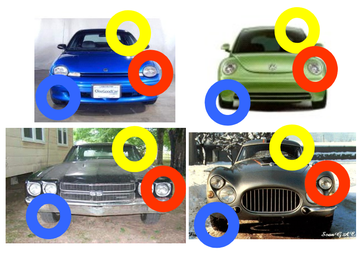
ALGORITMOS APROXIMADOS Y EXACTOS PARA ESTRUCTURAS PICTÓRICAS

Sebastián Ubalde subalde@dc.uba.ar
Norberto A. Goussies ngoussie@dc.uba.ar
Marta E. Mejail marta@dc.uba.ar

Departamento de Computación – Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Introducción

Trabajamos sobre la idea de representar un objeto como un conjunto de partes organizadas en una configuración deformable. Este tipo de modelo recibe el nombre de ***estructura pictórica***.



Modelo

Una estructua pictórica es un **grafo** $G = (V, E)$ donde los vértices V corresponden a las partes y los ejes E corresponden a las conexiones entre partes.

Una instancia de objeto se especifica mediante una **configuración** $L = (l_1, ..., l_2)$ dónde l_i especifica la ubicación de v_i para cada $v_i \in V$.

Dada una imagen I , buscamos un L que ubique satisfactoriamente el objeto modelado en la misma.

Hacemos un **planteo probabilístico** del problema:

$$p(L|I, \theta) \propto p(I|L, \theta)p(L|\theta)$$

Donde $\theta = (u, E, c)$ son los parámetros del modelo.

Además, modelamos:

$$p(I|L, \theta) = p(I|L, u) \propto \prod_{i=1}^{n} p(I|l_i, u_i)$$

$$p(L|\theta) = p(L|E, c) = \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij})$$

Obteniendo, por lo tanto:

$$p(L|I, \theta) \propto \left(\prod_{i=1}^{n} p(I|l_i, u_i) \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij}) \right)$$

Aprendizaje del Modelo

Dadas las imágenes $\{I^1, \ldots, I^m\}$ y sus correspondientes configuraciones $\{L^1, \ldots, L^m\}$. Buscamos los parámetros del modelo $\theta = (u, E, c)$. Usando **Máxima Verosimilitud** tenemos que maximizar :

$$\theta^* = \arg\max_{\theta} \prod_{k=1}^{m} p(I^k|L^k, \theta) \prod_{k=1}^{m} p(L^k|\theta)$$

Los **parámetros de apariencia** pueden ser **maximizados** en forma **independiente:**

$$u_i^* = \arg\max_{u_i} \prod_{k=1}^{m} p(I^k|l_i^k, u_i)$$

Los **parámetros de conexión** pueden ser **maximizados** incluso sin conocer sin saber cuáles serán parte de E, en forma **independiente:**

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

Definimos la **calidad de una conexión** como:

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

Finalmente, usamos **árbol generador máximo** para buscar encontrar las conexiones de **mejor calidad**:

$$E^* = \arg\max_{E} \prod_{(v_i,v_j)\in E} q(v_i, v_j)$$

Inferencia Exacta

Buscamos la configuración que mejor ubica al objeto en la imagen. Es decir, buscamos resolver el siguiente problema de minimización:

$$L^* = \arg\min_{L} \left(\sum_{i=1}^{n} -\log p(I|l_i, u_i) \sum_{(v_i,v_j)\in E} -\log p(l_i, l_j|c_{ij}) \right)$$

La **minimización es NP-Hard** en general. Sin embargo, si **E es un árbol** se puede utilizar **programación dinámica**. Se puede ver que las ecuaciones que se obtienen son:

$$l_r^* = \arg\min_{l_r} \left(-\log p(I|l_j, u_j) + \sum_{v_c\in C_j} B_c(l_j) \right)$$

$$B_j(l_i) = \min_{l_j} \left(-\log p(I|l_j, u_j) - \log p(l_i, l_j|c_{ij}) + \sum_{v_c\in C_j} B_c(l_j) \right)$$

La complejidad temporal es entonces $\mathcal{O}(nH)$ donde H es el tiempo necesario para calcular $B_j(l_i)$. Si se usa la definición de B para su cálculo obtenemos que $H \in \mathcal{O}(|L|^2)$. Sin embargo, usando **algoritmos basados en la transformada distancia**:

$$\mathcal{D}_f(x) = \min_{y\in\mathcal{G}} (\rho(x, y) + f(y))$$

Se puede obtener un algoritmo **lineal**.

Inferencia Aproximada

1: *calcular* $p(l_r|I, \theta)$
2: *push*(Q,r)
3: **while** no este vacía Q
4: k := pop(Q)
5: *muestrear* de $p(l_k|l_{\pi_k}, I, \theta)$
6: **para cada** hijo i de k en E
7: *push*(Q,i)

Una posible forma de calcular $p(l_r|I, \theta)$ es usando marginalización:

$$p(l_r|I, \theta) = \sum_{l_1} \sum_{l_{r-1}} \sum_{l_{r+1}} \sum_{l_n} \left(\prod_{i=1}^{n} p(I|l_i, u_i) \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij}) \right)$$

El problema con este enfoque es que requiere **tiempo exponencial**. Sin embargo, como **E es un árbol**, lo anterior se puede re-escribir:

$$p(l_r|I, \theta) \propto p(I|l_r, u_r) \prod_{v_c\in C_r} S_c(l_r)$$

Con S **definidas** como:

$$S_j(l_i) \propto \sum_{l_j} \left(p(I|l_j, u_j)p(l_i, l_j|c_{ij}) \prod_{v_c\in C_j} S_c(l_j) \right)$$

Para el paso 5 del algoritmo, es necesario calcular la función de densidad antes de muestrear. Para eso, se tiene en cuenta que:

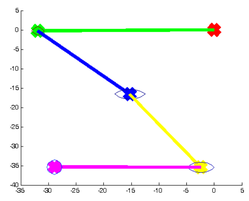
$$p(l_j|l_j, I, \theta) \propto p(I|l_j, u_j)p(l_i, l_j|c_{ij}) \prod_{v_c\in C_j} S_c(l_j)$$

El cómputo de las funciones S se puede acelerar expresándolas como **convoluciones con una función Gaussiana** y usando que las mismas son separables.

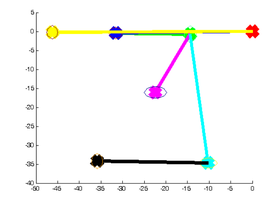
Aplicación: Caras

Modelo de apariencia: $p(I|l_i.u_i) \propto \mathcal{N}(\alpha(l_i), \mu_i, \Sigma_i)$

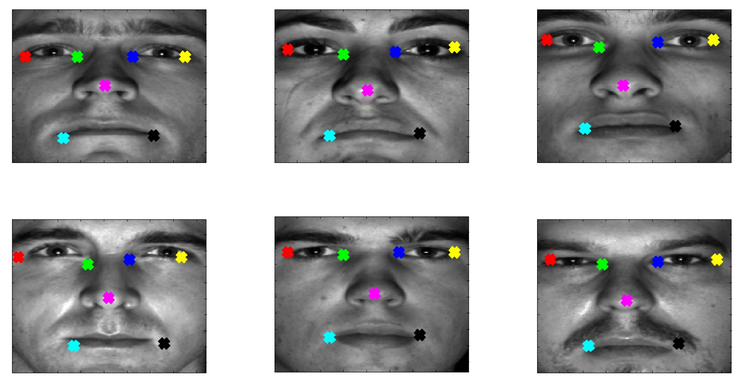
Modelo de conexiones: $p(l_i, l_j|c_{ij}) = \mathcal{N}(l_i - l_j, s_{ij}, \Sigma_{ij})$



Estructura aprendida con 5 partes



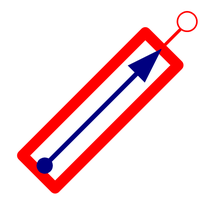
Estructura aprendida con 7 partes



Aplicación: Cuerpo humano

Modelo de ubicación:

$$l = (x, y, s, \theta)$$



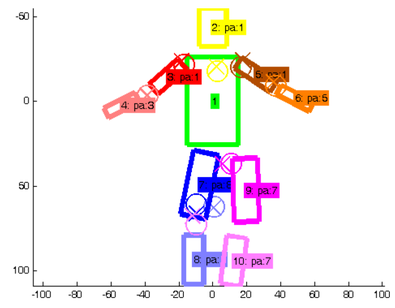
Modelo de conexiones:

$$p(l_i, l_j|c_{ij}) = \mathcal{N}(x_i' - x_j', 0, \sigma_x^2) \mathcal{N}(y_i' - y_j', 0, \sigma_y^2) \mathcal{N}(s_i - s_j, 0, \sigma_s^2) \mathcal{M}(\theta_i - \theta_j, \theta_{ij}, \kappa)$$

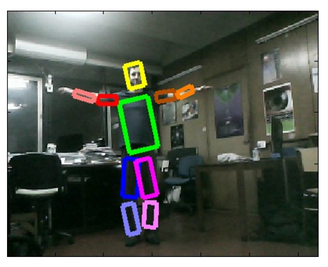
Cálculo de la posición de las juntas:

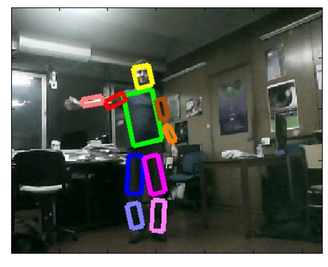
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + s_i R_{\theta_i} \begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix}$$

$$\begin{bmatrix} x_j' \\ y_j' \end{bmatrix} = \begin{bmatrix} x_j \\ y_j \end{bmatrix} + s_j R_{\theta_j} \begin{bmatrix} x_{ji} \\ y_{ji} \end{bmatrix}$$



Estructura aprendida





Referencias

[1] P. F. Felzenswalb and D.P. Huttenlocher. Pictorial structures for object recognition. *Int. J. Computer Vision*, 61(1), January 2005.

[2] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67-92, January 1973.

[3] D. Ramanan and C. Sminchisescu. Training deformable models for localization. IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[4] Y. Yang, D. Ramanan. Articulated Pose Estimation using Flexible Mixtures of Parts. *Computer Vision and Pattern Recognition.* June 2011.