Probabilistic Context Free Grammars for Representing Action

Song Mao

November 14, 2000

Action Understanding in Computer Vision

- Interpretation of basic movements
  - Sitting, walking , running …
- Description of motion of several objects
- Recognition of gestures
- High-level event

High-level Events

- Consists of primitives
  - For car drop-off event: car-enter, car-stop, person-enter, person-exit, etc.
- Spatio-temporal structure & constraint
- Semantically defined activities
- Span extended periods of time
- Multi-object interactions

Approaches

- Statistical techniques
- Syntactic techniques
- Methods that combine the two techniques

Statistical Techniques

- Classifying pattern by assuming an statistical model
  - Tennis stroke recognition
  - Gesture recognition
  - Visual language recognition
- Advantages
  - Real world data are noisy in nature (signal noise)
  - Uncertainty in observation (sensor noise)

Statistical Techniques (cont.)

- Disadvantages
  - Insufficient data
  - Semantic ambiguity
  - Temporal ambiguity
  - Known structure

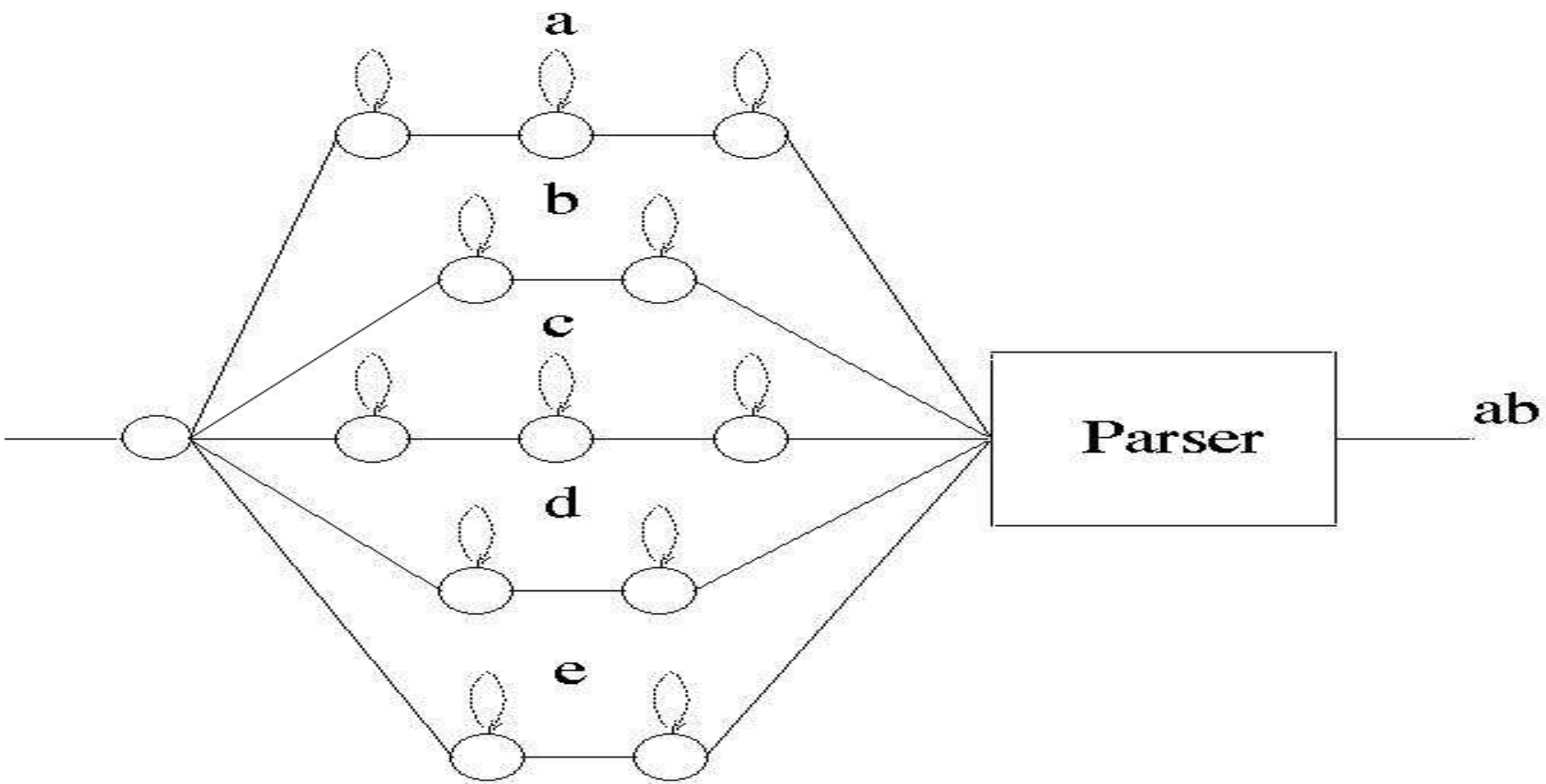
Syntactic Techniques

- Describe pattern structure
- Formal grammar
- Context free grammar (CFG)
- Stochastic context free grammar (SCFGs)
- Parsing

Combine the Two Techniques

- Independent primitives detection using statistical techniques
- Actions (structured primitives) recognition by syntactic techniques
  - Parsing primitives by SCFGs
  - Removing ambiguity by parsing SCFGs
  - Correcting errors (substitution, insertion, deletion) by adding SKIP rules and penalty function to SCFGs

Decoupling Primitive Detection and Primitive Structuring



Parsing

- What is Parsing?

  The process of taking an input and producing some sort of structure for it.(Jurafsky & Martin)

- Structure assigned by Context Free Grammar (CFG) / Stochastic Context Free Grammar (SCFG)

Parsing Approaches

- Top-down approach
- Bottom-up approach
- Dynamic programming approach
  - Cocke-Younger-Kasami parser (CYK)
  - Graham-Harrison-Ruzzo parser (GHR)
  - Earley parser

Context Free Grammars (CFG)

- A set of non-terminal symbols N
- A set of terminal symbols Σ
- A set of productions P of form $A \to \alpha$

  $A \in N, \alpha \in (\Sigma \cup N)^*$
- Start symbol S
- Directly derivation:

  if $A \to \beta$, $\alpha, \gamma \in (\Sigma \cup N)^*$,

  $\alpha A \gamma \Rightarrow \alpha \beta \gamma$

Context Free Grammar (Cont.)

- Derivation: $\alpha_1 \to \alpha_2, \alpha_2 \to \alpha_3, \ldots, \alpha_{m-1} \to \alpha_m,$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

- A language generated by a grammar G

$$\{L_G = w \mid w \in \Sigma^*, S \overset{*}{\Rightarrow} w\}$$

Stochastic Context Free Grammar (SCFG)

- Modify production as: $A \rightarrow \lambda [p]$
- Where $p = P(A \rightarrow \lambda \mid A)$ is the rule probability of the production $A \rightarrow \lambda$ from a Context-Free Grammars (CFGs)
- Rules are conditionally independent

$$P(A \rightarrow \lambda, B \rightarrow \beta, C \rightarrow \gamma \mid A, B, C)$$
$$= P(A \rightarrow \lambda \mid A) \cdot P(B \rightarrow \beta \mid B) \cdot P(C \rightarrow \gamma \mid C)$$

Earley Parsing Algorithm

- A set of states for each position in the input
- Dot denotes the current input position
- A state with the dot at the right most position is a complete state
- A state produced by prediction is a predicted state
- A state produced by completion is a completed state

Earley Parsing Algorithm (cont.)

- A State: $i : X_k \rightarrow \lambda \bullet Y\mu$

- Prediction:

$$\begin{cases} i : X_k \rightarrow \lambda \bullet Y\mu, \\ Y \rightarrow \nu, \end{cases} \Rightarrow \quad i : Y_i \rightarrow \bullet \nu$$

Earley Parser (cont.)

- Scanning: $i: X_k \to \lambda \bullet a\mu \Rightarrow i+1: X_k \to \lambda a \bullet \mu$
- Completion:

$$\begin{cases} j: X_k \to \lambda \bullet Y\mu, \\ i: Y_j \to \nu \bullet, \end{cases} \Rightarrow i: X_k \to \lambda Y \bullet \mu$$

An Example

S → NP VP
S → Aux NP VP
S → VP
NP → Det Noun
VP → Verb

Verb → book
Det → that
Noun → flight
NP → ProperNoun
VP → Verb VP

state set (0)

0 : 0_0 →• S
predicted
0 : S_0 →• NP VP
0 : NP_0 →• Det Noun
0 : NP_0 →• ProperNoun
0 : S_0 →• Aux NP VP
0 : S_0 →• VP
0 : VP_0 →• Verb
0 : VP_0 →• Verb NP

(1) Book

scanned
1 : $Verb_0$ → book•
completed
1 : VP_0 → Verb•
1 : S_0 → VP•
1 : VP_0 → Verb• NP
predicted
1 : NP_1 →• Det Noun
1 : NP_1 →• ProperNoun

(2) that

scanned
2 : Det_1 → that•
completed
2 : NP_1 → Det• Noun

(3) flight

scanned
3 : $Noun_2$ → flight•
completed
3 : NP_1 → Det Noun•
3 : VP_0 → Verb NP•
3 : S_0 → VP•

Earley-Stolcke Parser (1)

- A state $i : X_k \rightarrow \lambda \bullet Y\mu \ [\alpha, \gamma]$
- Forward probability
  $\alpha = \sum_{path} P(\text{path of length i that ends in state } i : X_k \rightarrow \lambda \bullet Y\mu)$
- Inner probability

  $\gamma = \sum_{path} P(\text{path of length } i-k \text{ that start in state } k : X_k \rightarrow \bullet \lambda Y\mu$
  $\text{and end in state } i : X_k \rightarrow \lambda \bullet Y\mu)$

- Earley path: a sequence of states needed to reach the current state
- Length of path: number of scanning states

Earley-Stolcke Parser (2)

- Prediction

$$\begin{cases} i : X_k \rightarrow \lambda \bullet Z\mu\,[\alpha, \gamma] \\ Y \rightarrow \nu, \end{cases} \Rightarrow \quad i : Y_i \rightarrow \bullet\, \nu\,[\alpha', \gamma']$$

where

$$\alpha' = \sum_{\lambda,\mu} \alpha(i : X_k \rightarrow \lambda \bullet Z\mu) \cdot R_L(Z, Y) \cdot P(Y \rightarrow \nu)$$

$$\gamma' = P(Y \rightarrow \nu)$$

Compute R_L

- Left-recursion in grammar
- Possibly infinite prediction loop that accumulate probability computation
- Example: $A \rightarrow Aa,\ A \rightarrow a$
- Left Corner relation: $X \rightarrow_L Y,$ iff $\forall\ X \rightarrow Y\lambda$

- $R_L(Z,Y) = P_L(Z \overset{0}{\Rightarrow} Y) + P_L(Z \overset{1}{\Rightarrow} Y) + \ldots$
- $P_L(Z \overset{k}{\Rightarrow} Y) = P_L(Z \Rightarrow Y_1 \Rightarrow Y_2 \Rightarrow \ldots \Rightarrow Y_{k-1} \Rightarrow Y)$

Compute R_L (cont.)

- Matrix form

$$R_L = P_L^0 + P_L^1 + \ldots = \sum_{k=0}^{\infty} (I - P_L)^{-1}$$

- Computed once for the grammar, and used at each iteration of the prediction step

Earley-Stolcke Parser (3)

- Scanning

$$i : X_k \to \lambda \bullet a\mu \ [\alpha, \gamma] \Rightarrow i+1 : X_k \to \lambda a \bullet \mu \ [\alpha, \gamma]$$

- Completion

$$\begin{cases} j : X_k \to \lambda \bullet Z\mu [\alpha, \gamma], \\ i : Y_j \to \nu \bullet [\alpha'', \gamma''], \end{cases} \Rightarrow i : X_k \to \lambda Z \bullet \mu [\alpha', \gamma']$$

where

$$\alpha' = \sum_{\nu} \alpha(j : X_k \to \lambda \bullet Z\mu) \cdot R_U(Z, Y) \cdot \gamma''(i : Y_j \to \nu \bullet)$$

$$\gamma' = \sum_{\nu} \gamma(j : X_k \to \lambda \bullet Z\mu) \cdot R_U(Z, Y) \cdot \gamma''(i : Y_j \to \nu \bullet)$$

Compute R_U

- Unit production: $X \rightarrow_U Y$, iff $X \rightarrow Y$
- Infinite completion by unit production
  - e. g. $A \rightarrow B,\ A \rightarrow a,\ B \rightarrow A$
- Unit production relation matrix P_U
- Similarly as computing R_L in prediction step

$$R_U = P_U^0 + P_U^1 + \ldots = \sum_{k=0}^{\infty} (I - P_U)^{-1}$$

Uncertainty in the Input

- Source of the input symbols is probabilistic
- Modify scanning of the Earley-Stolcke parser

$$\begin{cases} i : X_k \rightarrow \lambda \bullet a\mu \ [\alpha, \gamma] \Rightarrow \ i+1 : X_k \rightarrow \lambda a \bullet \mu \ [\alpha', \gamma'] \\ \forall a, \text{ s.t. } \ P(a) > 0 \end{cases}$$

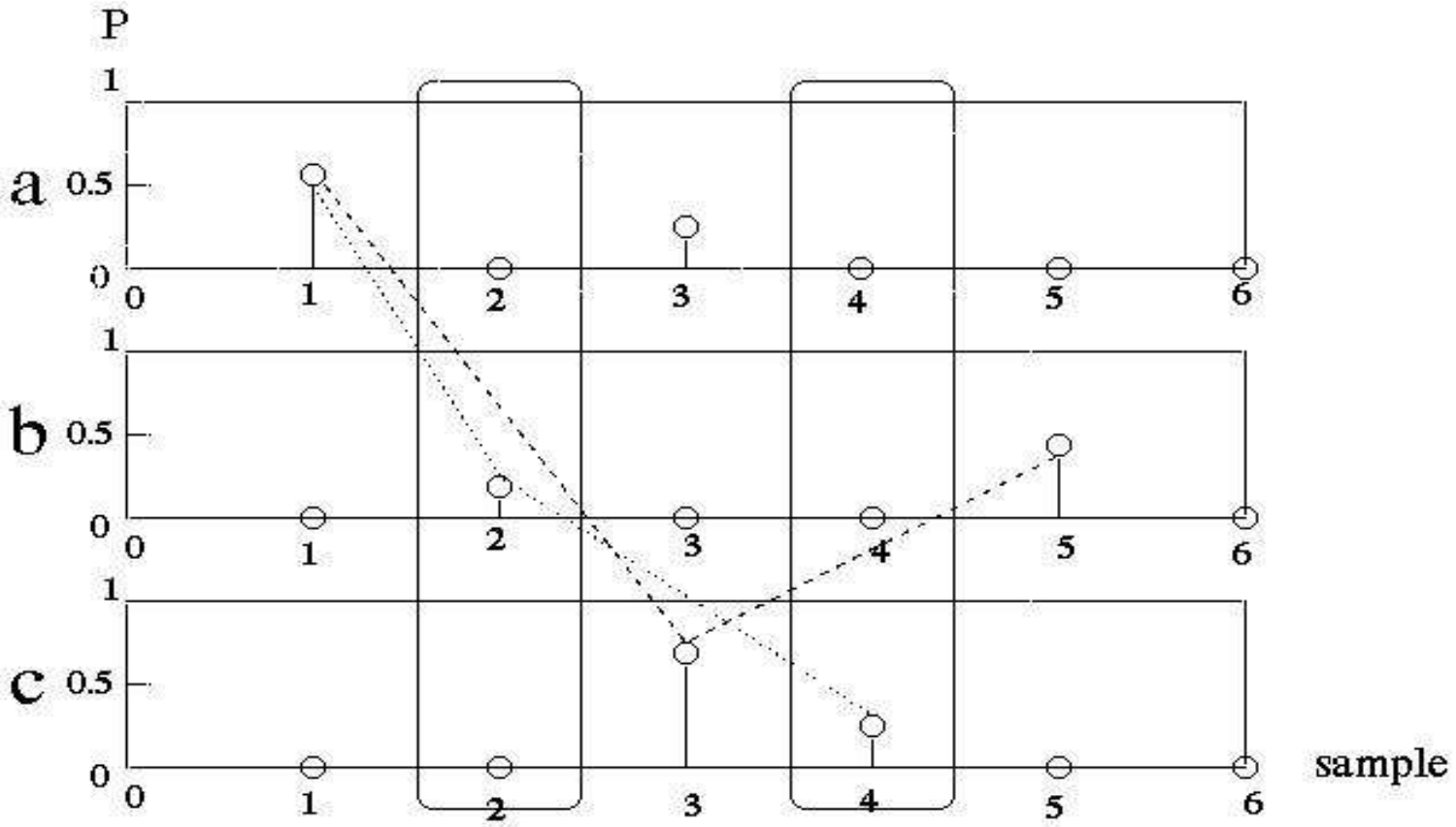
$$\alpha' = \alpha\,(i : X_k \rightarrow \lambda \bullet \alpha\mu\,) \cdot P(a)$$

$$\gamma' = \gamma\,(i : X_k \rightarrow \lambda \bullet \alpha\mu\,) \cdot P(a)$$

- Address the substitution error

An Example for a Grammar

$$A \rightarrow abc \mid acb$$



Insertion and Deletion

- Use a robust form grammar $\hat{G}$ of G

  G : $\Rightarrow$ $\hat{G}$

  $A \rightarrow bC$ $\quad$ $A \rightarrow \hat{B}C$

- $\hat{B} \rightarrow b \mid SKIP\ b \mid b\ SKIP$
- $SKIP \rightarrow b \mid c \mid ... \mid b\ SKIP \mid c\ SKIP \mid ...$
- Includes all repetitions of all terminals
- Set $P(SKIP\ rule)$ small
- Penalize derivation consuming less terminals

Enforcing Consistency (1)

- Types of consistency
  - Temporal consistency
  - Spatial consistency
  - Object identity consistency
- Add 2 vector valued state variables
  - l : low mark
  - h : high mark
- Containing the data for computing distance penalty between two joining states

Enforcing Consistency (2)

- Prediction

$$\begin{cases} i : X_k \to \lambda \bullet Z\mu\,[l,h] \\ Y \to \nu, \end{cases} \Rightarrow \quad i : Y_i \to \bullet\, \nu\,[S_t, S_t]$$

- Scanning

$$i : X_k \to \lambda \bullet a\mu\,[l,h] \Rightarrow$$

$$\begin{cases} i+1 : X_k \to \lambda a \bullet \mu\,[l_a, h_a] \text{ if } \lambda = \varepsilon \\ i+1 : X_k \to \lambda a \bullet \mu\,[l, h_a] \text{ else} \end{cases}$$

Enforcing Consistency (3)

- Completion

$$\alpha' = f(d)\sum_{\nu}\alpha(j: X_k \to \lambda \bullet Z\mu)\cdot R_U(Z,Y)\cdot \gamma''(i: Y_j \to \nu\bullet)$$

$$\gamma' = f(d)\sum_{\nu}\gamma(j: X_k \to \lambda \bullet Z\mu)\cdot R_U(Z,Y)\cdot \gamma''(i: Y_j \to \nu\bullet)$$

- $f(d)$: distance penalty function
- Computed based on high mark of completed state and low mark of completing state

Choice of $f(d)$

- Sever penalty: step function
  - e. g.

$$f(d) = \begin{cases} 0, & d < 0 \\ 1, & \text{else} \end{cases}$$

- Softer penalty: exponential function
  - e. g.

$$f(d) = Ce^{-\frac{d^2}{\theta}}$$

Application:
Vedio Surveillance of Parking Lot

- Outdoor environment – occlusions and lighting change
- Static cameras
- Real-time performance
- Labeling activities and person-vehicle interactions in a parking lot
- Handling simultaneous events

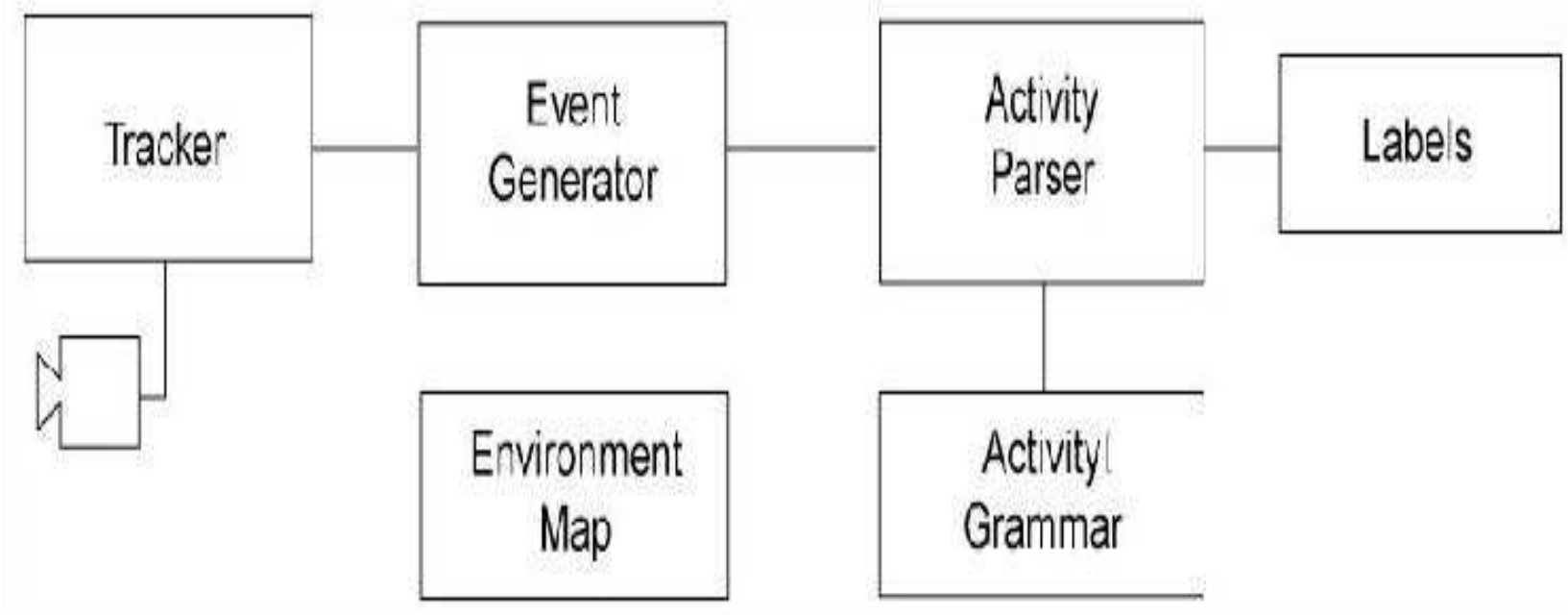
Known Structure, Uncertain Elements

- Activities as sequences of primitives represented by SCFG
  - Car drop-off, car pick-up
  - Dancing
- Input primitives are uncertain
  - Uncertain observation of primitive
  - Noisy symbols

Approach

- First detect primitives using statistical method
  - Tracker
  - Event generator
- Then Recognize activity by parsing input stream of uncertain primitives (partial tracks) by an SCFG parser

System Overview



System Overview (cont.)

- Tracker
  - Assign identity to the moving objects
  - Collects the trajectory data into partial tracks
- Event generator
  - Maps partial tracks onto predetermined set of events
- Parser
  - Labels sequences of events by parsing using a SCFG
  - Enforce consistency constraint

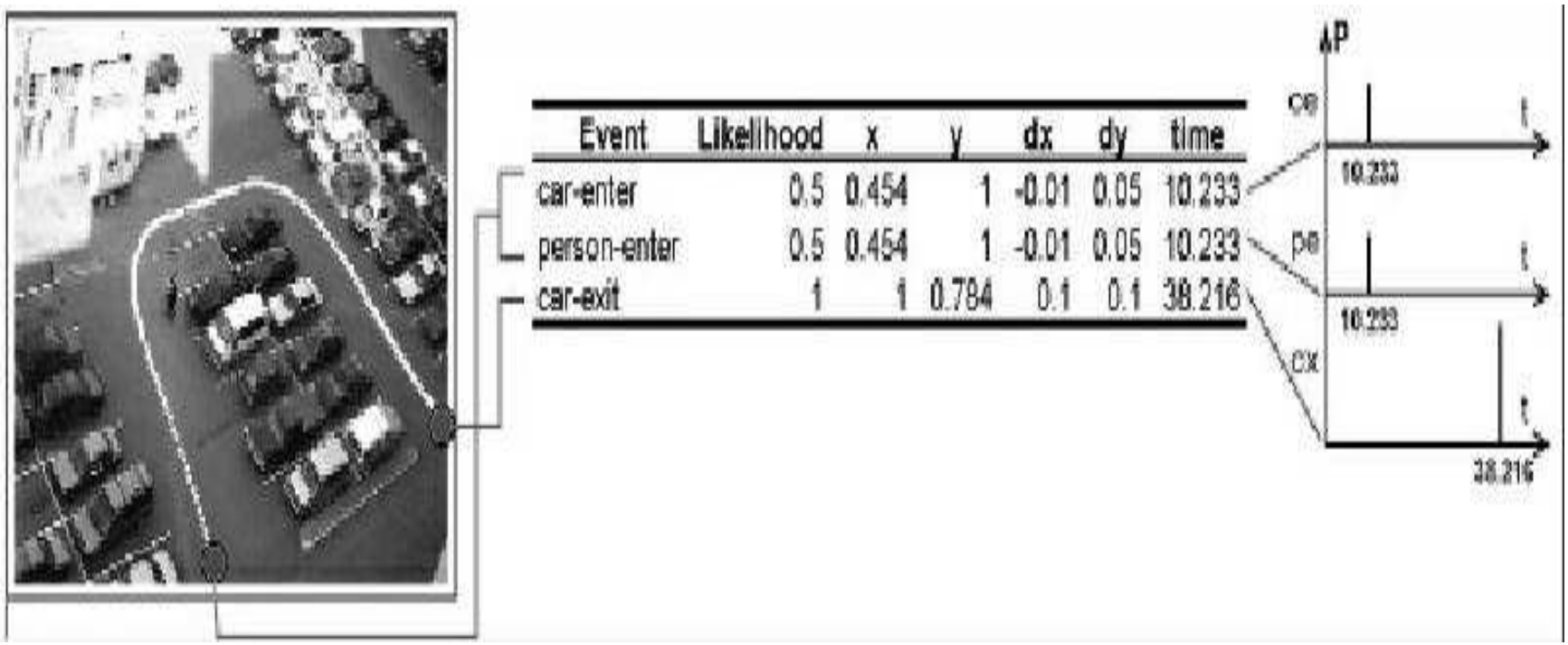
Tracker

- Object found
- Assign a unique ID
- Track changes in objects' appearance, position, velocity
- Based on the data, assign each object a class label (e.g. a car or a person)
- Object lost
- Object exit

Event Generator

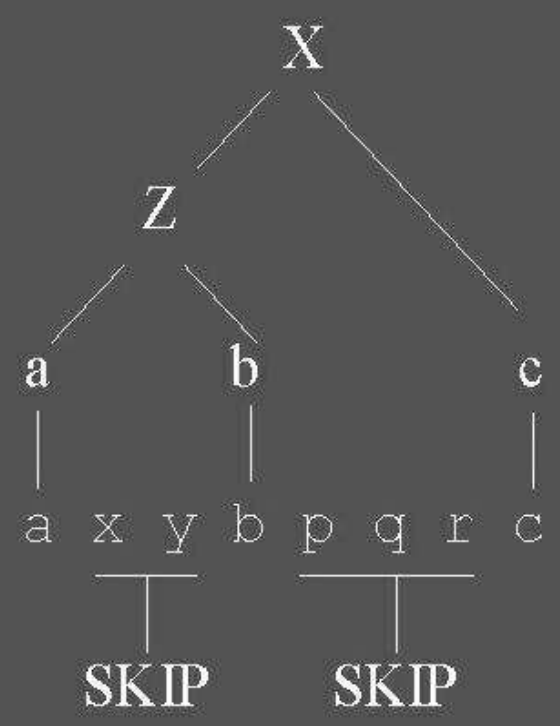
- Based on data from tracker
  - Object-enter
  - Object-found
  - Object-exit
  - Object-lost
  - Object-stopped
- Initially, tracker can not figure out class label,
- When object exit, tracker has enough information to assign a class label to the object

An Example of Generating Events



Event	Likelihood	x	y	dx	dy	time
car-enter	0.5	0.454	1	-0.01	0.05	10.233
person-enter	0.5	0.454	1	-0.01	0.05	10.233
car-exit	1	1	0.784	0.1	0.1	38.216

Parsing Events



$X \rightarrow Zc$

$Z \rightarrow ab$

- production rules (states)

X - target non-terminal (label)

Z - intermediate non-terminal

- input stream (tracking events)

- noise rules

For the production X, events a, b and c should be consistent

Sample Stochastic Context-Free Grammar

Partial Listing:

```
TRACK              ->        CAR-TRACK          [0.50]
                           | PERSON-TRACK       [0.50]

CAR-TRACK          ->        CAR-THROUGH        [0.25]
                           | CAR-PICKUP         [0.25]
                           | CAR-OUT            [0.25]
                           | CAR-DROP           [0.25]
CAR-THROUGH . . .

                     . . .

CAR-EXIT           ->        car-exit           [0.70]
                           | SKIP car-exit      [0.30]
```

Non-terminals - semantically significant groups of events

Terminals - individual events

SKIP-rule - noise symbol

Rule probabilities

Tracker and Event Generator Data for Parser States

- Tracker event generator provides data for "low mark" and "high mark" of parser states

$$l = \begin{pmatrix} f_l \\ t_l \\ x_l \\ y_l \\ dx_l \\ dy_l \end{pmatrix} \quad h = \begin{pmatrix} f_h \\ t_h \\ x_h \\ y_h \\ dx_h \\ dy_h \end{pmatrix}$$

f: frame number

t: timing stamp

(x,y): location

(dx,dy): velocity

Distance Penalty Function

- t_1, r_1, dr_1: high mark data of state being completed
- t_2, r_2 :low mark of the completing state

$$f(t_1, t_2, r_p, r_2) = \begin{cases} 0, \text{ if } (t_2 - t_1) < 0 \\ \exp\left(\dfrac{(r_2 - r_p)^T (r_2 - r_p)}{\theta} \right), \text{else} \end{cases}$$

- Where $r_p = r_1 + dr_1(t_2 - t_1)$ is the predicted position of the object at time t_2

An Example

Completion - check if Z can be used as a supporting evidence for X:

$$i: X_k \rightarrow \lambda . Z \mu \quad [\alpha_1, \gamma_1, \mathbf{l}_1, \mathbf{h}_1]$$

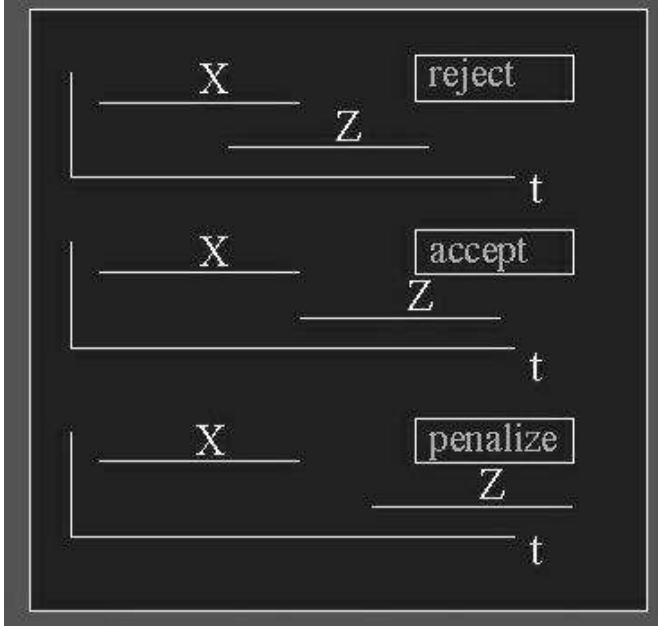
?

$$j: Z_i \rightarrow \eta . \quad [\alpha_2, \gamma_2, \mathbf{l}_2, \mathbf{h}_2]$$

Legend:

$\mathbf{r} = (x, y)$

$d\mathbf{r} = (dx, dy)$



Predict new position:

$$\mathbf{r}_p = \mathbf{r}_1 + d\mathbf{r}_1 (t_2 - t_1)$$

Penalize:

$$f(\mathbf{r}_p, \mathbf{r}_2) = \begin{cases} 0, & \text{if } (t_2 - t_1) < 0 \\ \exp\left(\dfrac{(\mathbf{r}_2 - \mathbf{r}_p)^T (\mathbf{r}_2 - \mathbf{r}_p)}{\theta} \right) \end{cases}$$

Events Data for Drive-In and Drop-Off Activities

DROP-OFF

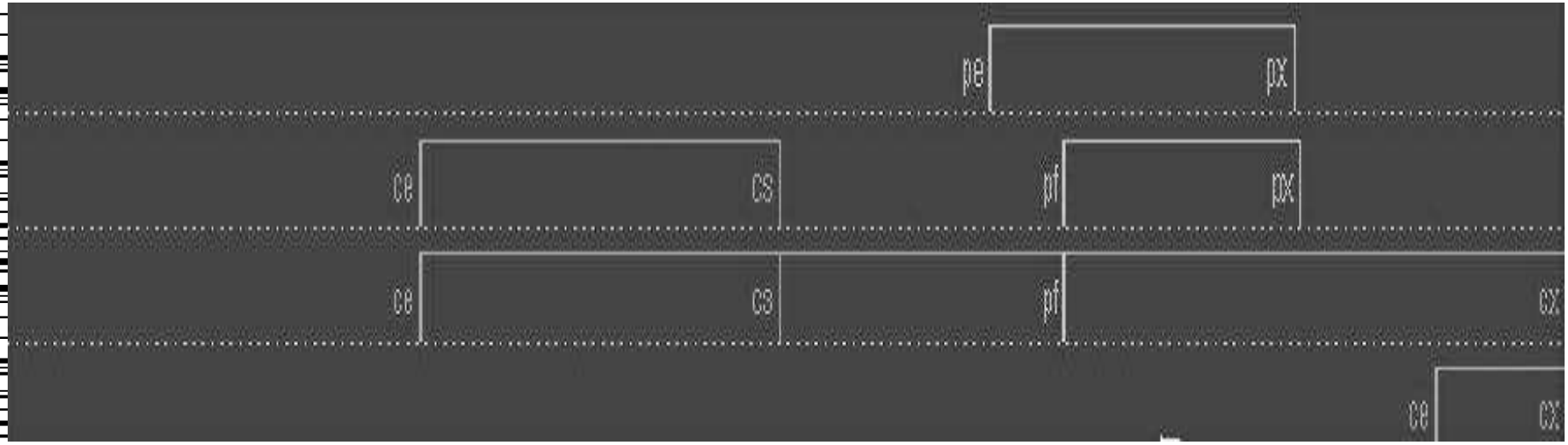
DRIVE-IN

Event	UID	Avg. Size	Class	P	x	y	t	frame
ENTER	724	0.122553	0	0.5	0.450094	0.938069	917907137.8	1906
ENTER	665	0.046437	1	0.5	0.6107	0.94674	917907122.5	1799
PERSON-LEAVE	665	0.045869	1	0.997846	0.648089	0.98855	917907142.7	1938
STOPPED	724		0	0.995784	0.348569	0.345513	917907146.5	1964
ENTER	780	0.034293	1	0.5	0.74188	0.980292	917907151.3	1998
ENTER	790	0.069093	0	0.5	0.814565	0.032611	917907153.4	2012
FOUND	787	0.033573	1	0.5	0.297585	0.357887	917907153.1	2010
CAR-LEAVE	790	0.061263	0	0.997285	0.975971	0.211984	917907155.3	2025
PERSON-LEAVE	780	0.038616	1	0.999923	0.974494	0.865237	917907158.6	2047
PERSON-LEAVE	787	0.032045	1	0.999997	0.296519	0.183704	917907158.7	2048
ENTER	813	0.034776	1	0.5	0.012821	0.348379	917907160.9	2063
ENTER	816	0.093513	0	0.5	0.960425	0.793899	917907161.9	2070
CAR-LEAVE	724	0.097374	0	0.993211	0.972272	0.693728	917907165.2	2091
CAR-LEAVE	816	0.089424	0	0.99023	0.693699	0.990798	917907165.2	2091

Vedio Frame Illustration

Person passed through

Person drove in

Person drop off

Car passed through