1. Introduction

Understanding images by recognizing its constituent objects is a challenging task. The reason is that objects have a large variance in appearance. While visual cues can help the recognition process, it is unlikely that all variants of objects will be present in the training data. Recently, it has been shown that higher-level knowledge, such as hierarchical decomposition of scenes into lower-level substructures, contextual information and geometrical constraints, can improve object detectors [2, 7, 5]. However, so far, there does not exist a general and principled way to employ all this high-level knowledge in vision problems. It is precisely this gap that we try to bridge by introducing a logical language for visual scene understanding that meets several requirements: 1) it allows one to describe the data and domain knowledge in an interpretable manner, 2) it can incorporate both appearance and higher-level features (e.g. output of local classifiers, numeric features, qualitative relations), and 3) it works well in practice, both in what concerns performance and scalability. We employ kLog [3], a logical and relational language for kernel-based learning which can handle both numeric and symbolic data. Relational representations allow one to elegantly and declaratively represent structured scenes, that is, both the entities and the qualitative relationships among them. Furthermore, one can easily incorporate domain knowledge using logical rules. In this abstract, our goal is to understand scenes hierarchically, by recognizing known objects at different levels of granularity.

Most other approaches to object recognition combine local features in purely statistical frameworks [6]. Bag-of-words represent an object as a set of visual words, spatial pyramids add loose 2D spatial locations, while part-based models are graphical models of topologies between visual parts. Although they model spatial constraints between features, their representational power is limited by an upper bound on the number of parts and a lack of symbolic dependencies. More flexible hierarchical approaches are based on

2. Problem representation and formulation

We represent an image by a set of parts together with their properties and relations. In a hierarchy an image is described at several layers. Figure 1 shows a partial hierarchical description of a house facade. At each layer, an image consists of parts, relations and properties. Parts are regions of interest in the image, e.g. corners, windows/doors and houses at layers 1, 2 and 3, respectively. Relations between parts specify the internal spatial configuration of an object (e.g. a top-left corner is above a bottom-left one in a door), membership relations (e.g. a window is part of a house), or functional relationships (e.g. relative height). Properties of
parts and relations specify discrete or numerical attributes. Each part has a class property, plus possibly others (e.g. appearance features, aspect ratio). A relation may have properties as well (e.g. the Euclidian distance between parts).

At each layer, we build an image description consisting of parts at that layer, relations between them, their internal spatial structures and properties. Parts are generated using local spatial configurations of classified lower-level parts. A configuration of parts is grouped into a higher-level part using the part-of relation. The target problem at one layer is then to predict the class property of the parts (e.g. window or door at layer 2). A hierarchical description of the image can be obtained by collapsing all layer descriptions into one larger description. Using this representation, hierarchical information and context are incorporated by relations between the parts, while appearance cues are given as properties. Domain knowledge can be easily specified by defining relations. For example, we can define: closeUp(A, B, D_e) ← up(A, B), close(A, B, D_e, Th)

Employing the kLog Language. We now employ kLog, which starts directly from the logical and relational representation of the problem, generates a set of features and then uses them to learn linear models. The representation used by kLog is illustrated on the right hand side of Figure 1. The ground atom part1(p2, type1, topL) defines an entity, while closeUp(p2, p1, 60.1) introduces a relationship. Consequently, each scene corresponds to a visual logical interpretation, that is, a set of ground tuples in the relational representation. kLog learns to recognize objects from visual interpretations. More formally, the task is to map an interpretation $i = (x, y)$ into a feature vector $\theta(i) = \theta(x, y)$, with $\theta(x, y)$ belonging to a feature space where supervised learning algorithms can be used. kLog implicitly computes $\theta(i)$ via a kernel function $\kappa(i, i') = \langle \theta(i), \theta(i') \rangle$. First $i$ is mapped into an undirected grounded and labeled graph $G_i$ via the graphicalization step. In the example, this results in the figure on the left. For every tuple in the database, a vertex is labeled with the name of the relation, grounded with the list of properties and introduced in the graph. Edges connect vertices that share identifiers in the tuples.

Next, feature generation allows a propositional formulation of the problem. For this, $\theta(i)$ is extracted from $G_i$ by computing a kernel function on pairs of graphs $\kappa(i, i') = \kappa(G_i, G_i')$. The graph kernel should allow fast computations, given the large grounded graphs that can result. In addition, it should be a general purpose kernel with a flexible bias so to adapt to a wide variety of domains; some problems can be solved appearance-based, others are more structured or contextual. The graph kernel of kLog assures these properties. It is a decomposition kernel where parts are pairs of sub-graphs. Graph parts are defined by the relation $R_{r,d}(A_v, B_u, G)$ between 2 rooted graphs $A_v, B_u$ and a graph $G$, which selects all pairs of neighborhood graphs of radius $r$ whose roots are at distance $d$ in $G$. The decomposition kernel on the relation $R_{r,d}$ is defined as:

$$\kappa_{r,d}(G, G') = \sum_{A_v, B_u \in \kappa_{r,d}^{-1}(G)} \kappa((A_v, B_u), (A'_v, B'_u))$$

where $\kappa((x, y), (x', y'))$ is a soft match weighted decomposition kernel relying on local multinomial distributions of the vertex labels. It can deal with dense graphs, discrete and numerical labels. The final decomposition kernel is $\kappa(G, G') = \sum_{r,d} \kappa_{r,d}(G, G')$.

Experiments and Results. We experimented using kLog on a fixed split of 60 images of rows of house facades from different countries. The task was to detect windows, doors and houses. Images were annotated with object classes and their bounding boxes. Experiments were performed layer-wise. We started at the base layer, where corners were first detected and then classified to identify the ones belonging to windows and doors. A model based on feature descriptors was trained in this scope. Next, window/door detectors were built starting from from the classified corners. High-scored detections of windows and doors were then employed at the next layer to recognize houses, for which a house detector was trained. Precision-recall curves are shown in Figure 2. The learning steps can be reiterated to also consider hierarchical features as top-down feedback.

Laura Antanas is supported by the European Commission under contract FP7-248258-First-MM.

References


1Rules $up(A, B)$ and close(A, B, D_e, Th) are defined similarly.

2In practice a normalized version is used.


