Lecture 13.

Object models are simpler if interest points (IPs) are used. These are sparser and more descriptive than edges, but they provide a weaker descriptor of the object (often unintuitive).

Detect IPs → Brady-Kadir detector with SIF descriptor

Describe IPs → SIFT descriptor

Intuition

Histograms of edges inside four windows

Edges/orientation roughly invariant to illumination

Histogram of edges = invariant to illumination & some viewpoint changes

Model

OR node + variable + Background Model

The OR node selects between models of form

\[ \text{polar} \quad \begin{cases} \mathcal{Q}^D(w,\mathcal{Q},\omega,\omega) \\ (1,2,\ldots,n) \\ (1,2,3), (2,3,4), (3,4,5), \ldots \end{cases} \]

AND-models

Data term for each node

\[ 2^D \mathcal{Q}^D(c,\mathcal{Q},I) \] attracts \( \omega \) to an IP. clique - set of vertices s.t. each pair of vertices are connected by an edge.

For OR nodes \( \sum_{\mathcal{Q} \in \mathcal{Q}} \sum_{\mathcal{Q} \in \mathcal{Q}} \mathcal{S}(\mathcal{Q}^D(w,\mathcal{Q},\omega,\omega), f(\mathcal{Q})) \) function.

Background model - generates remaining IPs in the image.
State Variables

\( \mathbf{y} = (x, \theta, A, u) \)
- attribute
- position
- orientation

Model → full energy

= sum of data terms \( 2p, q \)
+ sum of geometric terms \( x, y \)
+ one node
+ background

Inference: if model is specified
(graph structure + parameter \( \lambda \))
then can do inference by dynamic programming
followed by exhaustive search to deal with the one node.

Note: robustness = 2 out of 3 rule

- if we miss the state of one node for
a triangle clique then we can
predict its position from the position of
the other two.

(Helpful if one IP is not detected
because of occlusion or failure of the detector)

Learning

- suppose the structure is known but we do not know the lambda 2 parameter.
- then we can learn from training data
by maximum likelihood using the EM algorithm - the hidden variables, which we need to estimate, are which sub-model is used and what are the positions, orientations, and attributes of the nodes.

Standard EM - we can use dynamic programming (DP) to make the learning practical. Initial Conditions?
Learning the Structure and Initializing Parameters

Dictionaries

Ignore spatial relations:

Perform clustering on the appearance of the IR's

(clustering - e.g. k-means)

Treat each cluster as an element of an appearance dictionary $D^a$

Each IP can be represented by an element of $D^a$

Each IP is a very simple graphical model with single node

$$\theta \sim \mathcal{N}(\mathbf{X}, \mathbf{A}, \mathbf{W})$$

Note: in previous lecture, we defined a dictionary for edges - e.g. $$\mathcal{L} \in \mathbb{R}$$

Now cluster triplets of IP by geometry.

$$\Delta \beta$$

As in previous lecture,

Spatial relation - gaussian distributed on relative position

This gives a Triplet Dictionary $D^t$

Note: this procedure is similar to learning a level-2 dictionary of edge structures from a level-1 dictionary of edgettes.

Each element of $D^t$ is a small graphical model

Each dictionary element has a score $\theta$ times in $\mathbb{R}$

We can now build an object model by combining elements from the triplet-dictionary.
**Learning the Structure & Inclusion**

**Default Model:** All data is generated by a background model.

\[ P(n) = \prod_{i=1}^{n} P(x_i | \theta, \phi) \]

- \( P(n) \) is the data.
- \( P(x_i | \theta, \phi) \) is the model for each \( x_i \).

**Key Point:** This model assumes that the data points are generated independently from a distribution \( P(x_i | \theta, \phi) \).

**Note:** In practice, little difference if we replace \( P(x_i | \theta, \phi) \) by a uniform distribution.

**First Step:**

- Select a triplet \( \beta \) from the Triplet Directory.

- Use model: \( \beta + \text{Background} \)

- \( \beta \) is the best model that can explain the data.

- Estimate parameters using EM, by \( \beta \) or by background dictionary.

- Perform model selection:

\[ 0 \rightarrow \beta \rightarrow 0 \]

Select the best model:

- \( \beta \) or any other candidate

**Second Step:**

- Two options:
  - Either add a new triplet which is compatible with the first.
  - Or, add a new triplet as an alternative.
More Generally:

After N stages:

model is of form:

\[ \text{OR} \quad + \quad \text{B} \quad \text{Bagged} \]

\[ \begin{array}{c}
\text{OR} \\
\text{B} \\
\text{F} \\
\text{B} \\
\text{B} \\
\text{B} \\
\end{array} \]

Can add triple from the Triplet Dictionary to any OR

in each case - must ensure compatibility.

(2) or can add a new triplet

For all cases, perform model selection to determine
whether the new model is better than the old model.

If not, stop growing the model.

What does the OR's do?

Suppose the dataset consists of several different types of objects?

\[ \begin{array}{c}
\text{Motorbike} \\
\text{Face} \\
\text{Background} \\
\end{array} \]

This procedure should learn

Learning OR's is possible because the
Interest Points are distinctive.
Enhancing the model by adding a mask.

Choose unique feature $f(x)$.

\[ P(f(x) | L) = \prod_{x : L(x) = 2} P(f(x) | L(x) = 2) \prod_{x : L(x) = 0} P(f(x) | L(x) = 0) \]

i.e. assume the statistics of $f(x)$ are different inside and outside the object.

\[ P(L) = \prod_{x} P(L(x) | M) \]

where $M$ is a mask.

Hence $P(L | f(x)) = P(f(x) | L) P(L)$. (High prob)

But, how to deal with changes in size, orientation, and position?

How to learn the mask $M$? i.e. $P(L(x) | M)$?

Couple the mask to a model with IP's

\[ \omega = f(\omega_0, \omega_y, \omega_x, \omega_s) \]

state of $\omega$ gives the position, orientation, and scale of the mask.

Strategy: learn the model with IP's

1. learn a mask model using IP model to help train it.