Pictorial Structures for Object Recognition
P. FELZENSZWALB et al.

Anissa El Idrissi Feissel, Celio Boulay, Myriam Lebatteux

February 3, 2025

Overview

Introduction

- Part based modeling and recognition
- Using statistical models and energy minimization

Figure: Examples of object recognition

Pictorial Structures

Pictorial structure model

A collection of parts with connections between certain pairs of parts.

This is a general framework.

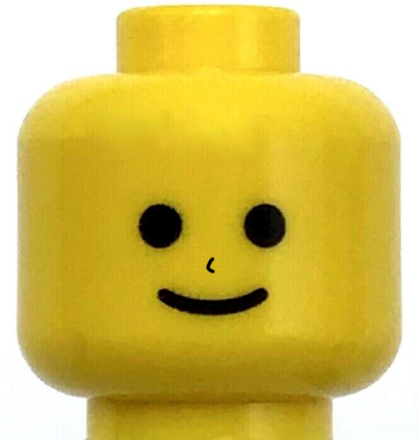
Graph representation

G=(V,E)
$V = \{v_1, ..., v_n\}$: the parts
Edges $(v_i, v_j) \in E$: connections between parts

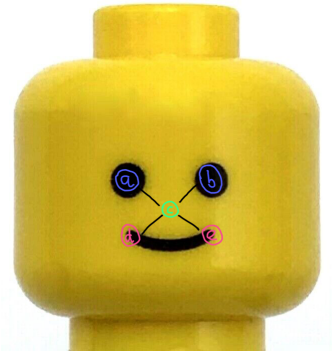
Target

We are looking for $L = (l_1, ..., l_n)$ the locations of parts $(v_i)_{i \in [1,n]}$

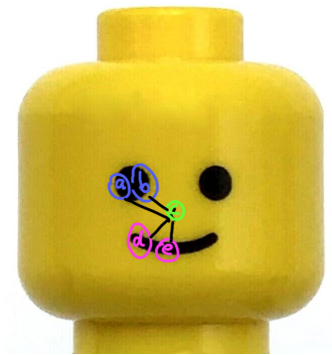
Example



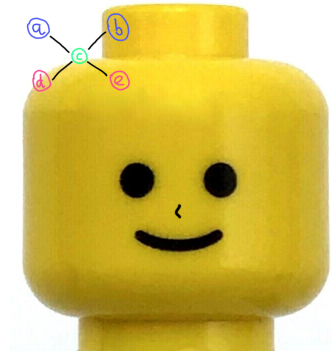
Example (cont.)



Objective



Bad structure



Bad matching

Energy minimization

$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right). \quad (1)$$

Parameters

L^* is the optimal configuration
$m_i(l_i)$: degree of mismatch
$d_{ij}(l_i, l_j)$: degree of deformation

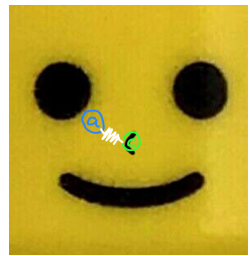
An overall decision is made base on match costs and deformation costs.
The objective is to optimize the algorithm to run in linear time.

Energy minimization (cont.)

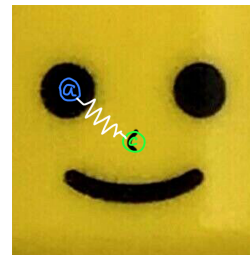
$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right). \quad (1)$$

Good matching

High missmatch



Spring is too compressed



Good relative position of the vertices

Objectives

- Previous methods don't find optimal solutions.
- In addition they require good initialization.
- We aim to have a linear running time in the number of possible locations for each part.

Hypotheses

1. Acyclic graph, allowing a running time in $O(h^2n)$

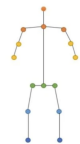


Figure: The human body seen as a acyclic graph

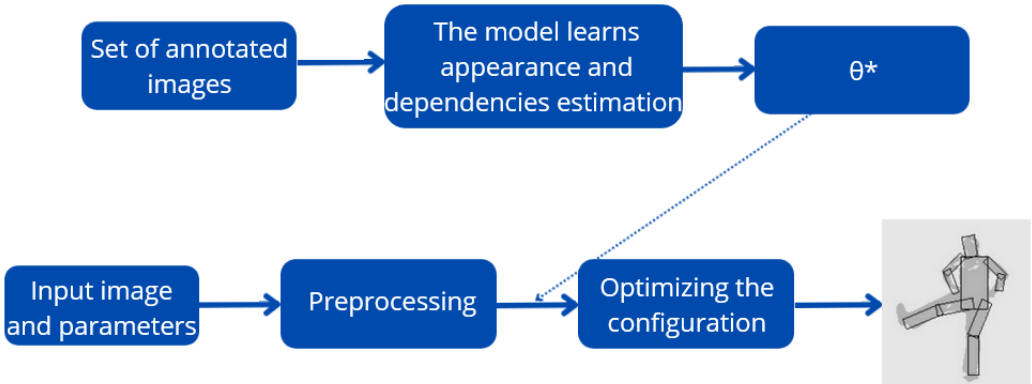
2. Mahalanobis distance, allowing a running time in $O(h'n)$

Mahalanobis Distance

$$d_{ij}(l_i, l_j) = (T_{ij}(l_i) - T_{ji}(l_j))^T M_{ij}^{-1} (T_{ij}(l_i) - T_{ji}(l_j))$$

$T_{ij}(l_i)$ represents the set of possible transformed locations as positions in a grid
M_{ij}^{-1} measures the deformation of the spring connecting v_i and v_j

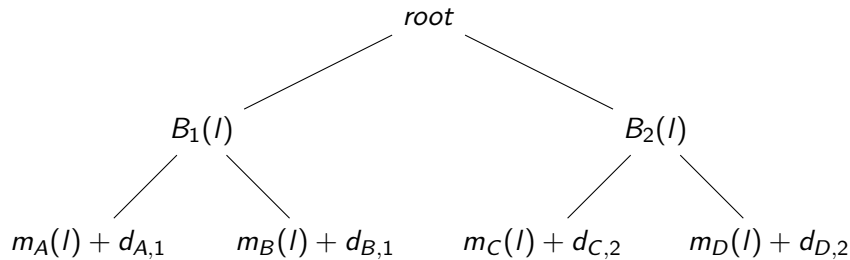
Pipeline


Set of annotated images
The model learns appearance and dependencies estimation
θ*
Input image and parameters
Preprocessing
Optimizing the configuration

Algorithm: Forward Pass - Cost Propagation

Computation of $B_j(l_i)$:

- Start from the **leaf nodes** and propagate optimal costs upwards.



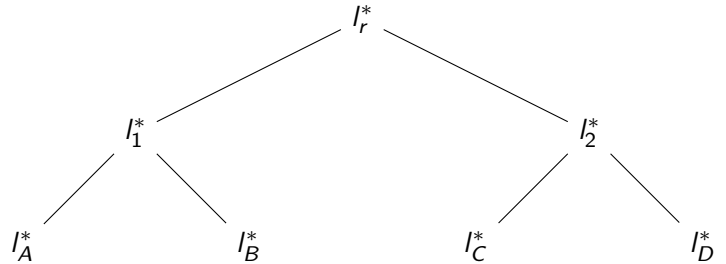
The dynamic programming update equation:

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) + \sum_{v_c \in C_j} B_c(l_j) \right). \tag{2}$$

Algorithm: Backward Pass - Solution Reconstruction

Computation of l_j^* and optimal assignment:

- Start from the **root** and reconstruct the optimal configuration by propagating downward.



The optimal assignment equation:

$$l_j^* = \arg\min_{l_j} \left(m_j(l_j) + d_{pj}(l_p^*, l_j) + \sum_{v_c \in C_j} B_c(l_j) \right). \tag{3}$$

Algorithm: Efficient Cost Computation

Given a grid $(\mathcal{G}, \rho(x, y))$:

- The distance transform D_f is computed as:

$$D_f(x) = \min_{y \in G} \left(\rho(x, y) + f(y) \right). \tag{4}$$

- Given this, one can compute B_j as follow :

$$B_j(l_i) = D_f(T_{ij}(l_i)). \tag{5}$$

Learning: Parametrization

Set of parameters

$\theta = (u, E, c)$

- $u = \{u_1, ..., u_n\}$ the appearance parameters
- E is the set of edges
- $c = \{c_{ij} \mid (v_i, v_j) \in E\}$ the connection parameters

Posterior distribution

$p(L|I, \theta)$ is what we are interested in.

The Bayes' rule gives: $p(L|I, \theta) = p(I|L, \theta) * p(L|\theta)$

Learning: Model Parameters

Ground truth

We wish to find the perfect parameters using:

- $\{I_1, ..., I_n\}$ a set of images
- $\{L_1, ..., L_n\}$ the corresponding good configurations

$$\theta^* = \arg \max_{\theta=(u,E,c)} \underbrace{\prod_{k=1}^{m} p(I^k \mid L^k, u)}_{\text{Appearance}} \underbrace{\prod_{k=1}^{m} p(L^k \mid E, c)}_{\text{Dependencies}}. \tag{6}$$

Learning: Splitting the parameters

One can compute the parameters independently as follow:

$$u_i^* = \arg\max_{u_i} \prod_{k=1}^{m} p(l_k | l_k^i, u_i). \tag{7}$$

$$c_{ij}^* = \arg\max_{c_{ij}} \prod_{k=1}^{m} p(l_i^k, l_j^k | c_{ij}). \tag{8}$$

Now that we have the position of each vertex in the graph, we still need its edges.
We introduce the quality:

$$q(v_i, v_j) = \prod_{k=1}^{m} p(l_i^k, l_j^k \mid c_{ij}^*). \tag{9}$$

E^* should form a tree $\rightarrow$ we can solve a MST (Kruskal).

Application examples: Iconic models

- A part is here defined by its position (x,y) in the image
- Model based on the response of Gaussian derivative filters (different orders, orientations and scales)
- Creation of an iconic index for every position
- Connection between vi and vj is defined by $u_i = (s_{ij}, \Sigma_{ij})$ following a gaussian law

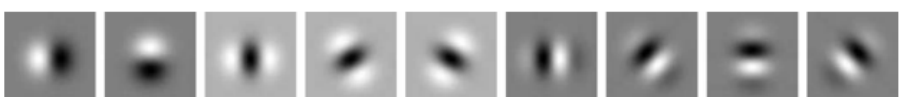


Figure: Gaussian derivative basis functions used in the iconic presentation

Experiments of iconic models

Figure: Matching results on an image with several faces

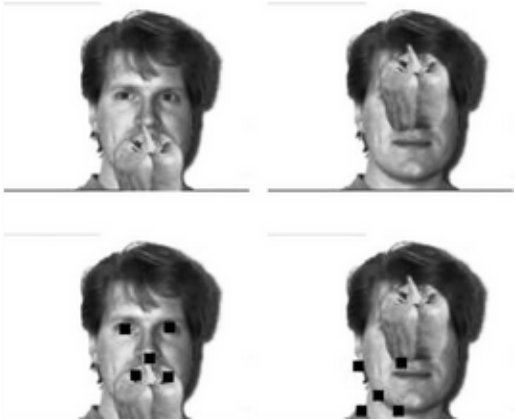


Figure: Matching results on occluded faces

Application examples: Articulated models (from binary images)

Each rectangle is defined by parameters:

- (x, y) the coordinates of its center
- $s \in [0, 1]$ its foreshortening
- θ its orientation

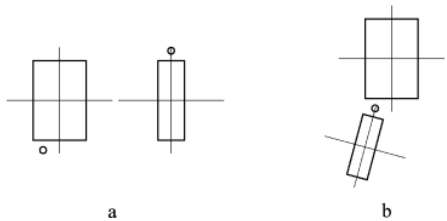


Figure: Two parts of an articulated object, (a) in their own coordinate system, (b) in the ideal configuration

Experiments of articulated models

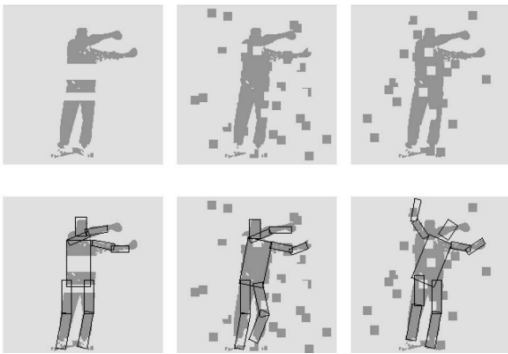


Figure: Matching results on corrupted images

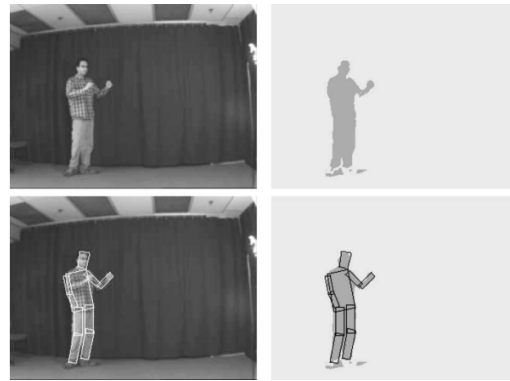


Figure: Example of a lack of information from the binary image

Personal insights: Implentation ideas

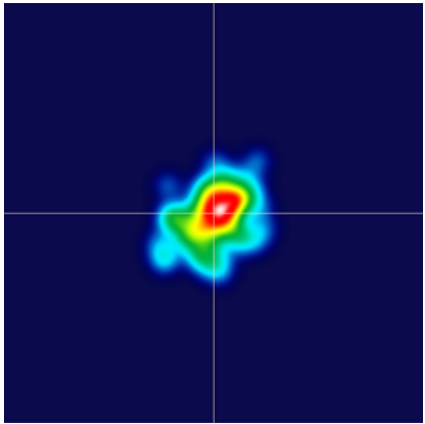


Figure: Probabilities heatmap

Figure: Background substraction

Personal insights: Idea of other data / application

- Detection of the vertebrae on a human spine.

Figure: Scan of a human spine (SFCR)

Thank you for your attention.