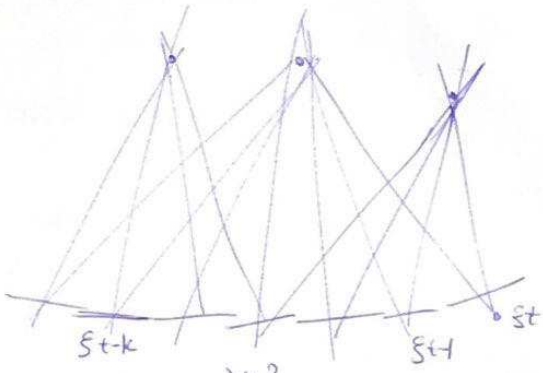
⇒ +) Key frames
- Have a local motion estimate
- But only when the motion is great enough that the baseline is sufficient
⇒ create new key frame

Only aggregate motion considering key frames



+) Local optimization windows

Optimize over more than 2 images



- Bundle adjustment

$$E(X_{t-k:t}, \xi_{t-k:t}) = \sum_{j=0}^{k} \sum_{i=0}^{N_{t-j}} \| y_{t-j,i} - \pi(T(\xi_{t-j}) \cdot x_{t-j,i}) \|_2^2$$

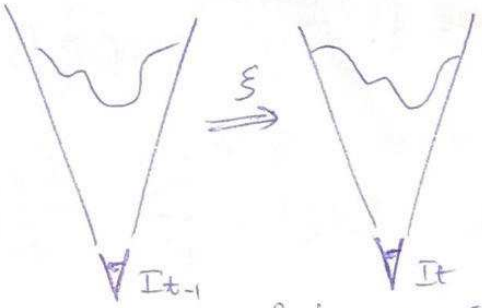
⇒ minimize a bundle of transformation, not just 1 at a time

avoid manually designed keypoints

3) Direct Methods

+) Warping requires depth

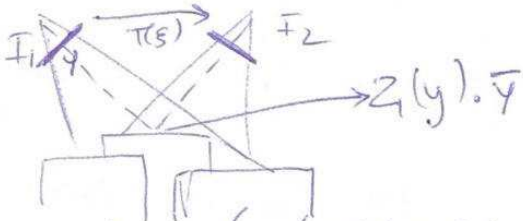
⇒ need camera: RGB-D / Fixed-baseline stereo / Temporal stereo ...



2 images → find warp ξ

$$E(\xi) = \int_{u \in \Omega} |I_1(u) - I_2(\omega(u, \xi))| \, du$$

(over the whole image)



+) $I_1(y) = I_2(\pi(T(\xi) Z_1(y) \bar{y}))$

$= I_2(\pi(T(\xi) x))$

$Z_1(y) \cdot \bar{y} = Z_1(y) \cdot \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = x$

coordinates in image plane / coordinate in 3D space

coord in image plane / coord. in 3D space

$I_2(\underbrace{\pi(T(\xi) x)}_{\text{reprojection of } T(\xi)x \text{ to image plane}}) =$ image at that pose

+) Photometric error: $I_1(y) = I_2\left(\pi\left(T(\xi) \cdot Z_1(y) \cdot \bar{y}\right)\right)$

Geometric error: $\left[T(\xi) Z_1(y) \bar{y}\right]_Z = Z_2\left(\pi\left(T(\xi) Z_1(y) \bar{y}\right)\right)$

depth of points after apply transformation $T(\xi)$ — depth in image 2 of ...

+) With noise: $I_1(y) = I_2\left(\pi\left(T(\xi) Z_1(y) \bar{y}\right)\right) + \varepsilon$

$\varepsilon \sim \mathcal{N}(0, \sigma_I^2)$

- If assume that pixel measurements are stochastically independent

maximize $p(\xi \mid I_1, I_2) \propto p(I_1 \mid \xi, I_2) \cdot p(\xi)$

$\propto p(\xi) \prod_{y \in \Omega} \mathcal{N}\left(I_1(y) - I_2\left(\pi(T(\xi) \cdot Z_1(y) \mid \bar{y})\right); 0, \sigma_I^2\right)$

minimize $E(\xi) = \sum_{y \in \Omega} \frac{r(y,\xi)^2}{\sigma_I^2}$ with $r(y, \xi) = I_1(y) - I_2\left(\pi\left[T(\xi) Z_1(y) \bar{y}\right]\right)$

- Stack into vector: $E(\xi) = r(\xi)^T W r(\xi)$

$\Rightarrow$ This is nonlinear least square problem

+) Gauss-Newton: to solve nonlinear least square

Linearize $\tilde{r}(\xi) = r(\xi_i) + \nabla_\xi r(\xi_i)(\xi - \xi_i)$ $\quad J_i := \nabla_\xi r(\xi_i) \in \mathbb{R}^{\dim(r) \times \dim(\xi)}$

$\xi = \underset{\xi}{\operatorname{argmin}}\ \tilde{E}(\xi) = \frac{1}{2} \tilde{r}(\xi)^T W \tilde{r}(\xi)$

$\nabla_\xi \tilde{E}(\xi) = J_i^T W \tilde{r}(\xi)$

$\nabla_\xi^2 \tilde{E}(\xi) = J_i^T W J_i = H_i \in \mathbb{R}^{\dim(\xi) \times \dim(\xi)}$

Find minimum of linearized system, linearize and set $\nabla_\xi \tilde{E}(\xi) = 0$

$\xi_{i+1} = \xi_i - \left(\nabla_\xi^2 \tilde{E}(\xi_i)\right)^{-1} \nabla_\xi \tilde{E}(\xi_i)$

$= \xi_i - H_i^{-1} J_i^T W r(\xi_i)$

+) Levenberg - Marquardt Method

$$\xi_{i+1} = \xi_i - (H_i + \lambda I)^{-1} J_i^T W r(\xi_i)$$

- If error increase, reject update, increase $\lambda \Rightarrow$ act like gradient descent
- If error decrease, decrease $\lambda \Rightarrow$ act like Gauss-Newton

+) Practical implementation:

H_i and b_i are summed over individual pixels

$$H_i = \sum_{y \in \Omega} \frac{w(y, \xi_i)}{\sigma_I^2} J_{i,y}^T J_{i,y} \quad ; \quad b_i = \sum_{y \in \Omega} J_{i,y}^T \frac{w(y, \xi_i)}{\sigma_I^2} r(y, \xi_i)$$

$$J_{i,y} = \nabla_{\delta\xi} r(y, \delta\xi \oplus \xi_i)$$

allow parallel processing with GPU

+) Lie Algebra:

$$\xi = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6 \qquad \omega \in \mathbb{R}^3, \; v \in \mathbb{R}^3$$

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

- From a twist in Lie Algebra to transformation

$$T = \exp(\hat{\xi}) = \begin{bmatrix} \exp(\hat{\omega}) & Av \\ 0 & 1 \end{bmatrix}$$

with $$\exp(\hat{\omega}) = I + \frac{\sin|\omega|}{|\omega|}\hat{\omega} + \frac{1-\cos|\omega|}{|\omega|^2}\hat{\omega}^2$$

$$A = I + \frac{1-\cos|\omega|}{|\omega|^2}\hat{\omega} + \frac{|\omega| - \sin|\omega|}{|\omega|^3}\hat{\omega}^2$$

- From a transformation to a twist

$$\hat{\xi} = \log(T) = \begin{bmatrix} \log R & A^{-1}t \\ 0 & 0 \end{bmatrix}, \quad T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

$$\log(R) = \frac{|\omega|}{2\sin|\omega|}(R - R^T), \quad |\omega| = \cos^{-1}\left(\frac{\mathrm{tr}(R) - 1}{2}\right)$$

- Inversion of hat operator

$$\begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}^{\vee} = [\omega_1 \ \omega_2 \ \omega_3 \ v_1 \ v_2 \ v_3]^T$$

$$\Rightarrow \quad \xi(T) = (\log(T))^{\vee}, \quad T(\xi) = \exp(\hat{\xi})$$

- Pose inversion:

$$\xi^{-1} = \log(T(\xi)^{-1}) = -\xi$$

- Pose concatenation

$$\xi_1 \oplus \xi_2 = (\log(T(\xi_2)T(\xi_1)))^{\vee}$$

- Pose difference

$$\xi_1 \ominus \xi_2 = (\log(T(\xi_2)^{-1}T(\xi_1)))^{\vee}$$

+) Practical Considerations

- Coarse-to-fine Optimization
- Residual distributions

SLAM

L16)

- Simultaneous Localization and Mapping
- Visual SLAM: SLAM with vision sensors

+) To ensure map consistency ⇒ global / local optimization methods

Global

Bundle adjustment

Pose-graph optimization

Offline (create a map then optimize)

Local

Incremental tracking & mapping

Visual Odometry with local maps

Online (not as good as global one, but faster)

Hybrids

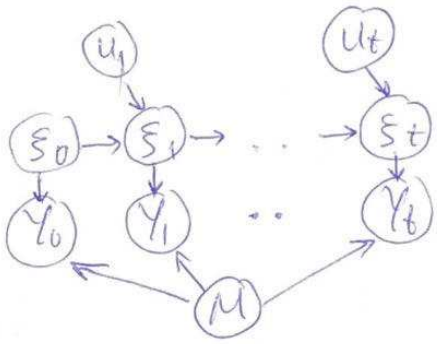
Realtime local SLAM
+
Global optimization
in a slower parallel process

+) Challenges
- wrong correspondences ⇒ divergence
- important to model uncertainties of observations

+) Formularization:

- Input: images at discrete time steps t
  control input $u_{1:t}$
- Output: Camera pose $\xi_t = \xi(T_t)$ $T_t \in SE(3)$
  environment map M
- Data association: $c_{t,i} = j$ $1 \le i \le N$, $1 \le j \le S$
  ⇒ link $y_{t,i}$ with m_j
- Probability:
  - SLAM posterior prob. $p(\xi_{0:t}, M \mid Y_{0:t}, U_{1:t})$
  - Observation likelihood $p(Y_t \mid \xi_t, M)$
  - State transition prob $p(\xi_t \mid \xi_{t-1}, U_t)$

+) Application: Robotics, Augmented / Virtual Reality, Photogrammetry, Autonomous Driving



4a) 1) Online SLAM:

- Marginalize out previous pose

$$p(\xi_t, M \mid y_{0:t}, u_{1:t}) = \int \cdots \int p(\xi_{0:t}, M \mid y_{0:t}, u_{1:t}) \, d\xi_{t-1} \cdots d\xi_0$$

[Tracking-and-Mapping: Alternating pose and map estimation

Probabilistic filters: EKF-SLAM

+) Tracking & Mapping

$$p(\xi_t \mid y_{0:t}, u_{1:t}, M) \qquad p(M \mid y_{0:t}, u_{1:t}, \xi_t)$$

- Examples: Semi-dense direct visual odometry
  Kinect Fusion
  PTAM (Parallel Tracking and Mapping

+) Filter-based Visual ~~Filte~~ SLAM

- EKF SLAM
- Mono SLAM

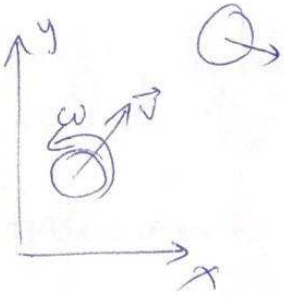
Read

2, EKF SLAM: include Landmarks into state variables

$$x_t = [\xi_t \;\; m_{t,1} \;\; .. \;\; m_{t,S}]^T, \quad \Sigma_t = \begin{bmatrix} \Sigma_{t,\xi\xi} & \Sigma_{t,\xi m_1} & .. & \Sigma_{t,\xi m_S} \\ \Sigma_{t,m_1\xi} & \Sigma_{t,m_1 m_1} & - & \Sigma_{t,m_1 m_S} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{t,m_S\xi} & \Sigma_{t,m_S m_1} & - & \Sigma_{t,m_S m_S} \end{bmatrix}$$

$$= \begin{bmatrix} \Sigma_{t,\xi\xi} & \Sigma_{t,\xi m} \\ \Sigma_{t,m\xi} & \Sigma_{t,mm} \end{bmatrix}$$

⊗ Examples EKF SLAM in 2D World:



$$\xi_t = [x_t \;\; y_t \;\; \theta_t]^T \qquad m_{t,j} = [m_{t,j,x} \;\; m_{t,j,y}]^T$$

$$u_t = [v_t, \; \omega_t]^T$$

$$\xi_t = g_\xi(\xi_{t-1}, u_t) + \epsilon_{\xi,t}, \quad \epsilon_{\xi,t} \sim \mathcal{N}(0, \Sigma_{dt,\xi})$$

Transition model

$$g_\xi(\xi_{t-1}, u_t) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} -v_t/\omega_t \cdot \sin\theta_{t-1} + v_t/\omega_t \cdot \sin(\theta_t + \omega_t \Delta t) \\ v_t/\omega_t \cdot \cos\theta_{t-1} - v_t/\omega_t \cdot \cos(\theta_t + \omega_t \Delta t) \\ \omega_t \cdot \Delta t \end{bmatrix}$$

$$m_t = g_m(m_{t-1}) = m_{t-1}$$

$$\Rightarrow x_t = g(x_{t-1}, u_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Sigma_{dt})$$

$$g(x_{t-1}, u_t) = \begin{bmatrix} g_\xi(\xi_{t-1}, u_t) \\ g_m(m_{t-1}) \end{bmatrix}, \quad \Sigma_{d_t} = \begin{bmatrix} \Sigma_{dt,\xi} & 0 \\ 0 & 0 \end{bmatrix}$$

$$y_t = h(\xi_t, m_{t,c_t}) + \delta_t, \quad \delta_t \sim \mathcal{N}(0, \Sigma_{m_t})$$

$$= [d_t, \; \phi_t]^T$$

Observation model

$$h(\xi_t, m_{t,c_t}) = \begin{bmatrix} \| m^{rel}_{t,c_t} \|_2 \\ \text{atan2}(m^{rel}_{t,c_t,y}, \; m^{rel}_{t,c_t,x}) \end{bmatrix}$$

$$m^{rel}_{t,c_t} = R(-\theta_t)\left(m_{t,c_t} - \begin{bmatrix} x_t \\ y_t \end{bmatrix}\right)$$

- State initialization: $\bar{x_0} = 0$

$$\bar{\Sigma}_{0,ss} = 0$$

New landmark: $\bar{\Sigma}_{0,sm} = \bar{\Sigma}_{0,ms} = 0$

$$\bar{\Sigma}_{0,mm} = \infty I$$

- On Prediction: covariances are transformed to the new pose
- On Correction: update all state covariances related to the pose and new landmark

+) Problems - Full covariance matrix is huge, expensive

⇒ Fast SLAM: consider landmarks to be independent
but introduce more drift (because there still actual correlation)

⇒ Use keyframes ⇒ still have correlation
but less (only between keyframes)

3) Loop closure:

- Old landmarks get re observed
  - ⇒ Add strong correlations
  - ⇒ Pose & Landmarks are corrected
  - ⇒ Reduce uncertainty

⊗ Again, wrong correspondences lead to divergence

⊛ Example Mono SLAM

48/ - Image Patches 9×9, 15×15
- Any view point with only 12 landmarks
- Map initialized with known pattern

9/ Full SLAM:

- Optimize for whole trajectory and all landmarks at once
- Uses future measurements as well to update past poses
- Allow relinearization of all state transitions & measurements at each optimization step

+/ SLAM Graph optimization:

- Just like Bundle Adjustment, but with control inputs
- Common map element observation → constraints between poses

+/ Pose Graph Optimization:

- Relative pose constraints from image observations

Deep Learning for Video Analysis

48/ ## 1) CNNs for Video Analysis

Video Classification
Image tagging
Video to Text Description

+) Architecture: different ways to fuse features from multiple frame
- Single frame
- Late fusion
- Early fusion
- Slow fusion

+) Recap - RNN: $y_n = W_{hy} \cdot h_n$

$h_n = \max\{0, W_{xh} x_n + W_{hh} h_{n-1}\}$

- LSTM: each module has 4 layers

49/ ## 2) CNNs for motion estimation

⊗ Matching & Correspondence estimation

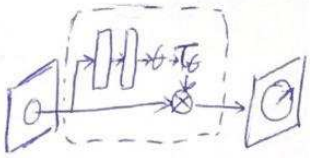
+) Learning similarity functions with Siamese Network
→ use in [Face Identification, Stereo estimation, Optical flow]

+) Metric learning: [Constrastive Loss: either pull or push
$\|f(x) - f(x_+)\| \to 0$, $\|f(x) - f(x_-)\| \geq m$
Triplet loss: pull & push at the same time

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$

+) Spatial Transformer Network : for patch normalization

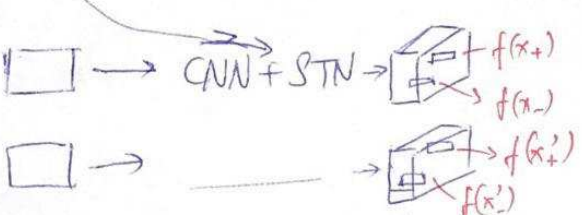
Input : a patch → output { scale, rotation



+) Universal Correspondence Network compute a patch descriptor

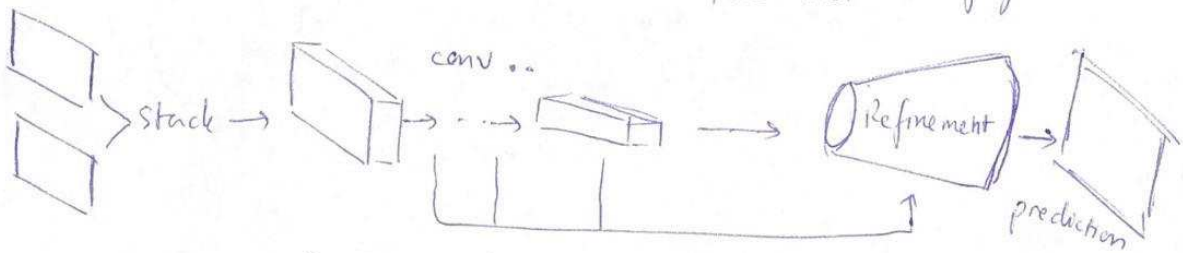
- Usage depends on input data

⇒ could be used for { stereo matching, estimating optical flow, semantic matching



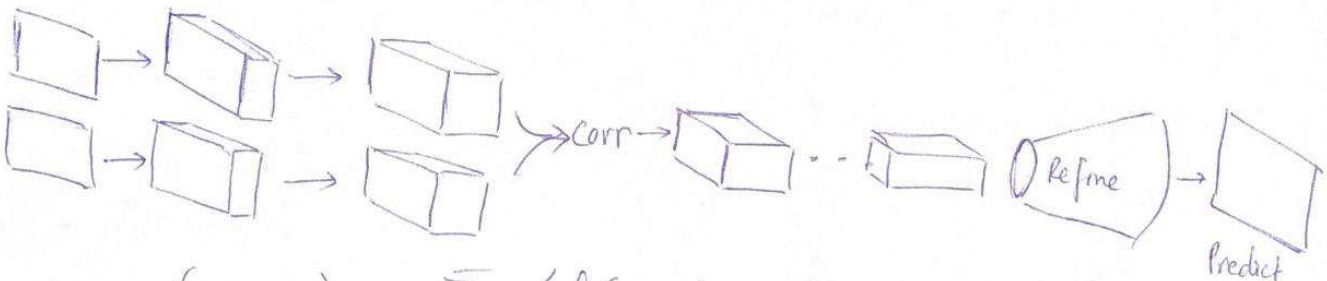
Optical Flow Estimation:

+) FlowNetS: Simple Design : Simply Stack 2 images
The network figures out itself



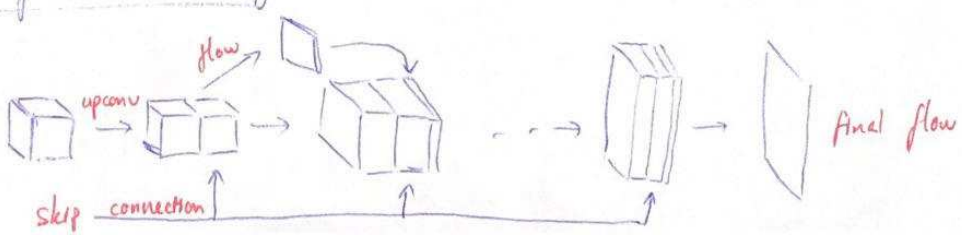
close to: pre-CNN methods but much faster

+) FlowNetC: Correlation Design



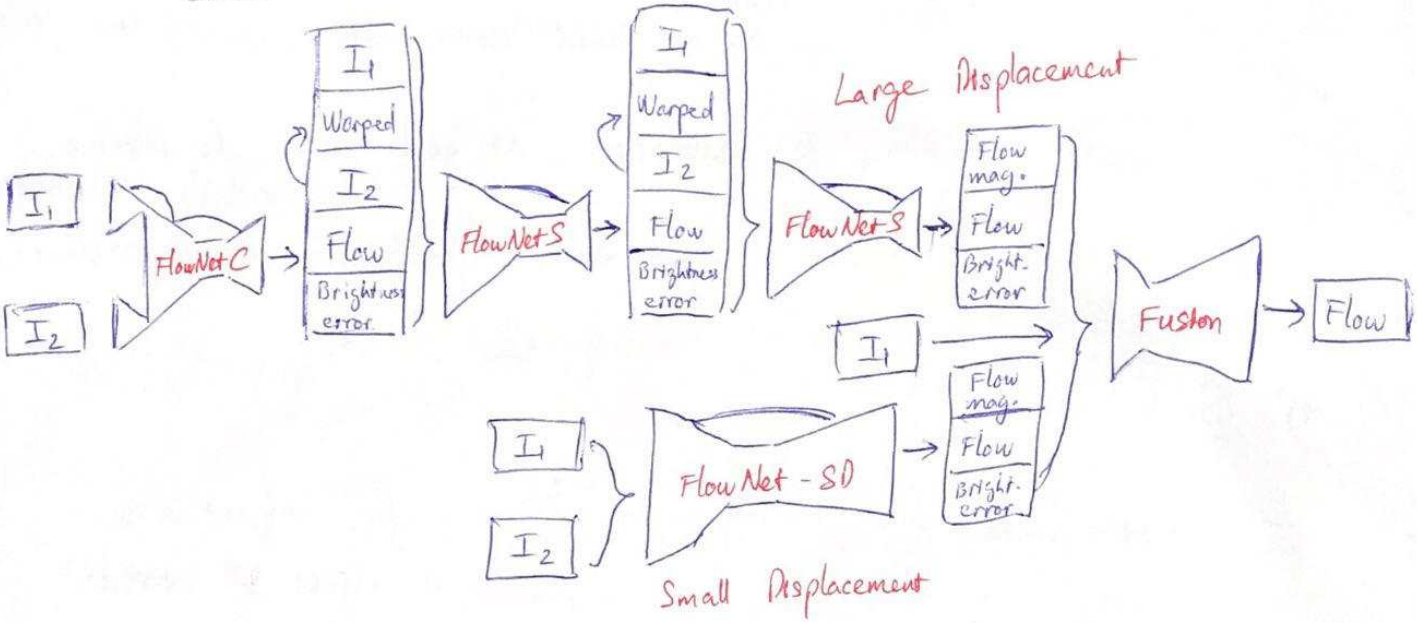
$$c(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle$$

+) Refinement stage



+) FlowNet 2.0 improved design

- Stack FlowNetC and FlowNetS



+) FlowNet-SD (FlowNet - Small Displacement)

- Replace 5×5, 7×7 by 3×3 kernels
  assume stride 1 instead of 2

⇒ FlowNet 2.0: similar accuracy as best pre-CNN methods but much faster

L20 3) Video Object Segmentation (VOS)

Object Detection ⟶ Object Tracking
↓ ↓
Object Segmentation ⟶ Video Object Segmentation

+) Task formulation:

- Semi-supervised VOS: given object mask in the first frame
  → pixel-accurate segmentation of entire video
- Unsupervised VOS (UVOS/ZVOS): no 1st-frame mask

+) Two Strategies:

- Mask propagation : given mask in 1st frame
  try to learn an online classification for that
- Proposal Tracking-by-Detection : at each frame, do instance segmentation
  then → Temporal association problem

+) Basic elements

- First-frame fine-tuning: pre-training for objectness
  then adaptation to object of interest
  Ex: OSVOS - applicable, but slow
  OSVOS-S with Semantic Prior
- Online Adaptation: not like Online Adaboost, as we do it with CNN
  ⇒ extremely slow
  Ex OnAVOS: result improved greatly
  but with great computational cost
- Mask Refinement: refine mask from previous frame
  Ex: Mask Track
- Optical Flow Mask Propagation: warp segmentation mask from one frame to the next one
- Data Augmentation: because we could overfit to 1st-frame appearance
  → crop out object, → augmentation (position, lighting, background)
  Ex: Lucid Data Dreaming
- Object Appearance Re-Identification: when object go in/out of view
  ⇒ need to re-identify objects
  Ex: ReID-VOS
- Proposal Generation: Instance Segmentation Networks
  Ex Mask-RCNN

+) PReM VOS combines all of the previous VOS principles

- Proposal generation : Mask RCNN (ResNet 101 backbone)
- Refinement : Mask Refinement
- Merging : First-frame fine tuning + Data Augmentation
  Mask propagation + Optical flow
  Re ID score

+) Combining Mask Propagation & Re-ID

- Ex RGMP (Region Guided Mask Propagation)

+) Using Recurrent NN : but with great cost for memory

Ex: STM (Space Time Memory Networks)
Seq 2 Seq

⊗ VOS to MOTS (Multi Object Tracking & segmentation)

- VOS is restricted : - 1st-frame mask given
  - short video with objects present in almost all frames
  - few objects to track

⇒ MOTS deals with these short-comings

CV 2 - Formulas

Ng Khac Duc

1, Background Modelling:

+, $I(t) - B \rightarrow abs \rightarrow$ threshold $\lambda \rightarrow M(t)$

+, $I(t) - B(t-1) \rightarrow abs \rightarrow T(\lambda) \rightarrow M(t)$

+, $D(N) = \| I(t) - I(t+N) \|$

+, $B(t) = \alpha . I(t) + (1-\alpha) B(t-1)$

$I(t) - B(t-1) \rightarrow abs \rightarrow T(\lambda) \rightarrow M(t)$

+, $\hat{\mu} = \frac{1}{N} \sum^{N} x_n$, $\hat{\sigma}^2 = \frac{1}{N-1} \sum^{N} (x_n - \hat{\mu})^2$

$$\hat{\mu}^{(t+1)} = \hat{\mu}^{(t)} + \frac{1}{N} x^{(t+1)} - \frac{1}{N} x^{(t+1-T)}$$

$$(\hat{\sigma}^2)^{(t+1)} = (\hat{\sigma}^2)^{(t)} + \frac{1}{N-1} \left[x^{(t+1)} - \hat{\mu}^{(t+1)} \right]^2 - \frac{1}{N-1} \left[x^{(t+1-T)} - \hat{\mu}^{(t+1-T)} \right]^2$$

$$\hat{\mu}^{(t+1)} = (1-\alpha) \hat{\mu}^{(t)} + \alpha . x^{(t+1)}$$

$$(\hat{\sigma}^2)^{(t+1)} = (1-\alpha)(\hat{\sigma}^2)^{(t)} + \alpha \left[x^{(t+1)} - \hat{\mu}^{(t+1)} \right]^2$$

+, $p(x) = \sum^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)$, $\Sigma_k = \sigma_k^2 I$

$$B = \arg\min_b \left(\sum_{k=1}^{b} \frac{\pi_k}{\sigma_k} > c \right)$$

+, $f_x . u + f_y . v + f_t = 0 \Rightarrow$ $\begin{bmatrix} I_x & I_y \end{bmatrix} . \begin{bmatrix} u \\ v \end{bmatrix} = - I_t$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{bmatrix} . \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

+, $\Delta p = H^{-1} . \sum_x \left(\nabla I \frac{\partial w}{\partial p} \right)^T \left[T(x) - I(w(x,p)) \right]$

$$H = \sum_x \left(\nabla I \frac{\partial w}{\partial p} \right)^T \left(\nabla I \frac{\partial w}{\partial p} \right)$$

2) Bayesian Filtering

+) $x_t \sim \mathcal{N}(D_t x_{t-1}, \Sigma_{dt})$, $y_t \sim \mathcal{N}(M_t x_t, \Sigma_{mt})$

$$\bar{x}_t = D_t x_{t-1}^+$$
$$\bar{\Sigma}_t = D_t \Sigma_{t-1}^+ D_t^T + \Sigma_{dt}$$

$$K_t = \bar{\Sigma}_t M_t^T \left(M_t \bar{\Sigma}_t M_t^T + \Sigma_{mt}\right)^{-1}$$
$$x_t^+ = \bar{x}_t + K_t (y_t - M_t \bar{x}_t)$$
$$\Sigma_t^+ = (I - K_t M_t) \cdot \bar{\Sigma}_t$$

+) $G_t = \left.\frac{\partial g(x)}{\partial x}\right|_{x = x_{t-1}^+}$, $H_t = \left.\frac{\partial h(x)}{\partial x}\right|_{x = \bar{x}_t}$

+) $func\left[\{x_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N, y_t\right]$

$\eta = 0$
for $i = 1 : N$
  $x_t^i \sim q(x_t \mid x_{t-1}^i, y_t)$
  $w_t^i = w_{t-1}^i \cdot p(y_t \mid x_t^i)$
  $\eta = \eta + w_t^i$
end
$w_t = w_t / \eta$

+) $N_{eff} = \frac{1}{\sum_{i=1}^N (w_t^i)^2}$

- uniform $N_{eff} = N$
- degenerated $N_{eff} = 1$

+) Arulampalam's Resampling Algorithm

$$\left[\{x_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}\right] = Resample\left[\{x_k^i, w_k^i\}_{i=1}^{N_s}\right]$$

3, Multi-Object Tracking

+) $\{ \underline{x}_1^{(k)}, \dots, \underline{x}_{N_k}^{(k)} \}$ and $\{ \underline{y}_1^{(k)}, \dots, \underline{y}_{M_k}^{(k)} \}$

$$z_\ell^{(k)} = j \quad \text{iff} \quad \underline{y}_j^{(k)} \leftrightarrow \underline{x}_\ell^{(k)}$$

+) Nearest Neighbor filter:

$$z_\ell^{(k)} = \underset{j}{\operatorname{argmin}} \left(\underline{x}_{p,\ell}^{(k)} - \underline{y}_j^{(k)} \right)^T \left(\underline{x}_{p,\ell}^{(k)} - \underline{y}_j^{(k)} \right)$$

better $= \underset{j}{\operatorname{argmin}} \; \underline{v}_{j,\ell}^{(k)T} \cdot \left(\Sigma_{p,\ell}^{(k)} \right)^{-1} \cdot \underline{v}_{j,\ell}^{(k)}$

with innovation $\underline{v}_{j,\ell}^{(k)} = \underline{y}_j^{(k)} - \underline{x}_{p,\ell}^{(k)}$

+) Track-splitting filter

$$L(\theta_{k,\ell}) = \prod_{j=1}^{k} p\left(z_{i_j,\ell} \mid z_{(j-1),i}, \theta_{k,\ell} \right)$$

$$\lambda_\ell(k) = \lambda_\ell(k-1) + \underline{v}_{i_k,\ell}^{(k)T} \cdot \left(\Sigma_\ell^{(k)} \right)^{-1} \cdot \underline{v}_{i_k,\ell}^{(k)}$$

+) MHT: $p\left(\Omega_j^{(k)} \mid \underline{Y}^{(k)} \right) = \eta \cdot p\left(\underline{Y}^{(k)} \mid z_j^{(k)}, \Omega_{p(j)}^{(k-1)} \right) \cdot p\left(z_j^{(k)} \mid \Omega_{p(j)}^{(k-1)} \right) \cdot p\left(\Omega_{p(j)}^{(k-1)} \right)$

+) LAP: $\sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} z_{ij}$, $\sum_{j=1}^{N} z_{ij} = 1$, $\sum_{i=1}^{M} z_{ij} = 1$

+) Network flow optimization:

$$\mathcal{T}^* = \underset{\mathcal{T}}{\operatorname{argmin}} \sum_i C_{in,i} \cdot f_{in,i} + \sum_i C_{i,out} \cdot f_{i,out} + \sum_{i,j} C_{ij} f_{ij} + \sum_i C_i \cdot f_i$$

source sink association detection

flow conservation: $f_{in,i} + \sum_j f_{j,i} = f_i = f_{out,i} + \sum_j f_{ij} \quad \forall i$

edges capacities: $f_i \leq 1$

4) Visual Odometry:

1) Cameras:

- Monocular cameras $I_{0:t} = \{I_0, \dots I_t\}$
- Stereo cameras $I^l_{0:t} = \{I^l_0, \dots I^l_t\}$

$$I^r_{0:t} = \{I^r_0, \dots I^r_t\}$$

- Active RGB-D camera $I_{0:t} = \{I_0, \dots I_t\}$

$$Z_{0:t} = \{Z_0, \dots Z_t\}$$

2) Direct Methods

Indirect

$$E(\xi) = \int_{u \in \Omega} |I_1(u) - I_2(w(u,\xi))| du$$

$$E(\xi) = \sum_i |y_{1,i} - w(y_{2,i}, \xi)|$$

Error

2D to 2D
$$E(T_t^{t-1}, X) = \sum_{i=1}^{N} \|\bar{y}_{t,i} - \pi(\bar{x}_i)\|_2^2 + \|\bar{y}_{t-1,i} - \pi(T_t^{t-1} \cdot \bar{x}_t)\|_2^2$$

2D to 3D
$$E(T_t) = \sum_{i=1}^{N} \|y_{t,i} - \pi(T_t \cdot \bar{x}_i)\|_2^2$$

3D to 3D
$$E(T_t^{t-1}) = \sum_{i=1}^{N} \|\bar{x}_{t-1,i} - T_t^{t-1} \cdot \bar{x}_{t,i}\|_2^2$$

3) Eight point algorithm:

$\tilde{y}_i \, E \, \tilde{y}_i' = 0 = a_i E_s \Rightarrow A \cdot E_s = 0$

$\Rightarrow A = U_A \Sigma_A V_A^T \Rightarrow E_s$: last column of V_A

$E = U \cdot \text{diag}(\sigma_1, \sigma_2, \sigma_3) \cdot V^T$

$\Rightarrow E = U \cdot \text{diag}(1, 1, 0) \cdot V^T$

$$\Rightarrow \begin{cases} R = U \cdot R_Z^T\left(\pm \frac{\pi}{2}\right) \cdot V^T \\ \hat{t} = U \cdot R_Z\left(\pm \frac{\pi}{2}\right) \text{diag}(1,1,0) \cdot U^T \end{cases}$$

4) Relative scale recovery

$$r_{i,j} = \frac{\|x_{t-2,t-1,i} - x_{t-2,t-1,j}\|_2}{\|x_{t-1,t,i} - x_{t-1,t,j}\|_2}$$

+) DLT: $\tilde{y}_i = [x_i' \quad y_i' \quad w_i']^T$, $\tilde{x}_i = [x_i \quad y_i \quad z_i \quad w_i]^T$

$\Leftrightarrow$ 2 constraints: $\begin{bmatrix} 0 & -w_i' \tilde{x}_i^T & y_i' \tilde{x}_i^T \\ w_i' \tilde{x}_i^T & 0 & -x_i' \tilde{x}_i^T \end{bmatrix} \cdot \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0$

from $\tilde{y}_i \times (P\tilde{x}_i) = 0$

+) Closed-form solution of 3D → 3D

$\mu_{t-1} = \frac{1}{N}\sum_{i=1}^{N} x_{t-1,i}$, $\mu_t = \frac{1}{N}\sum_{i=1}^{N} x_{t,i}$

Set $A = \sum_{i=1}^{N} (x_{t-1} - \mu_{t-1})(x_t - \mu_t)^T$, $A = U.\Sigma.V^T$

→ $R_{t-1}^t = V.U^T$ & $t_{t-1}^t = \mu_t - R_{t-1}^t \mu_{t-1}$

+) Direct Methods

photometric error $I_1(y) = I_2(\pi(T(\xi).Z_1(y).\bar{y}))$

Geometric error $[T(\xi)Z_1(y)\bar{y}]_z = Z_2(\pi(T(\xi).Z_1(y)\bar{y}))$

$E(\xi) = \sum_{y \in \Omega} \frac{r(y,\xi)^2}{\sigma_I^2}$, $r(y,\xi) = I_1(y) - I_2(\pi(T(\xi)Z_1(y).\bar{y}))$

minimize the residual

$= r(\xi)^T . W . r(\xi)$

+) Gauss-Newton: $\xi_{i+1} = \xi_i - [\nabla_\xi^2 E(\xi_i)]^{-1} \nabla_\xi E(\xi_i)$

$= \xi_i - H_i^{-1} . J_i^T W r(\xi_i)$

+) Levenberg-Marquardt: $\xi_{i+1} = \xi_i - (H_i + \lambda I)^{-1} J_i^T W r(\xi_i)$

+) Lie algebra: $\xi = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6$, $\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}$

$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}$

$T = \exp(\hat{\xi}) = \begin{bmatrix} \exp(\hat{\omega}) & Av \\ 0 & 1 \end{bmatrix}$

$\hat{\xi} = \log(T) = \begin{bmatrix} \log R & A^{-1}t \\ 0 & 0 \end{bmatrix}$

+) Contrastive Loss:

$$\|f(x) - f(x_+)\| \to 0$$

$$\|f(x) - f(x_-)\| \geq m$$

+) Triplet loss:

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$