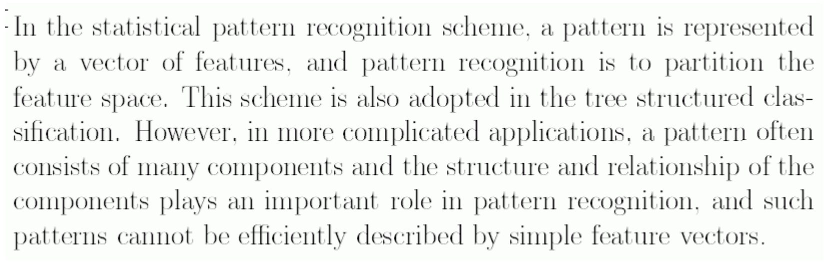
Lecture 11: Syntactic pattern recognition


- In the statistical pattern recognition scheme, a pattern is represented by a vector of features, and pattern recognition is to partition the feature space. This scheme is also adopted in the tree structured classification. However, in more complicated applications, a pattern often consists of many components and the structure and relationship of the components plays an important role in pattern recognition, and such patterns cannot be efficiently described by simple feature vectors.

For example, given an image/video, we ask:

1, Is this a *vehicle* ? // Object level
2, Is this a *birthday party* ? // Scene level
3, What is the *person doing* in the video ? // Event level

The definition of such classes relies on the compositional structures which may not be represented by a vector of fixed length, and
the recognition of such classes requires "hierarchical parsing" not merely a classification.

Syntactic pattern recognition

In the 1970s, K.S. Fu et al studied many stochastic grammars for *syntactic pattern recognition*, like curves, waveforms, handwritten digits etc. These patterns have intrinsic 1D structures. Fu etc. adopted the parsing algorithms from compiler theory in computer science and he also developed some error correcting techniques.

This research stream was out of favor in late 1980s due to its limited power for representing images. People turned to study appearance-based representation.

Most recently, there is a resurgence of grammatical methods in vision.

The basics of grammar

A grammar is often written as a quadruple $G=<V_N, V_T, R, s>$

A set of non-terminal nodes V_N, terminal nodes V_T, production rules R, and an initial node s.

A sentence or string *w* is valid if it can be derived by production rules in finite steps.

$$s \xrightarrow{R^*} w, \quad w \in V_T^*$$

The * sign means multiples: $V_T^* = \cup_{0\le n<\infty} V_T^n, \quad V_T^n = \underbrace{V_T \times \cdots \times V_T}_{n}$

The set of all valid sentences or strings is called its *language.*

$$L(G) = \{w : s \xrightarrow{R^*} w, \quad w \in V_T^*\}$$

Four types of string grammars

Four types of string grammars by the generality of their production rules:

Type 3 Finite-State grammar (finite state automation).

Its rules are of the following form, and each time it only expands one terminal node.
One typical example is the Hidden Markov Model, the hidden state follows a *Markov chain*.

$$\boxed{A \to aB \quad or \quad A \to b \qquad a,b \in V_T, A, B \in V_N}$$

Type 2 Context-free grammar.

Its rules are of the following form, and each time it expands a non-terminal node independent of context.
This leads to the Markov tree models and is also called the branching process (in continuous form).

$$\boxed{A \to \beta \qquad \beta \in (V_T \cup V_N)^+, \quad A \in V_N}$$

Type 1 Context-sensitive grammar.

Its rules are of the following form, and each time it expands a non-terminal node with its context.

$$\boxed{\xi_L A \xi_R \to \xi_L \beta \xi_R \qquad \beta \in (V_T \cup V_N)^+, \quad A \in V_N}$$

Type 0 General grammar with no limitations on its production rules.
It is believed that natural languages belong to type 0.

By Chomsky, 1957.

Example of image parsing by grammar

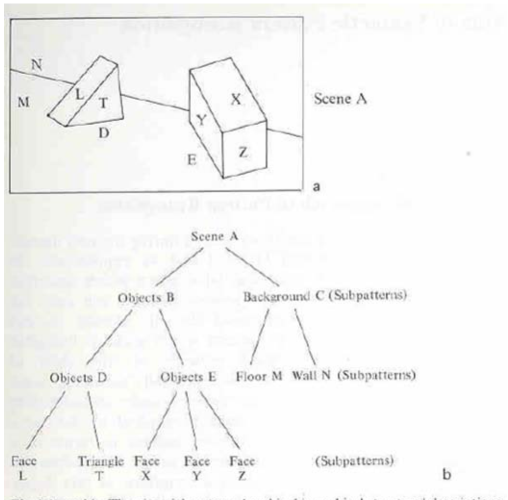


Fig. 1.1a and b. The pictorial pattern *A* and its hierarchical structural descriptions

Illustration from K.S. Fu

A parse tree is a hierarchical decomposition.

Parse graph = parse tree + relations

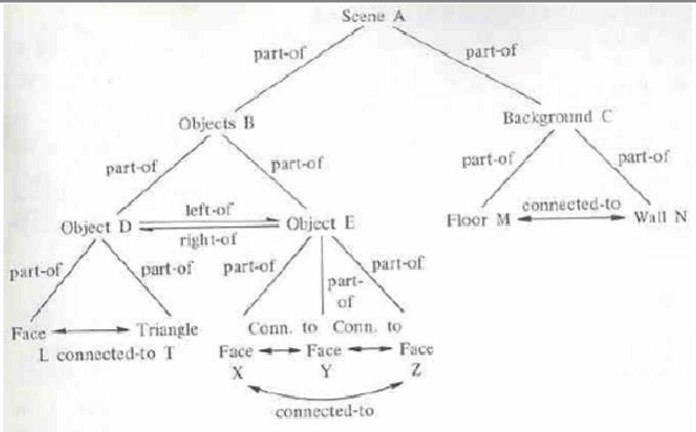


Fig. 1.2. A relational graph of scene *A*

Illustration from K.S.
From this parse graph, we can generate a narrative text description.

Stochastic grammars

A stochastic grammar is a 5-tuple $G=<V_N, V_T, R, p, \Sigma>$

V_N --- non-terminal nodes,
V_T --- terminal nodes,
R --- production rules,
Σ --- the language (a set of valid sentences)
p --- the probability

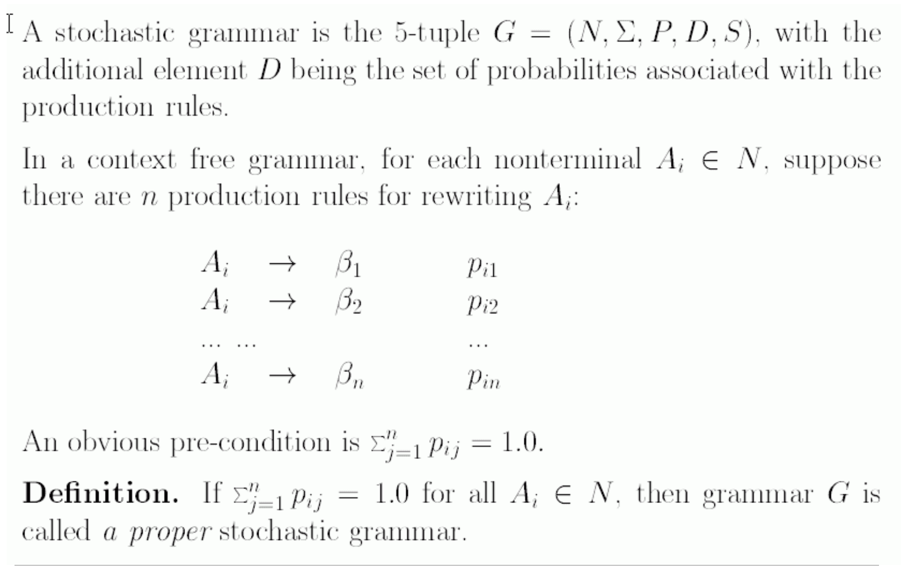
$$L(G) = \{ (w, p(w)) : \ s \xrightarrow{R*} w, w \in (V_N \cup V_T)^* \}$$

A sentence w may have multiple ways to parse, each is a series of production rules. Let's denote by the set of possible parses by

$$\Omega(w) = \{ ps_i = (r_{i,1}, r_{i,2}, ..., r_{i,n(i)}) : s \xrightarrow{r_{i,1} \cdot r_{i,2} \cdots r_{i,n(i)}} w, \ \ i = 1,2,...,m \}$$

$$p(w) = \sum_{ps_i \in \Omega(w)} p(r_{i,1}) \cdot p(r_{i,2}) \cdots p(r_{i,n(i)})$$

Stochastic grammars



A stochastic grammar is the 5-tuple $G = (N, \Sigma, P, D, S)$, with the additional element D being the set of probabilities associated with the production rules.

In a context free grammar, for each nonterminal $A_i \in N$, suppose there are n production rules for rewriting A_i:

$$\begin{array}{lllll} A_i & \rightarrow & \beta_1 & & p_{i1} \\ A_i & \rightarrow & \beta_2 & & p_{i2} \\ \ldots & \ldots & & & \ldots \\ A_i & \rightarrow & \beta_n & & p_{in} \end{array}$$

An obvious pre-condition is $\Sigma_{j=1}^{n} p_{ij} = 1.0$.

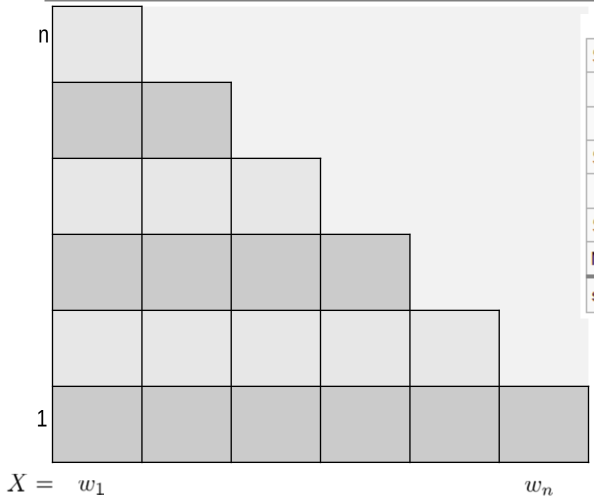
Definition. If $\Sigma_{j=1}^{n} p_{ij} = 1.0$ for all $A_i \in N$, then grammar G is called *a proper* stochastic grammar.

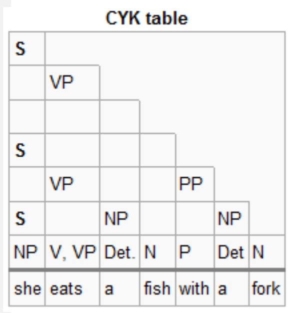
Typical parsing algorithms in the NLP literature

1, Pure bottom-up:	CYK – chart parsing, 1960s (Cocke and Schwartz 1967, Younger 1970, Kasami 1965)
2, Pure top-down:	Earley-parser, 1970s (Earley, Stockle)
3, Recursive/iterative:	Inside-outside algorithm, 1990s (Lori and Young)
4, Heuristic:	Best-first Chart Parsing, 2000s (Chaniak, Johnson, Klein, Manning, et al)

In vision, the algorithm really depend on the category!

CYK algorithm: an example




CYK table

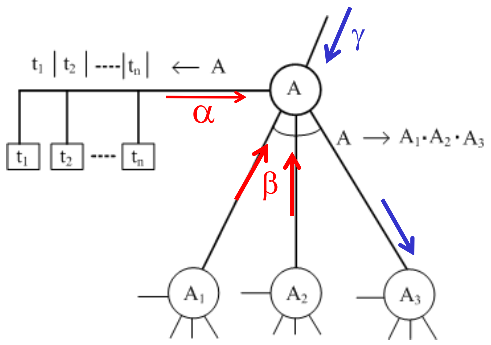
S						
	VP					
S						
	VP			PP		
S		NP			NP	
NP	V, VP	Det.	N	P	Det	N
she	eats	a	fish	with	a	fork

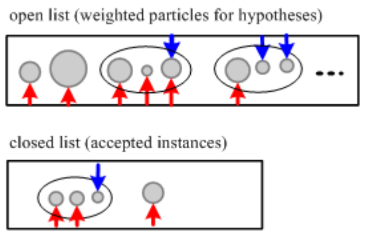
Defining the α, β and γ computing processes in AoG

The And-Or graph is a recursive structure. So, consider a node A.

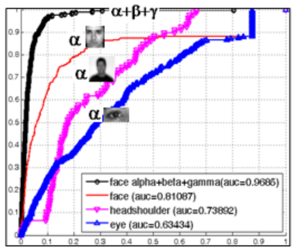
1, any node A terminate to leaf nodes at a coarse scale (ground).

2, any node A is connected to the root.

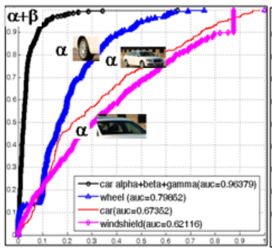

$t_1 | t_2 | ---- | t_n | \leftarrow A$
t_1
t_2
----
t_n
α
A
γ
β
$A \longrightarrow A_1 \cdot A_2 \cdot A_3$
A_1
A_2
A_3


open list (weighted particles for hypotheses)
closed list (accepted instances)

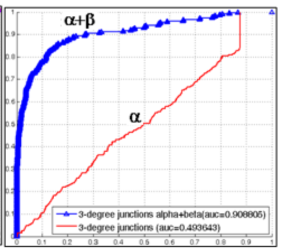
Numeric measures of the efficiency of the α, β and γ processes



(a) head/shoulder---face---eyes



(b) car---parts

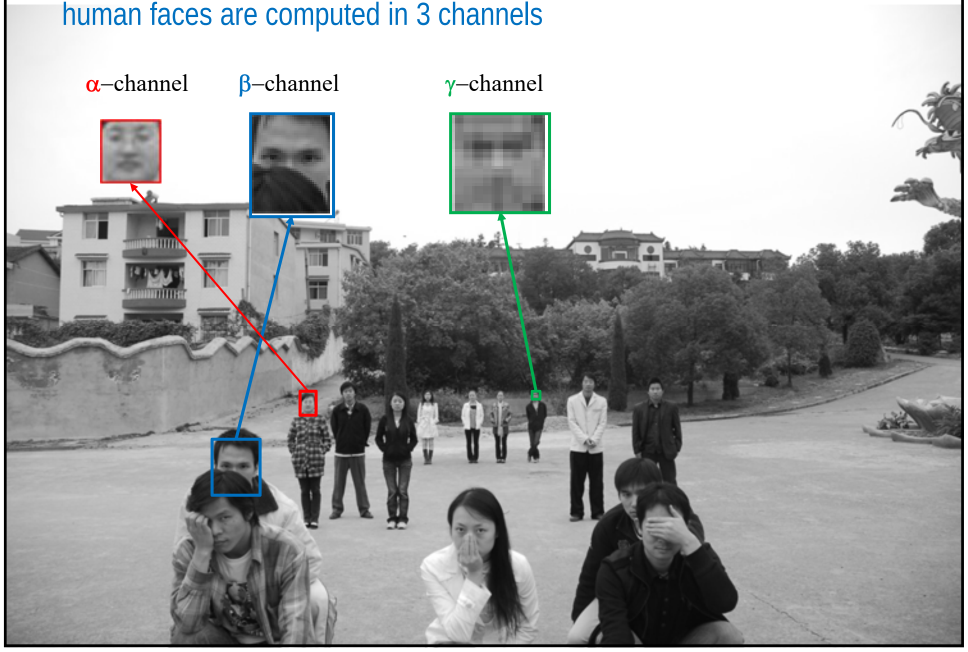


(c) junctions---linelets

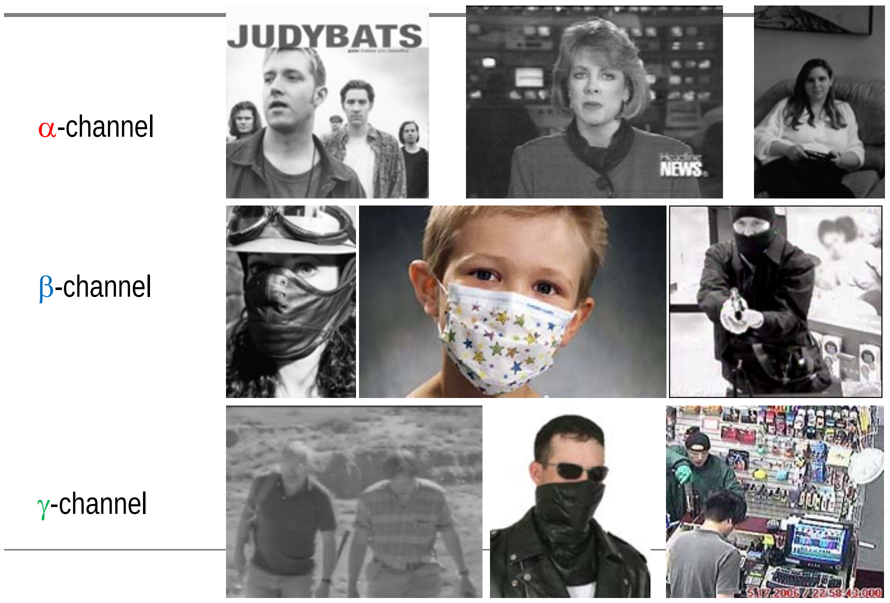
Ref: T.F. Wu and S.C. Zhu, "A Numeric Study of the Bottom-up and Top-down Inference Processes in And-Or Graphs," IJCV, 2011.

Case study 1: face detection

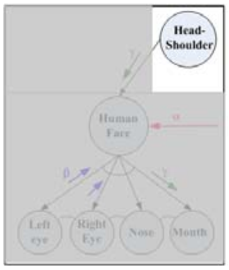
human faces are computed in 3 channels

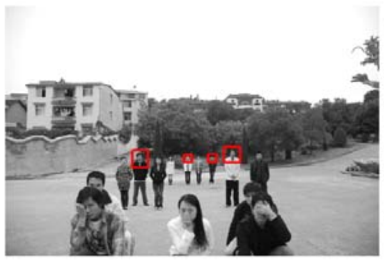


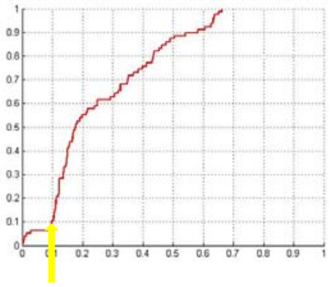
Human faces in real scenarios



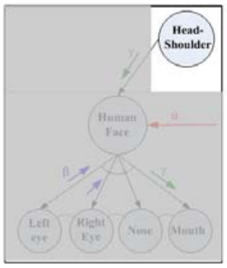
α–channel: head-shoulder

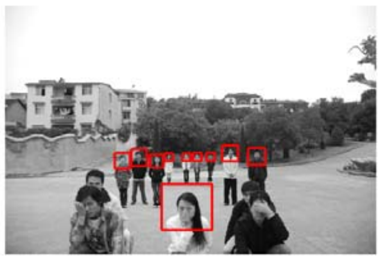


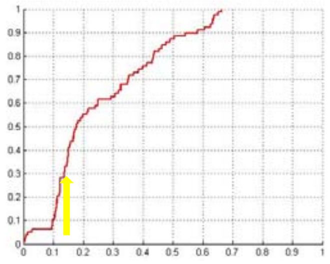




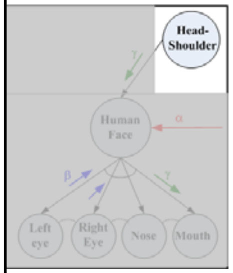
α–channel: head-shoulder

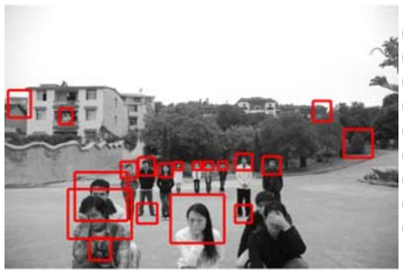


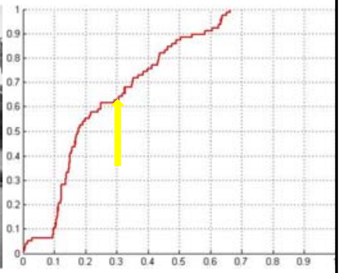


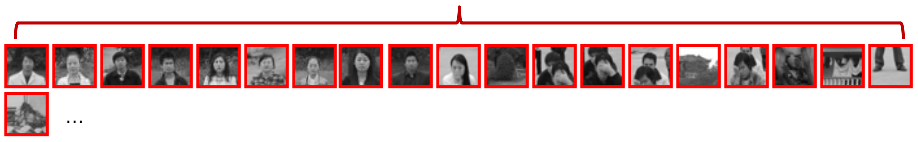


α–channel: head-shoulder

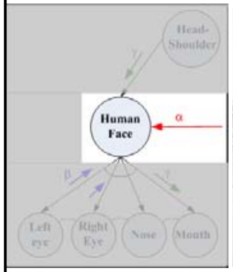


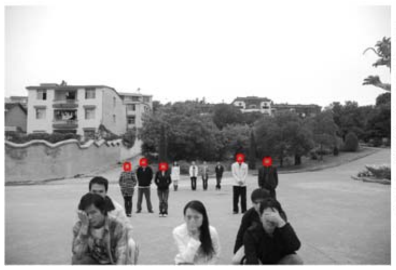


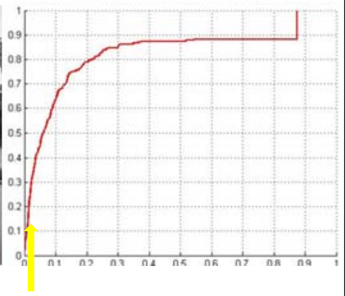


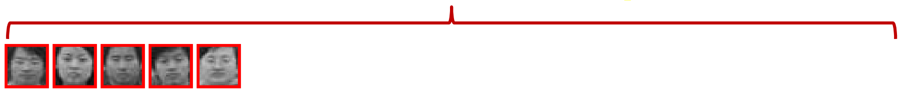

...

α–channel: face

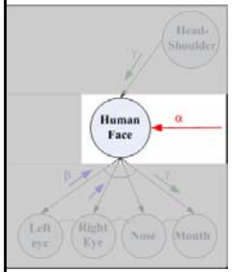


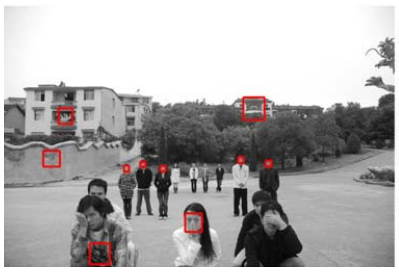


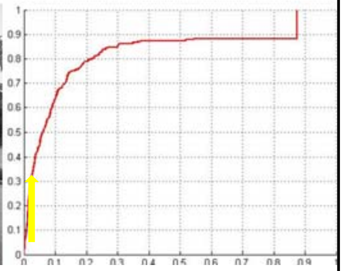


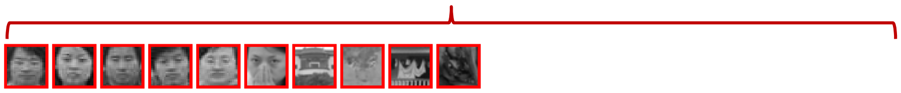


α–channel: face

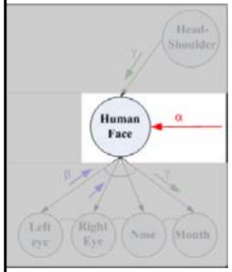


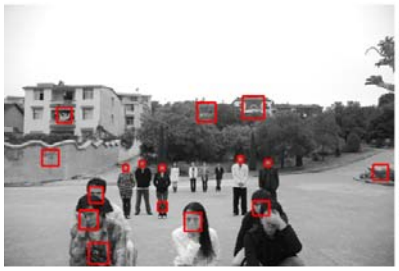


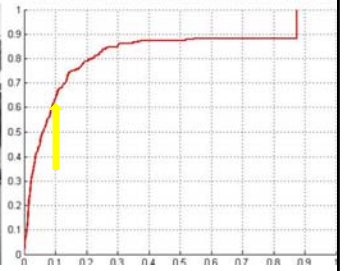


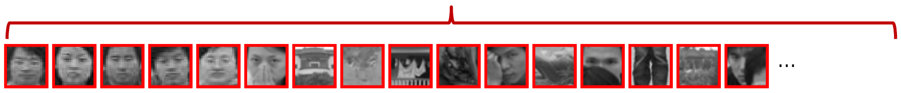


α–channel: face

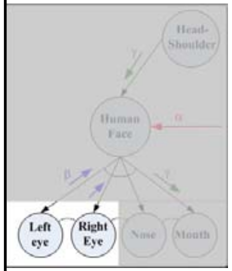


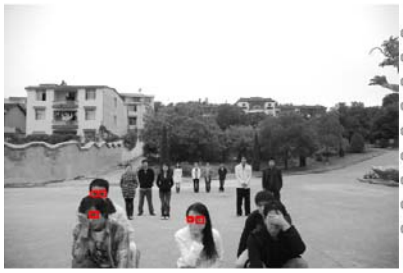


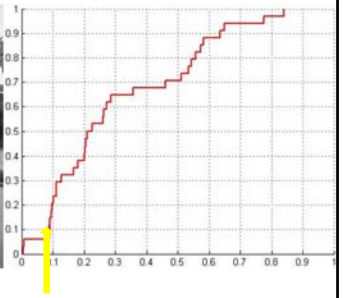


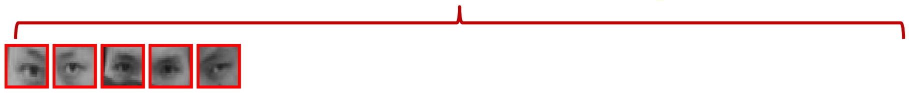


α–channel: eye

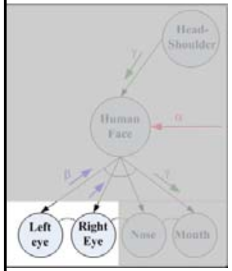


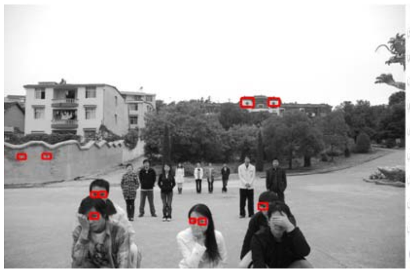


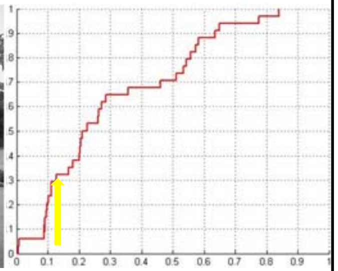


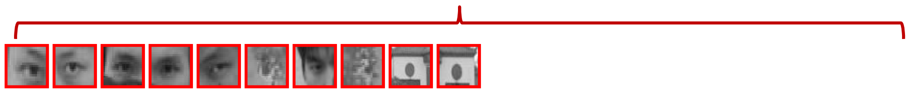


α–channel: eye

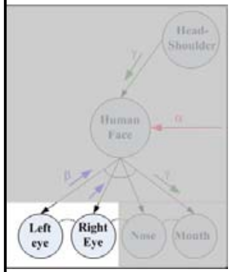


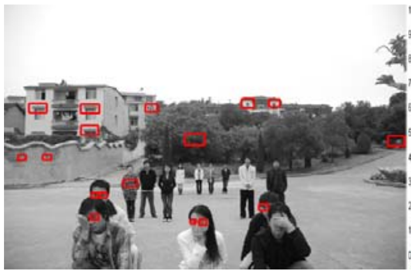


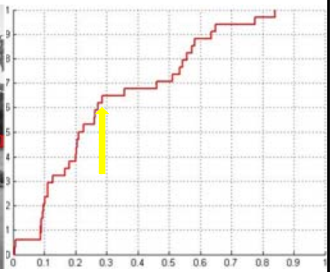


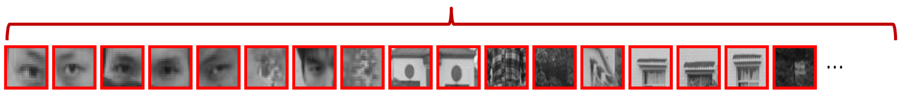


α–channel: eye

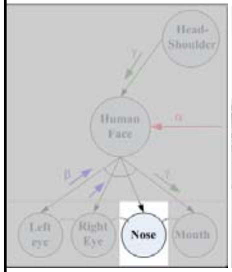


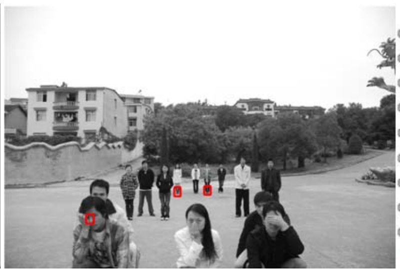


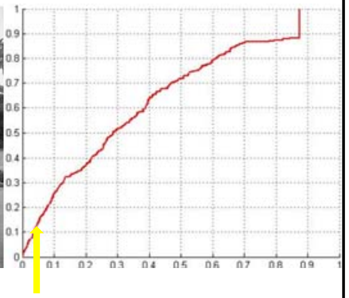


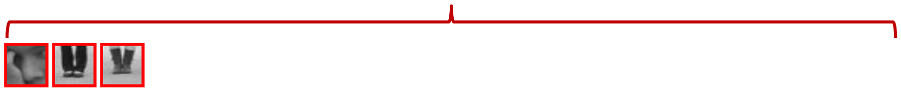


α–channel: nose

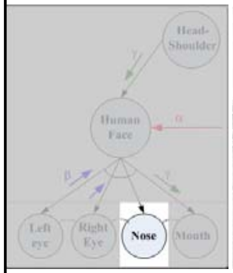


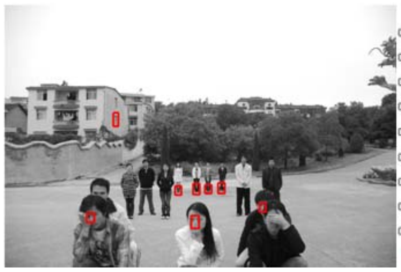


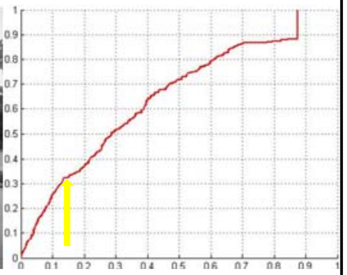


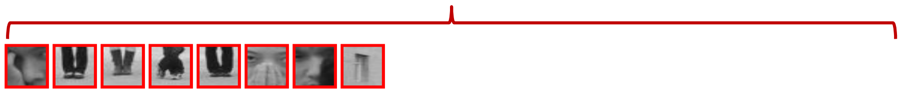


α–channel: nose

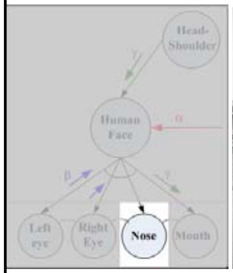


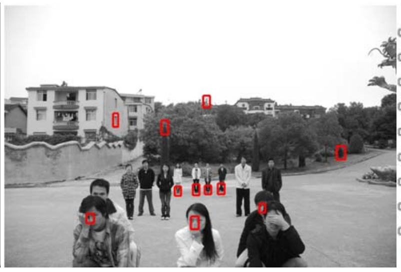


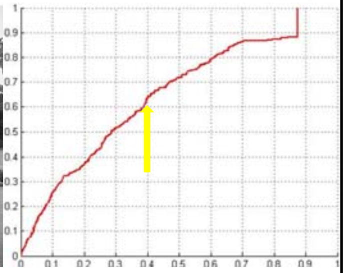


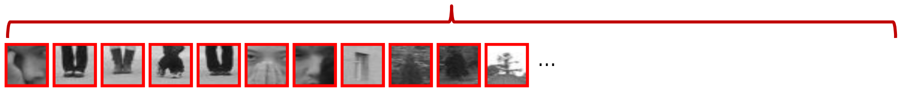


α–channel: nose



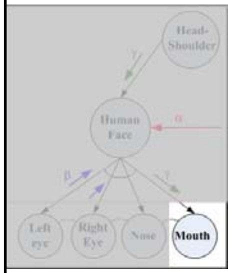


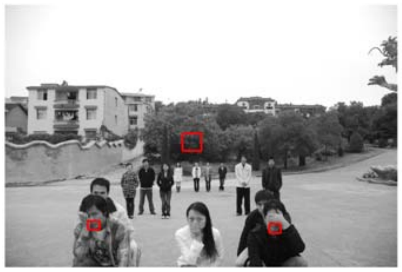


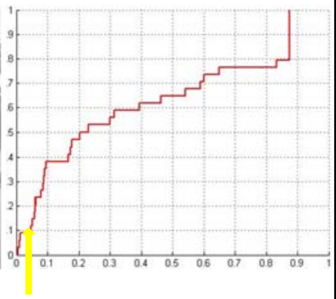


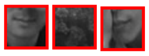
…

α–channel: mouth

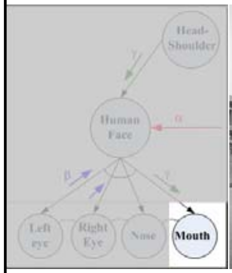


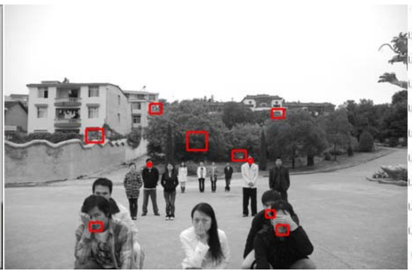


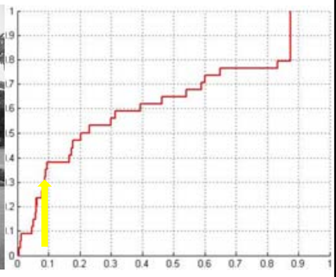


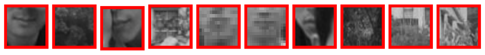


α–channel: mouth

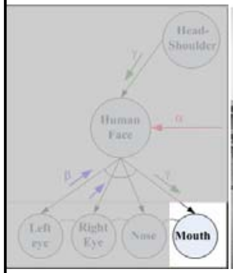


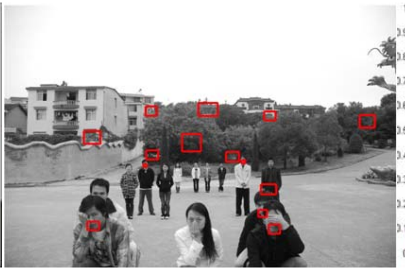


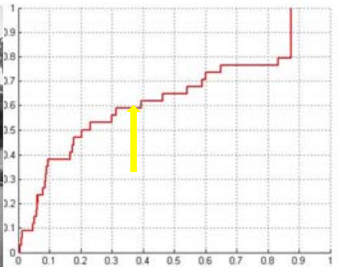


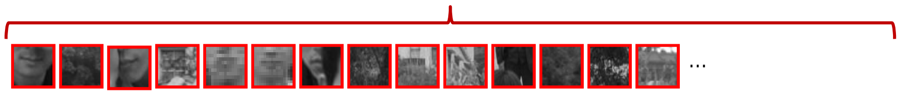


α–channel: mouth

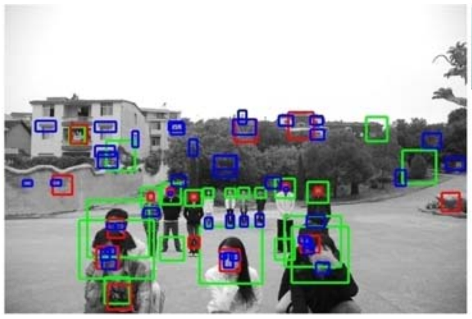






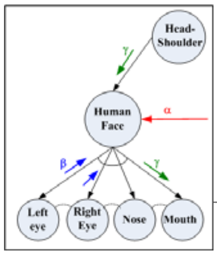


All α channels



keep things and throw away stuffs by

α β γ



Recursive decomposition of the Bayesian posterior probability

$$
\begin{aligned}
pg^* = \arg\max_{pg} \{ & \log p(A|O) + \sum_{i=1}^{2} \log p(X(C_i)|X(A)) \\
& + \underbrace{\log \frac{p(I_{\Lambda_{t_A}}|t_A)}{q(I_{\Lambda_{t_A}})}}_{\alpha(A)\ \text{process}} \\
& + \underbrace{[\underbrace{\sum_{i=1}^{2} \log \frac{p(I_{\Lambda_{t_{C_i}}}|t_{C_i})}{q(I_{\Lambda_{t_{C_i}}})}}_{\alpha(t_{C_i})\ \text{process}} + \underbrace{\log p(X(C_1), X(C_2))}_{\text{binding model}}]}_{\beta(A)\ \text{process}} \\
& + \underbrace{[\underbrace{\log \frac{p(I_{\Lambda_{t_P}}|t_P)}{q(I_{\Lambda_{t_P}})}}_{\alpha(P)\ \text{process}} + \underbrace{\log p(X(A)|X(P))}_{\text{prediction model}}]}_{\gamma(A)\ \text{process}} \}
\end{aligned}
$$

Recursive decomposition of the Bayesian posterior probability

Defining the weights for the candidate solutions

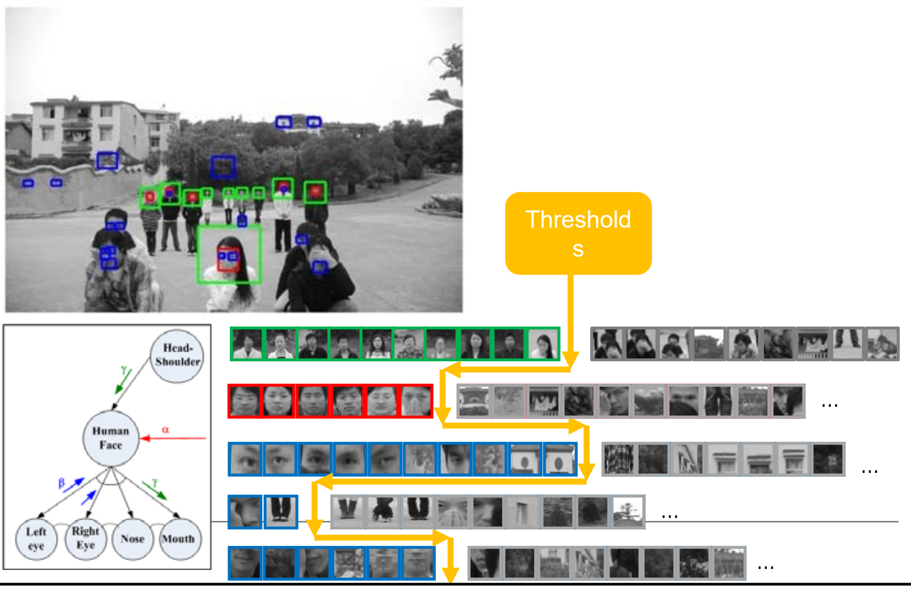
$$w_A^{\alpha} = \log \frac{p(I_{\Lambda_{t_A}}|t_A)}{q(I_{\Lambda_{t_A}})}$$

$$w_A^{\beta(\mathbf{c})} = \sum_{i=1}^{2} \log \frac{p(I_{\Lambda_{t_{C_i}}}|t_{C_i})}{q(I_{\Lambda_{t_{C_i}}})} + \log p(X(C_1), X(C_2))$$

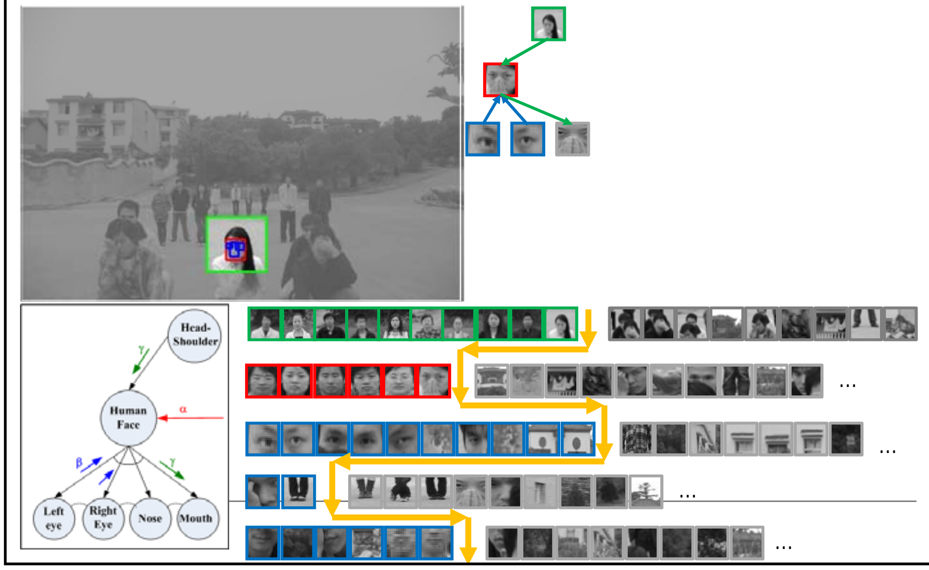
$$w_A^{\gamma(P)} = \log \frac{p(I_{\Lambda_{t_P}}|t_P)}{q(I_{\Lambda_{t_P}})} + \log p(X(A)|X(P))$$

$$
\begin{aligned}
pg^* = \arg \max_{pg \in \Omega_{pg}} \{ & \log p(A|O) + \sum_{i=1}^{2} \log p(X(C_i)|X(A)) \\
& + \underbrace{w_A^{\alpha}}_{\alpha(A)\ \text{process}} + \underbrace{w_A^{\beta(\mathbf{c})}}_{\beta(A)\ \text{process}} + \underbrace{w_A^{\gamma(P)}}_{\gamma(A)\ \text{process}} \}
\end{aligned}
$$

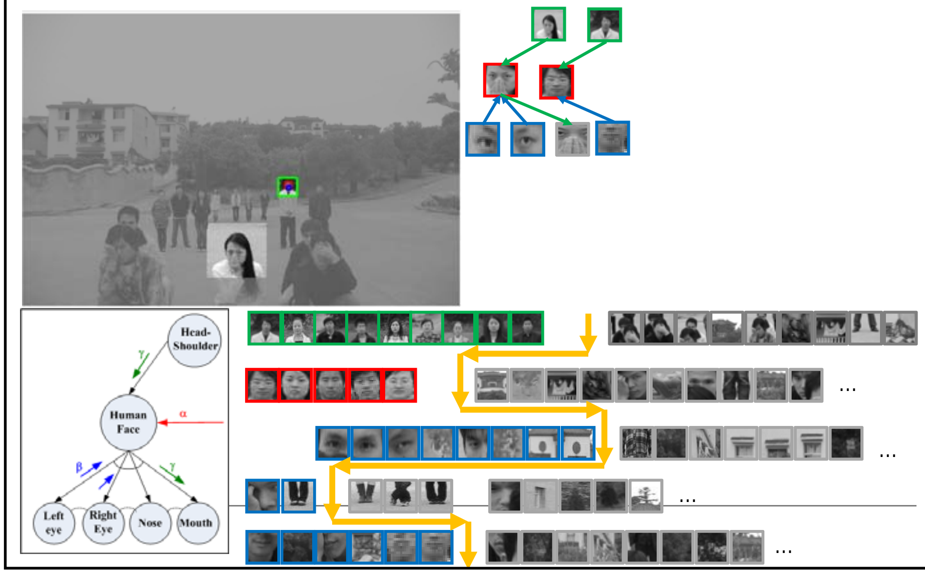
Best first chart parsing: Integrating α, β and γ channels



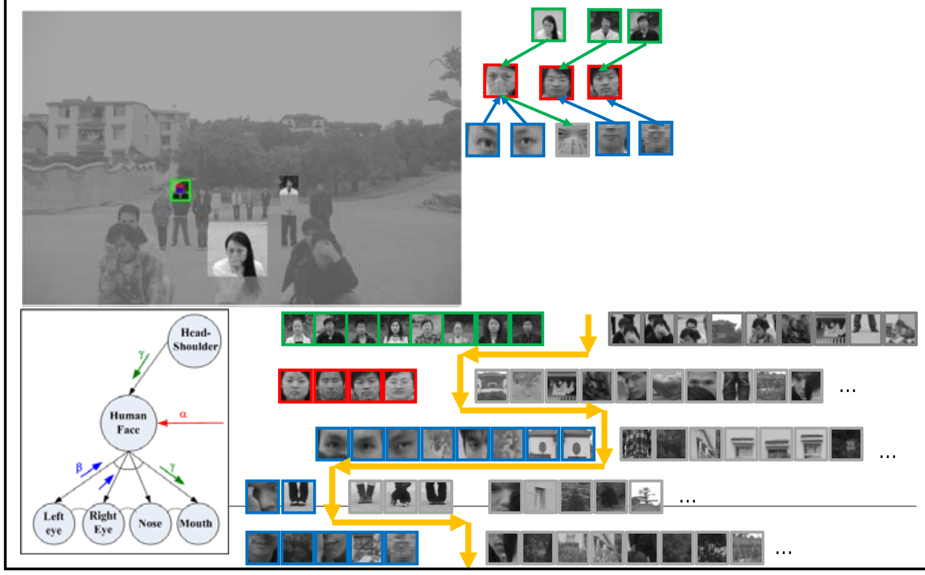
Integrating α, β and γ channels



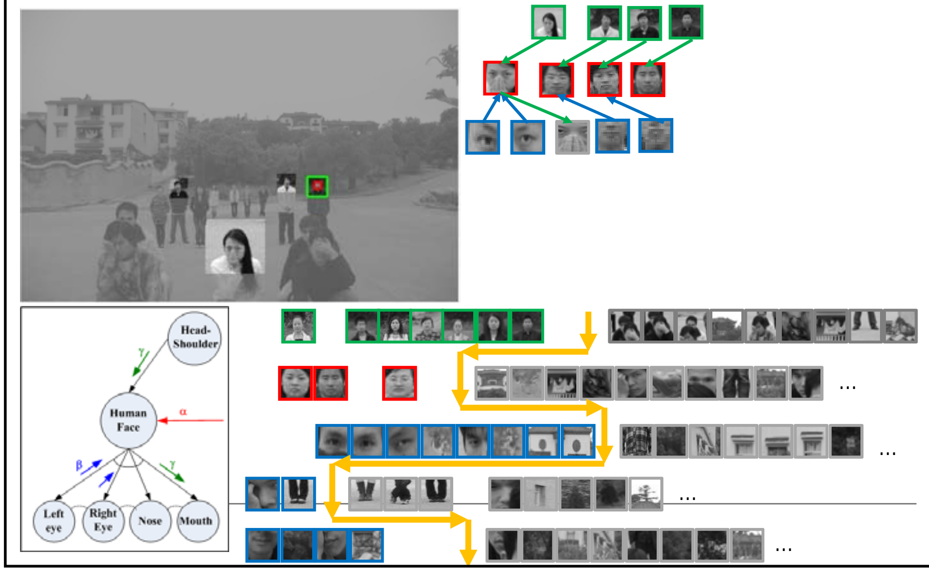
Integrating α, β and γ channels



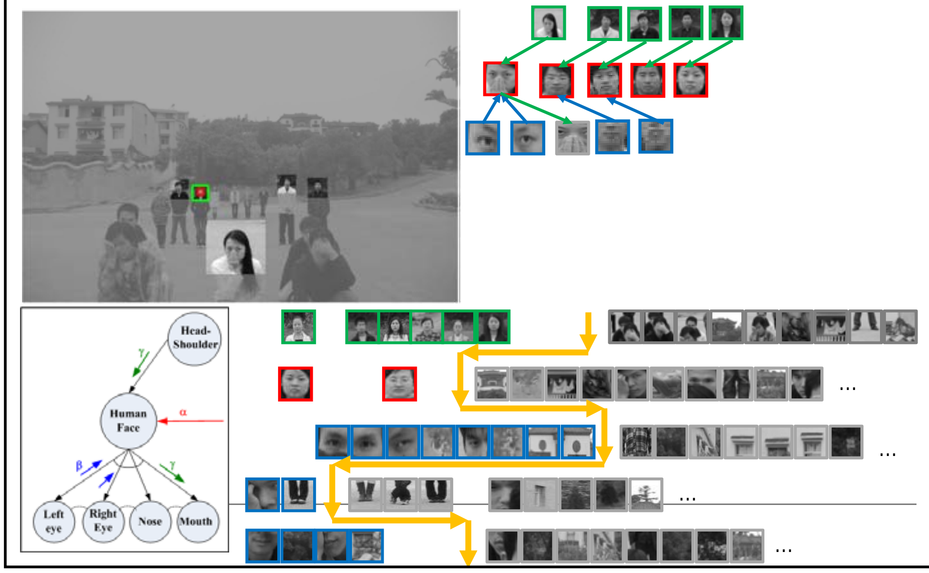
Integrating α, β and γ channels



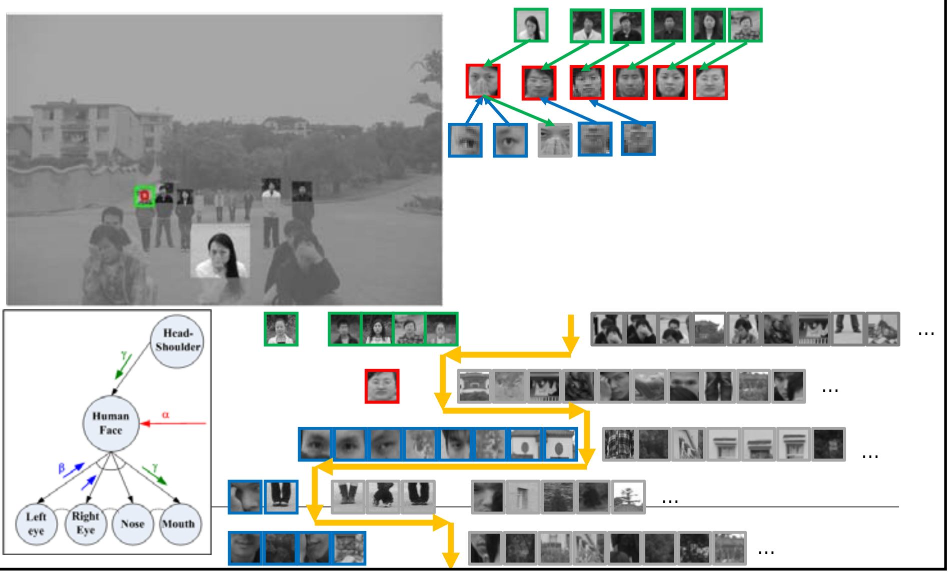
Integrating α, β and γ channels



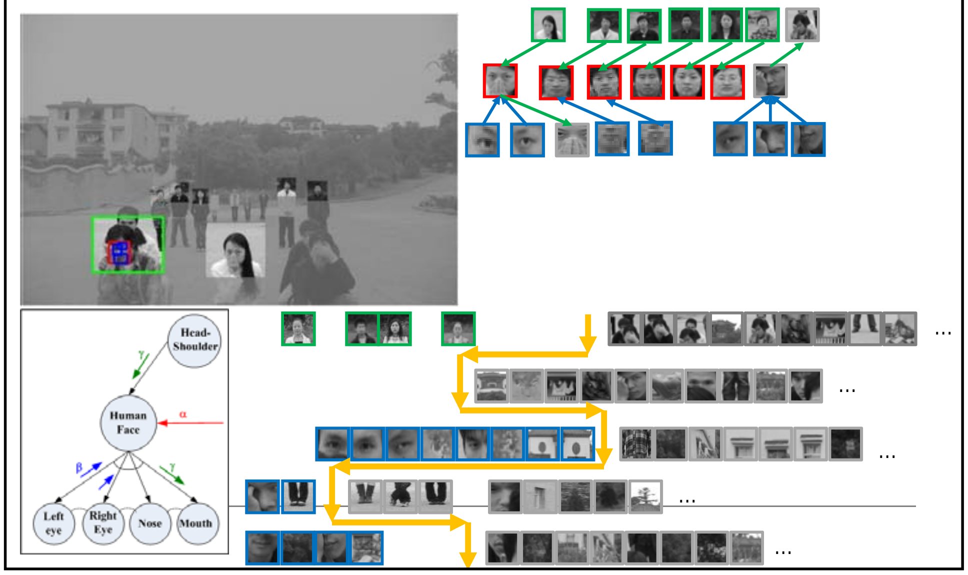
Integrating α, β and γ channels



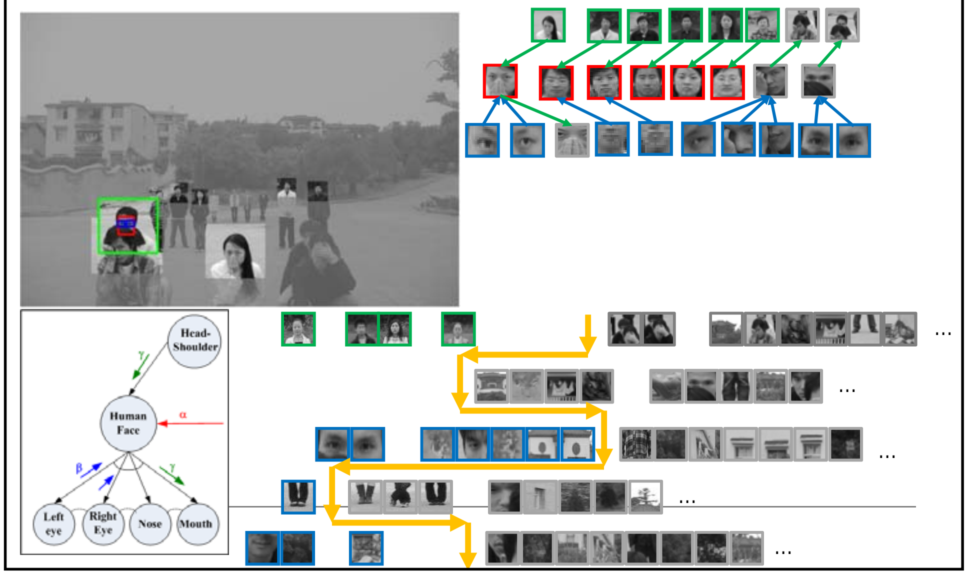
Integrating α, β and γ channels



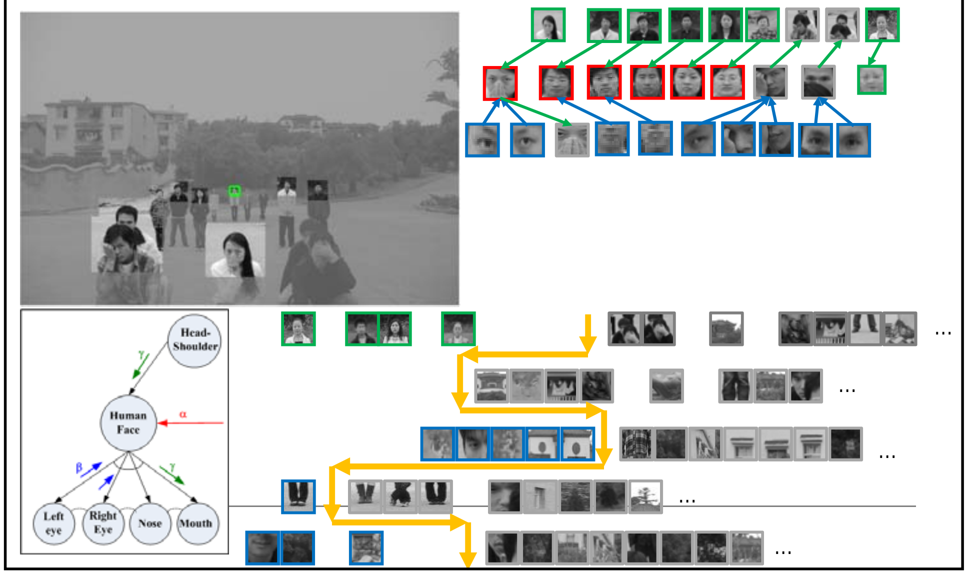
Integrating α, β and γ channels



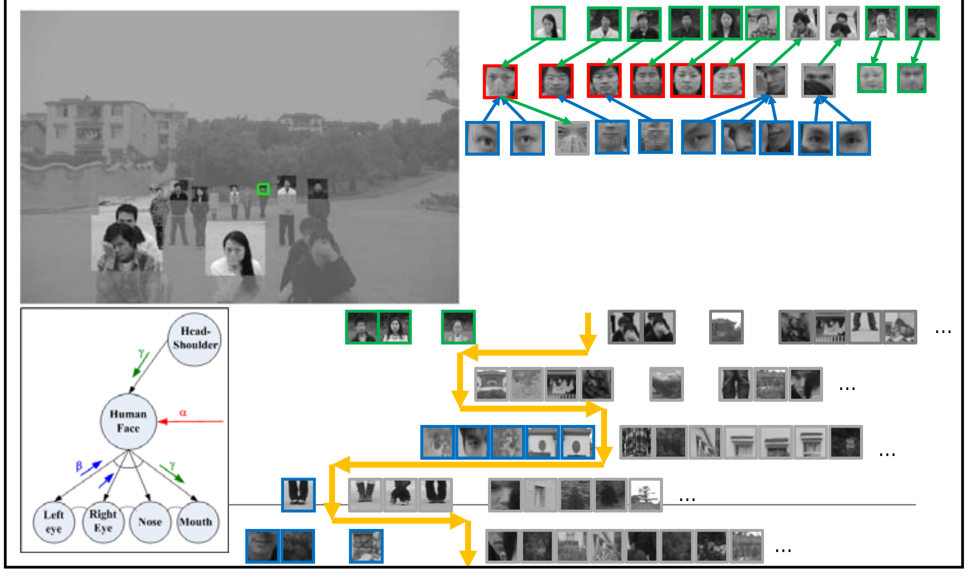
Integrating α, β and γ channels



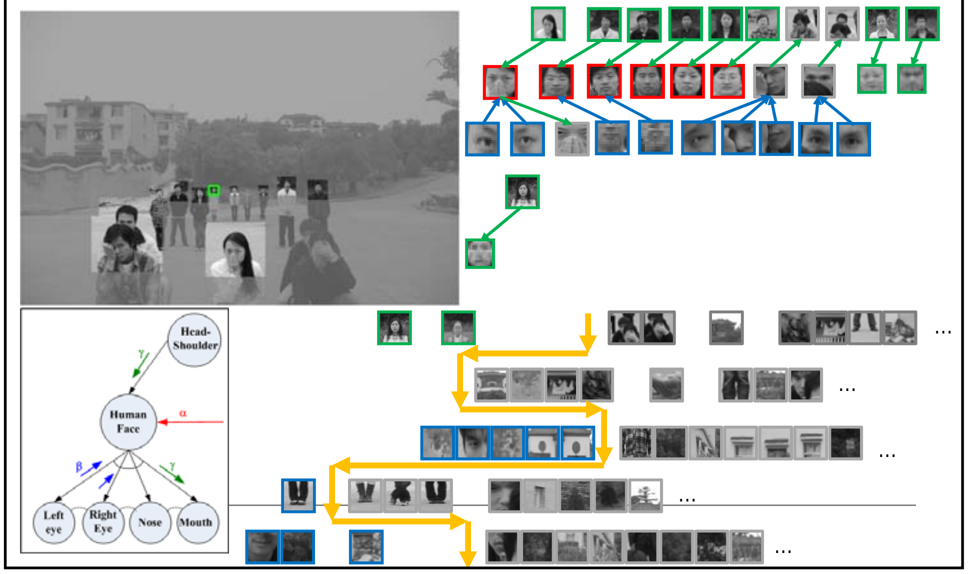
Integrating α, β and γ channels



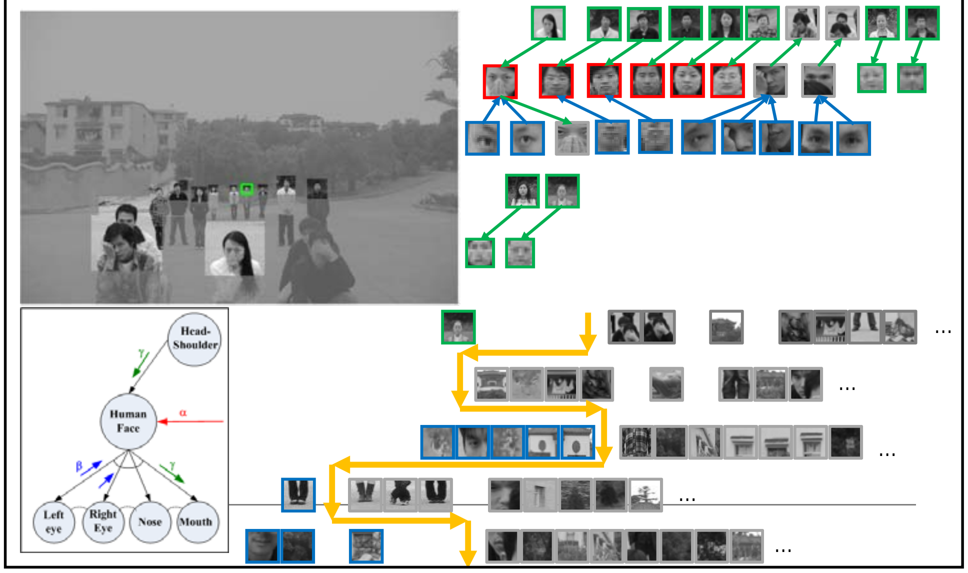
Integrating α, β and γ channels



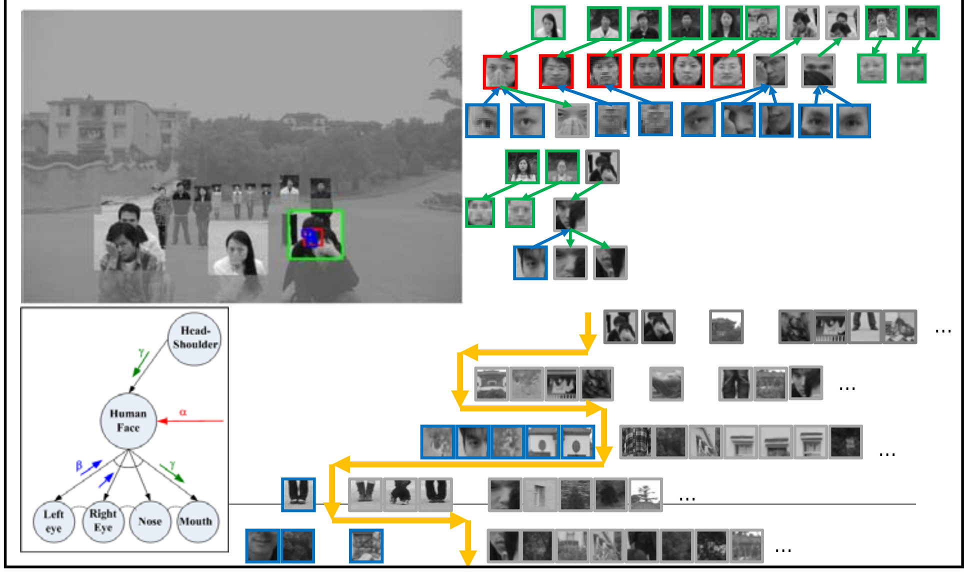
Integrating α, β and γ channels



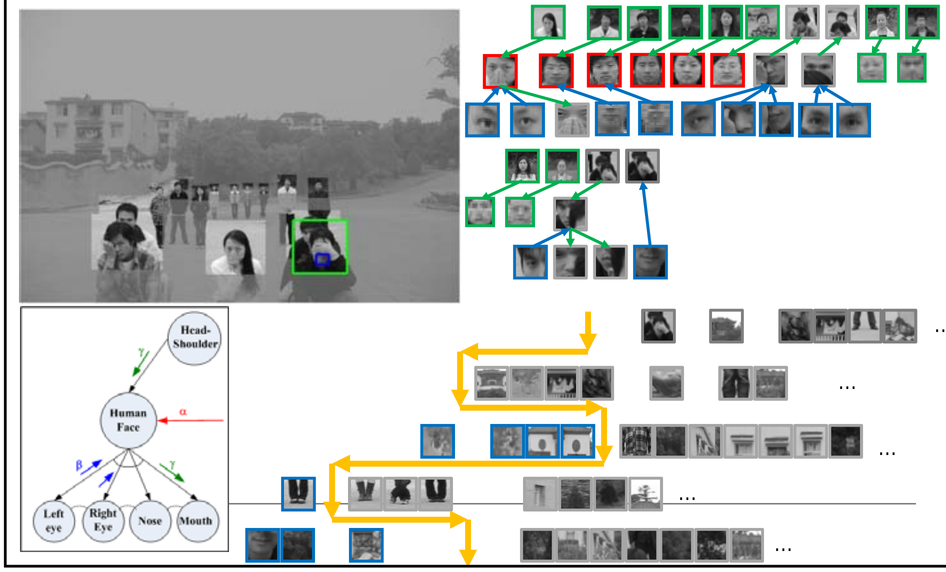
Integrating α, β and γ channels



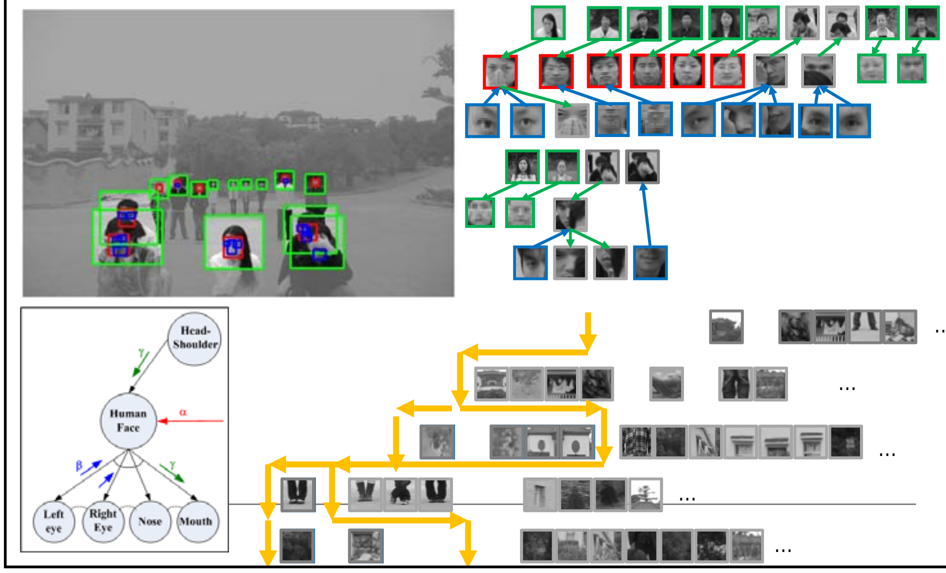
Integrating α, β and γ channels



Integrating α, β and γ channels

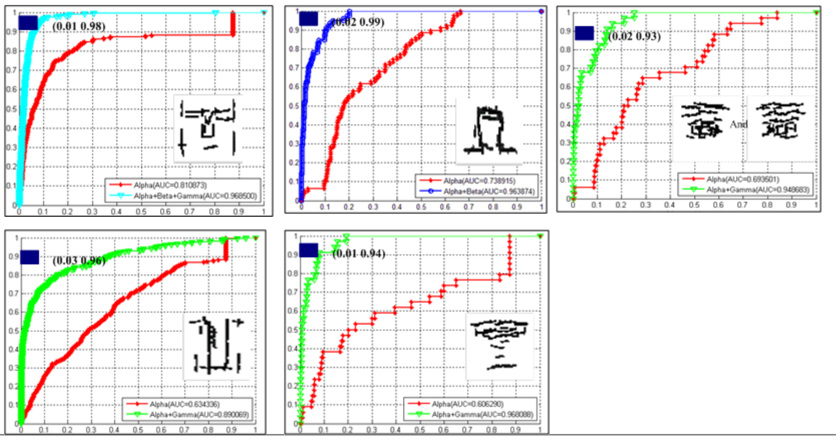


Integrating α, β and γ channels



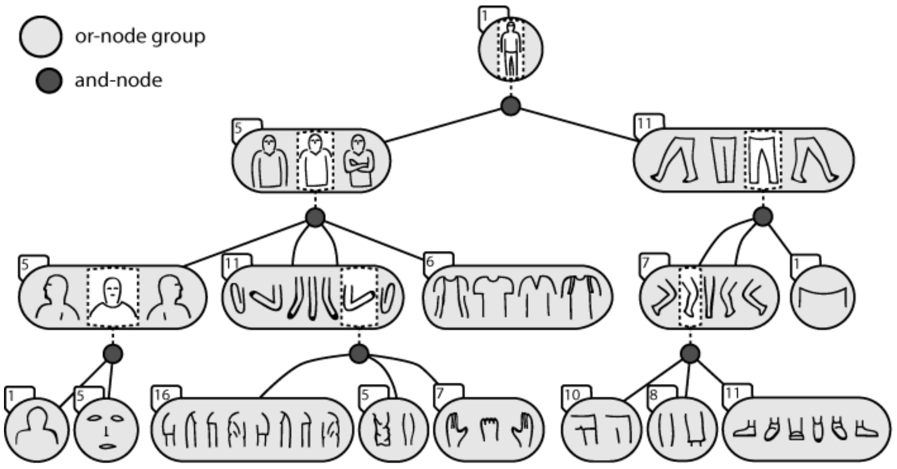
Performance comparison

red for α, blue for $\alpha+\beta$, green for $\alpha+\gamma$, cyan for $\alpha+\beta+\gamma$ channels

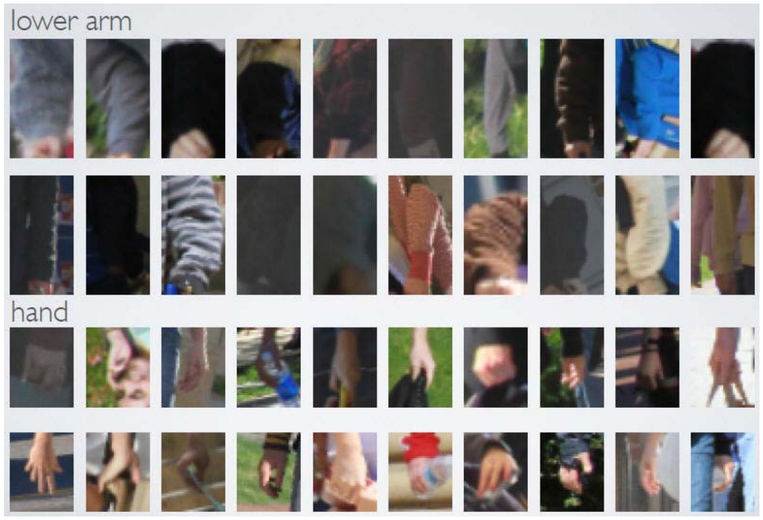


Ref: T.F. Wu and S.C. Zhu, "A Numeric Study of the Bottom-up and Top-down Inference Processes in And-Or Graphs," IJCV, 2011.

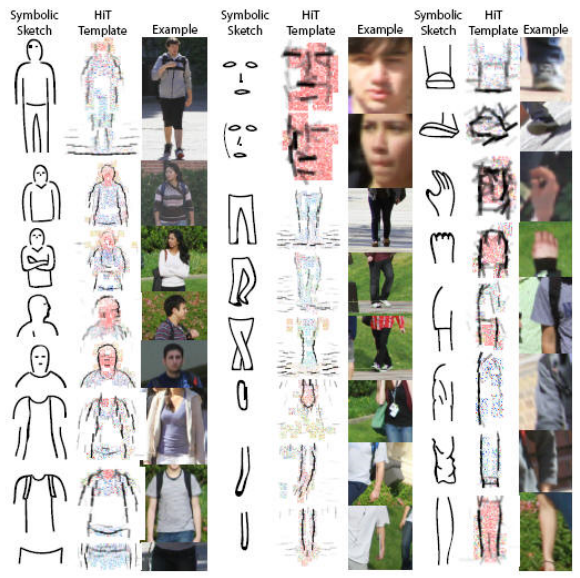
Case 2: parsing human figure



Problem: large variations in appearance and geometry

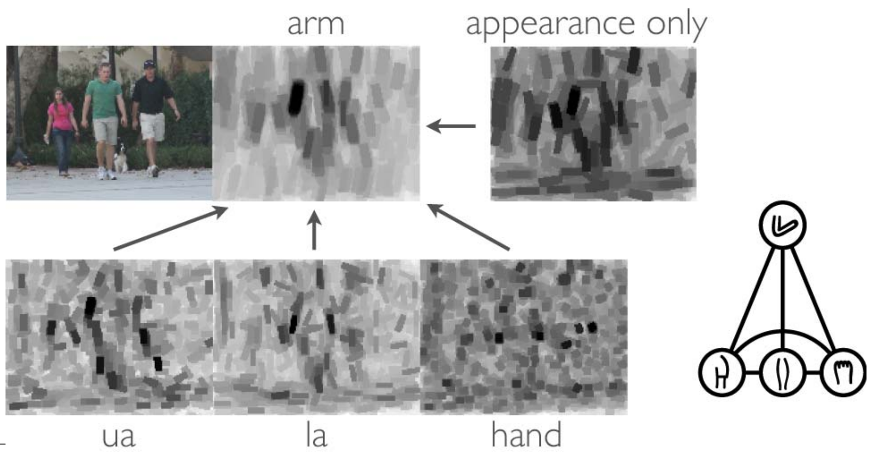


~120 Terminal nodes grounding the symbols

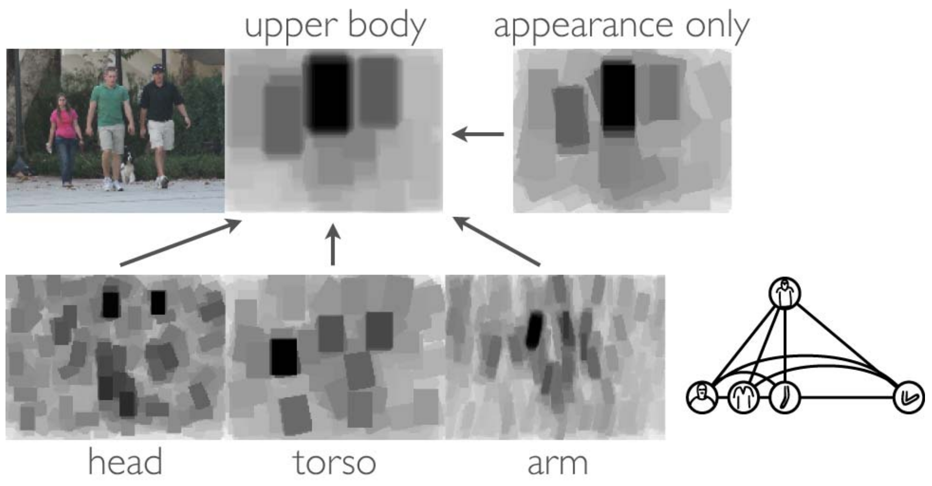


Local computation is hugely ambiguous

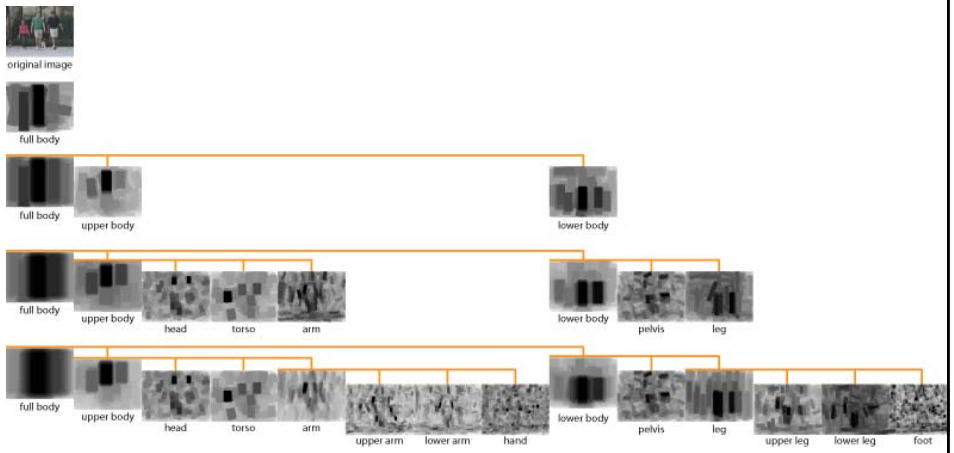
Dynamic programming and re-ranking



Composing Upper Body

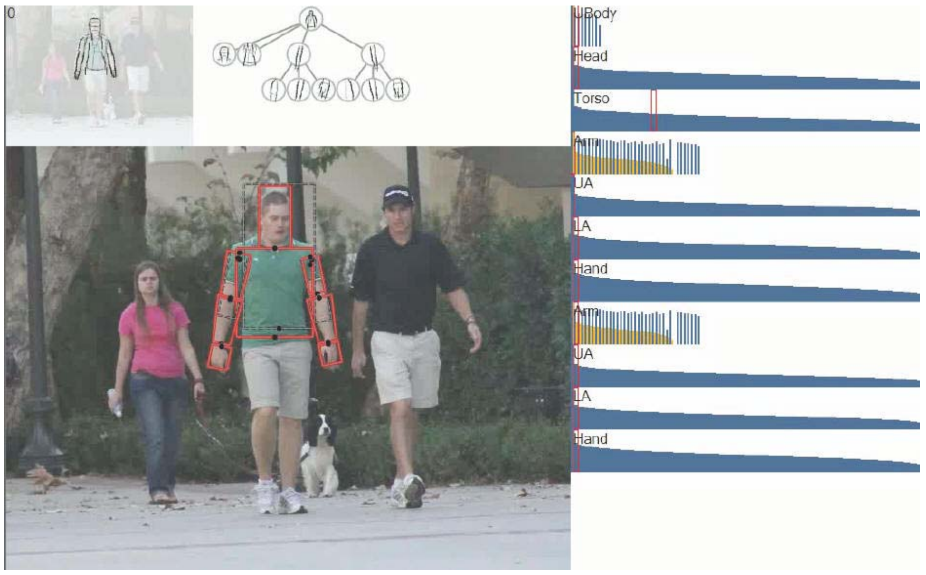


Composing parts in the hierarchy

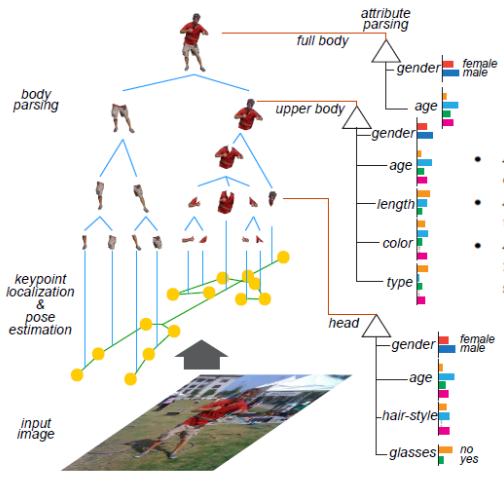


Ref: Brandon Rothrock "*Stochastic Image Grammars for Human Pose Estimation*," UCLS ph.D Dissertation, 2012.

Top-down / bottom-up inference



Joint parsing of pose, parts and attributes

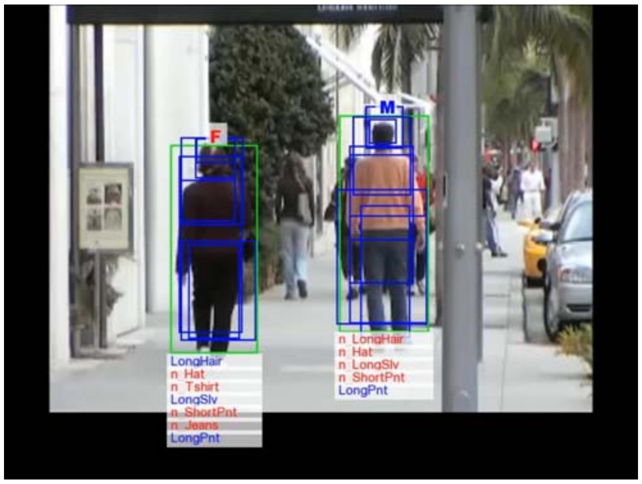


This parse graph integrates 3 grammars

- *A phrase structure grammar* representing the hierarchical decomposition from whole to parts;
- *A dependency grammar* modeling the geometric articulation by a kinematic graph of the body pose; and
- *An Attribute grammar* accounting for the compatibility relations between different parts in the hierarchy so that their appearances follow a consistent style.

Figure by Seyoung Park, UCLA.

Joint parsing of pose, parts and attributes


F
M
LongHair
n Hat
n Tshirt
LongSlv
n ShortPnt
n Jeans
LongPnt
n LongHair
n Hat
n LongSlv
n ShortPnt
LongPnt