Draft Version. Final version published in The Third Workshop on Recognition and Action for Scene Understanding (REACTS), 2015.

UPGMpp: a Software Library for Contextual Object Recognition

J.R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez-Jimenez

System Engineering and Automation Dept., University of Málaga, Campus de Teatinos, 29071, Málaga, Spain,
jotaraul@uma.es,
WWW home page: http://mapir.isa.uma.es/

Abstract. Object recognition is a cornerstone task towards the *scene understanding* problem. Recent works in the field boost their performance by incorporating contextual information to the traditional use of the objects' geometry and/or appearance. These contextual cues are usually modeled through *Conditional Random Fields* (CRFs), a particular type of undirected *Probabilistic Graphical Model* (PGM), and are exploited by means of probabilistic inference methods. In this work we present the *Undirected Probabilistic Graphical Models in C++* library (UPGMpp), an open source solution for representing, training, and performing inference over undirected PGMs in general, and CRFs in particular. The UPGMpp library supposes a reliable and comprehensive workbench for recognition systems exploiting contextual information, including a variety of inference methods based on *local search*, *graph cuts*, and *message passing* approaches. This paper illustrates the virtues of the library, i.e. it is efficient, comprehensive, versatile, and easy to use, by presenting a use-case applied to the object recognition problem in home scenes from the challenging NYU2 dataset.

Keywords: contextual object recognition, probabilistic graphical models, probabilistic inference, scene understanding

1 Introduction

Scene understanding systems aim to provide a valid interpretation of the perceived imagery which can be leveraged by a large variety of innovative technologies, like robotics, assistance to visual impaired, autonomous driving, etc. *Object recognition* is a key component of these systems, whose results become crucial for a proper understanding of the scene. Modern approaches improve the object recognition performance by incorporating contextual information of the objects, in addition to their usually employed geometry and/or appearance properties [1, 12–16, 20, 23]. This enables the disambiguation of confusing classifications provided by methods *only* relying on properties of the objects themselves [5]. Let's suppose, for example, a scene with a brown, cylindrical object. A method relying on geometric/appearance properties could have problems to classify it as a pot or a flowerpot, however, if it is found on a stove, the pot option is more probable.

The *Probabilistic Graphical Models* (PGMs) framework [7] has been widely used to exploit contextual relations among objects. Concretely, a particular type of PGM, namely *Conditional Random Field* (CRF), has focused the interest of researchers given its suitability to model this kind of problems. PGMs integrate a compact and powerful graph-based representation of complex probability distributions defined over high-dimensional spaces, and employ probabilistic inference algorithms to efficiently perform queries of interest over it. Of particular concern is the *Maximum a Posteriori* query (MAP), since it provides the recognition results by computing the most probable category assignations to the scene objects[1]. The simplest MAP inference method, called *exact inference*, exhaustively tests all the possible objects' category assignations, which is an unfeasible approach in many real-world problems. Instead, approximate methods are exploited, which can be roughly classified into three major groups: local search [2], graph cuts [4], and message passing algorithms [9].

Most contextual-based object recognition works rely on an ad-hoc implementations of both the PGMs framework and inference algorithms [1, 12, 20, 23]. This makes it difficult to conduct a fair comparison between state-of-the-art works, even when they report results resorting to the same dataset [15]. There are some publicly available software libraries implementing this framework [11, 18], but they are not suited for the contextual object recognition problem (e.g. they only handle *chain-structured* models), or their applicability to this issue is limited.

This paper presents the *Undirected Probabilistic Graphical Models in C++* (UPGMpp) library, a software package for working with undirected PGMs, as is the case of CRFs, and its application to scene object recognition. UPGMpp exhibits a number of features that make it suitable for facing this particular problem: i) it works with discrete random variables, like the ones needed to model the possible objects' categories (e.g. chair, table, book, etc.), ii) it handles unary and pairwise relations, needed for representing the objects' features and relationships, and iii) it enables the representation of arbitrary structures, i.e. it can codify any number of scene objects and relations among them. This library implements inference methods from the three major groups mentioned above, including for example Iterated Conditional Modes (graph search), α-β swaps (graph cuts), or Loopy Belief Propagation (message passing). Therefore, UPGMpp provides a good basis for their evaluation and integration into recognition systems exploiting context. From an algorithmic point of view, the library also includes mechanisms to train PGMs and to perform probability queries (carry out marginal inference), as well as functionality for storing/loading PGMs from files through serialization. UPGMpp is designed to be efficient, versatile, extensible, and easy to use through clear and intuitive APIs, and resorts to well known libraries for numerical optimization (libLBFGS [10]), matrix operations (Eigen [6]) and memory handling (Boost [17]). It is entirely open-source, and is publicly available under a GNU General Public License (`http://mapir.isa.uma.es/work/upgmpp-library`). The library is distributed along with a number of code tutorials, so the user can master and start using it quickly.

[1] Along this paper we employ the term inference to refer to MAP inference.

Draft Version. Final version published in The Third Workshop on Recognition and Action for Scene Understanding (REACTS), 2015.

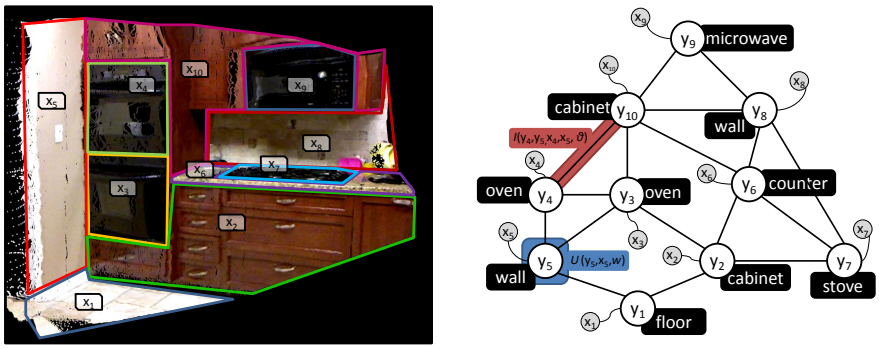

Fig. 1. Left, RGB-D image of a kitchen from the NYU2 dataset including the scene objects marked as $\boldsymbol{x} = \{x_1, ..., x_{10}\}$. Right, CRF structure built from the scene. The blue shape represents the scope of an unary factor, while the red one states the scope of a pairwise factor. Random variables are labeled with the categories assigned by the execution of a probabilistic inference method over the CRF.

As an illustrative example of its suitability to the contextual object recognition problem, we describe a use-case of recognizing objects from home scenes within the challenging NYU2 dataset [19]. Performance results regarding the execution time of inference and training methods within UPGMpp are also shown.

The next section describes the application of Conditional Random Fields to the scene object recognition issue, in order to provide a theoretical background for a better understanding of the library components. Then, section 3 presents the UPGMpp library, as well as the inference algorithms that it implements. Section 4 illustrates the UPGMpp application to the recognition of objects from scenes within the NYU2 dataset. Finally, section 5 outlines the conclusions and possible future work.

2 Contextual Object Recognition through Conditional Random Fields

The object recognition problem can be stated as the assignation of classes (e.g. table, chair, notebook, etc.) to a number of regions observed in imagery from a given scene. Let's consider the following definitions to address this problem from a probabilistic stance:

- Define $\boldsymbol{x} = \{x_1, .., x_n\}$ as the set of n objects appearing in the scene, where each x_i is characterized through a vector of m features, $\boldsymbol{f}_{\boldsymbol{x}_i u} = [f_{\boldsymbol{x}_i u_1}, .., f_{\boldsymbol{x}_i u_m}]^T$, e.g. their size, color, orientation, etc.
- Let $L = \{l_1, .., l_k\}$ be the set of k possible object classes.
- Define $\boldsymbol{y} = \{y_i, .., y_n\}$ as the set of discrete random variables over L, where each y_i assigns a class from L to its associated object x_i.

Thereby, the object recognition problem, modeled through a Conditional Random Field [7], is such of maximizing the probability distribution $P\left(\boldsymbol{y}|\boldsymbol{x}\right)$, i.e., to find the most probable classes' assignation from L to the random variables in $\boldsymbol{y}$ according to the characterized objects in $\boldsymbol{x}$. The structure of a CRF is represented by a graph $H = (V, E)$, where V is a set of nodes associated to random variables, and E stands for a set of edges liking related variables/nodes. Regarding the problem at hand, a node represents a variable from $\boldsymbol{y}$, and an edge connects two variables which associated objects are contextually related in the scene, i.e. they are placed close to each other. Figure 1-left shows an scene with ten objects, which are represented as nodes in the CRF in figure 1-right. We can see how, for example, the stove is related to the cabinet, the wall, and the counter, so their associated nodes are linked. Thereby, the probability distribution $P(\boldsymbol{y}|\boldsymbol{x})$ can be factorized over this graph structure H, which is expressed for convenience by means of log-linear models [7]:

$$P\left(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\theta}\right) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\theta})}\, e^{-\epsilon(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\theta})} \tag{1}$$

where $Z(\cdot)$ is known as the partition function, so $\sum_{\xi(\boldsymbol{y})} P\left(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\theta}\right) = 1$, being $\xi(\boldsymbol{y})$ a possible assignation to the variables in $\boldsymbol{y}$, $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ are vectors of weights learned during the CRF training, and $\epsilon(\cdot)$ is the energy function, defined as:

$$\epsilon(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\theta}) = \sum_{i\in V} U(y_i, x_i, \boldsymbol{\omega}) + \sum_{(i,j)\in E} I(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) \tag{2}$$

being $U(\cdot)$ and $I(\cdot)$ the so-called unary and pairwise factors respectively. These factors can be seen as functions encoding small parts of the whole $P(\boldsymbol{y}|\boldsymbol{x})$ over the nodes and edges of the graph H. Thus, an unary factor gives an intuition about how probable is for a node $\boldsymbol{y}_i$ to belong to a class from L according to the features of the object $\boldsymbol{x}_i$. On the other hand, a pairwise factor speaks about an edge, and states the compatibility of two related variables being assigned a certain pair of classes from L. The scope of these factors is shown in figure 1-right. They are defined by means of log-linear models as follows:

$$U(y_i, x_i, \boldsymbol{\omega}) = \sum_{l\in L} \delta(y_i = l)\boldsymbol{\omega}_l \boldsymbol{f}_{\boldsymbol{x}_i u} \tag{3}$$

$$I(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) = \sum_{l_1\in L}\sum_{l_2\in L} \delta(y_i = l_1)\delta(y_j = l_2)\boldsymbol{\theta}_{l_1,l_2} \boldsymbol{f}_{\boldsymbol{x}_i\boldsymbol{x}_j p} \tag{4}$$

where $\delta(y_i = l)$ is the Kronecker delta function, and $\boldsymbol{f}_{\boldsymbol{x}_i\boldsymbol{x}_j p}$ is the vector of pairwise features characterizing the relationship between the objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

The training of a CRF consist of finding the vectors of weights $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ that maximize the likelihood function:

$$\max_{\boldsymbol{\omega},\boldsymbol{\theta}} L_P(\boldsymbol{\omega},\boldsymbol{\theta} : D) = \max_{\boldsymbol{\omega},\boldsymbol{\theta}} \prod_{d\in D} P\left(y_d|x_d\right) \tag{5}$$

where D is the set of all the scenes used for training, compound each one of a set of characterized objects x_d, and their respective ground truth classes y_d. Solving Eq. 5 requires the computation of the partition function, which is unfeasible in practise. Section 3.1 introduces the approaches implemented in the UPGMpp library to face this issue.

Once the CRF is trained, it can handle the execution of inference algorithms to contextually recognize objects. Thus, given a scene, its particular graph structure $H = (V, E)$ is built according to the relations shown by its constituent objects (see figure 1). The (MAP) inference goal is to find the classes assignation $\hat{\boldsymbol{y}}$ that maximizes the probability distribution $P(\boldsymbol{y}|\boldsymbol{x})$ factorized over H, that is:

$$\hat{\boldsymbol{y}} = arg\max_{y} P(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\theta}) \tag{6}$$

Again the computation of the partition function $Z(\cdot)$ is needed. However, since given a certain scene its value remains constant, this expression can be simplified by:

$$\hat{\boldsymbol{y}} = arg\max_{y} e^{-\epsilon(\mathbf{y}, \mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\theta})} \tag{7}$$

Despite this simplification, to compute an exact solution of such an equation is still unfeasible due to the huge number of possible assignations to be checked (k^n), which motivates the use of approximate inference methods. The algorithms implemented in the UPGMpp library for this are described in section 3.2.

3 UPGMpp Library

The Undirected Probabilistic Graphical Models in C++ (UPGMpp) library is an open-source software for dealing with undirected PGMs, e.g. Markov Random Fields, or Conditional Random Fields. The library works with discrete random variables and handles local and pairwise relations, i.e. first and second order PGMs. UPGMpp provides tools for: i) defining graph representations, ii) completing a fast training of models, and iii) performing efficient inference queries (both probability and MAP queries). This section presents an overview of the most relevant features of the library and its components (section 3.1), as well as the available inference algorithms (section 3.2).

3.1 Overview

The UPGMpp library is divided into three packages (see figure 2):

- ***base***. Implements the functionality for building and managing PGM graphs.
- ***training***. Permits the definition of training datasets to tune a PGM.
- ***inference***. Implements algorithms to perform probability and MAP inference queries over PGMs.

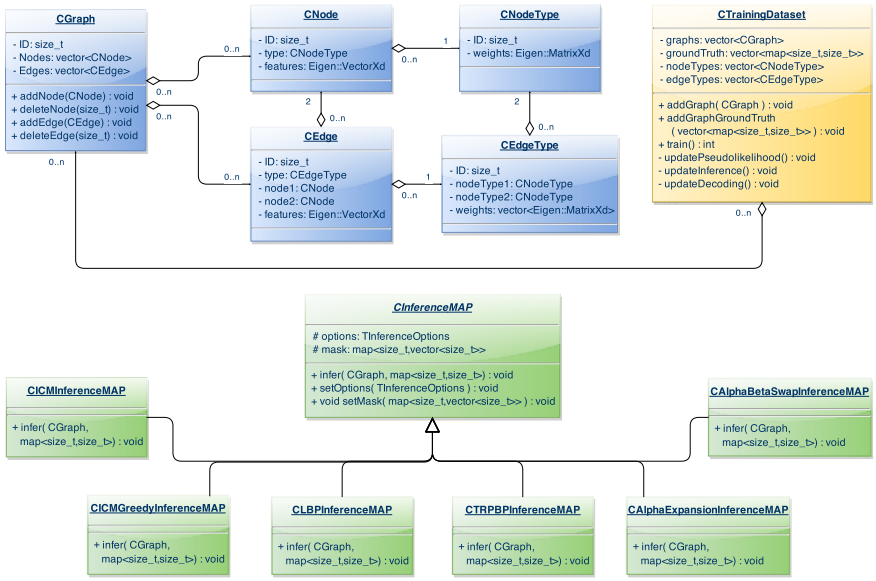

Fig. 2. Simplified UML class diagram of the main classes within the *base* (blue), *training* (yellow) and *inference* (green) packages within the UPMGpp library. For interpretation of references to color, the reader is referred to the web version of this work.

The *base package* provides an easy way to create and manage graphs representing PGM structures. Instances of nodes from V can be created employing the `CNode` class, as well as edges from E through the `CEdge` one. The `CNodeType` and `CEdgeType` classes permit us the definition of typed nodes and edges. Having the sets of nodes and edges, they can be inserted into an instance of the `CGraph` class, which represents the graph structure $H = (V, E)$. The factors within nodes (unary) and edges (pairwise) have been implemented through log-linear models (recall equations 3 and 4), although the user can easily define a different way to compute them through a prototype function.

The *training package* provides mechanisms for building datasets employing the `CTrainingDataset` class, i.e. sets of graphs along with their ground truth categories (see the *yellow* class and methods in figure 2). Once created and populated, a dataset can be used to train an undirected PGM, i.e. to find the vectors of weights $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ in equation 5. Recalling that the computation of such an equation is unfeasible in practice, two major approaches are considered in the literature: the definition of tractable alternative objective functions, like the *pseudolikelihood*, and the use of approximate inference processes (including MAP and marginal inference) [7]. Both approaches have been implemented and are available to the user in the *training package*.

Finally, the *inference package* implements a number of state-of-the-art inference algorithms for performing both, probability and MAP queries (recall equa-

tion 7), although in this work we focus on MAP since it provides the scene object recognition results. To facilitate its use and future expansion, every MAP inference algorithm inherit the same functionality from a base class, `CInferenceMAP`, and implements the same abstract method for performing inference (see *green classes* in figure 2). The implemented MAP inference methods are described in section 3.2.

UPGMpp resorts to the also open-source project libLBFGS [10] for performing numerical optimization, and the Eigen [6] library for performing fast matrix operations. The Boost library [17] is used to avoid unnecessary re-copy of data across the library methods by means of shared smart pointers. This library is also used for serialization purposes, which adds the possibility of storing/loading graphs from/to files, enabling the long-term life of PGMs beyond execution time.

3.2 MAP inference methods

This section briefly describes the theory behind the approximate MAP inference methods implemented in the UPGMpp library. The interested reader can refer to the provided citations for further information.

Local search methods. Local search methods are the simplest approaches for approximated MAP inference, and they are widely used due to their easy implementation and acceptable results. In a nutshell, these methods operate over a set of candidate solutions called *search states*, which define a *search space*. In object recognition, a search state can be seen as a certain assignation $\xi(\boldsymbol{y})$ to the variables in $\boldsymbol{y}$, which have an associated likelihood value, and the search space corresponds to the set of all possible assignations. Thus, starting at a certain state $\xi_c(\boldsymbol{y})$, a local search method checks if there is a state among the set of *similar states*, defined as $Sim(\xi_c(\boldsymbol{y}))$, showing a higher likelihood value. If so, the algorithm *moves* to it as the current search state $\xi_c(\boldsymbol{y})$. Thereby, these methods perform small movements while exploring the search space, always increasing the expected likelihood, until a local maximum is reached, i.e. there is not a similar state to the current one with a higher likelihood. Algorithms within this group differ in how they define the similarity function $Sim(\xi_c(\boldsymbol{y}))$ for a given state $\xi_c(\boldsymbol{y})$. Next, the *Iterated Conditional Modes* (ICM) local search method and its *Greedy* variant are described (see [2] for further detail.)

Iterated Conditional Modes. ICM operates by giving an initial assignation to the variables in $\boldsymbol{y}$, and iterating over those variables to maximize the local conditional probability:

$$\hat{y_i} = arg\max_{y_i} P(y_i|\boldsymbol{y}_{N_H(y_i)}, x_i, \boldsymbol{x}_{N_H(y_i)}) \tag{8}$$

where $\boldsymbol{y}_{N_H(y_i)}$ and $\boldsymbol{x}_{N_H(y_i)}$ are sub-vectors of the original $\boldsymbol{y}$ and $\boldsymbol{x}$ ones that contain the random variables and observations of the neighbor nodes of y_i in a certain graph H. Thus, being $\xi_c(\cdot)$ the current assignation to a set of random

variables, the set of similar states is defined as $Sim(\xi_c(\boldsymbol{y})) = \{\xi(\boldsymbol{y}) \mid \xi(\boldsymbol{y_{-i}}) = \xi_c(\boldsymbol{y_{-i}})\}$. This algorithm ends when convergence is achieved, i.e., an iteration over all the variables is completed without changing the search state, or when a given limit of iterations is reached.

Greedy ICM . The greedy variant takes the same initialization and ending criteria, but instead of performing a movement per random variable in $\boldsymbol{y}$, it first iterates over all the variables, and then applies the movement that yields the maximum likelihood increment. In this case the set of similar states is defined as: $Sim(\xi_c(\boldsymbol{y})) = \{\xi(\boldsymbol{y}) \mid diff(\xi(\boldsymbol{y}), \xi_c(\boldsymbol{y})) = 1\}$, where $diff(\xi(\boldsymbol{y}) - \xi_c(\boldsymbol{y}))$ yields the number of random variables with different assigned classes, i.e. two states are similar if only one random variable in $\boldsymbol{y}$ shows a different assignation. This algorithm requires on average more iterations to converge than the original ICM, but it is more robust against getting stuck in a local maximum.

Graph cuts methods. Graph cuts [4] have been extensively used to efficiently face early vision problems that can be formulated as a minimization of an energy function. This approach reduces the MAP inference task to instances of the minimum cut problem. Let's suppose a binary classification problem ($y_i = \{0, 1\}$) with factors codified over a graph $H = (V, E)$. To apply graph cuts, the graph is modified in the following way: a pair of nodes, s (source) and t (sink), are added so $V_c = \{V, s, t\}$, and two edges linking each node with s and t are included, obtaining the set $E_c = \{E\} \cup \{e_{s \to i}, e_{i \to t}, \forall i \in V\}$. Then, the minimum cut of this new graph $H_c = \{V_c, E_c\}$ is computed, which divides the set of nodes into two sets: the one containing the nodes connected to the source s, called V_s, and the set of nodes V_t linked to the sink t. Finally, the nodes in V_s are classified as belonging to the class 0, and those in V_t to the class 1. This method can be extended to handle non-binary classification problems, as illustrate the α-β *swaps* and the α-*expansions* algorithms [3].

α-β *swaps.* This algorithm iterates over all the possible class pairs (α,β) in L, and checks if there is a swap among the variables assigned to that classes that increments the expected likelihood. Let $V_\alpha = \{V_i = \alpha, \forall i \in V\}$ be the set of nodes/variables assigned to the class α, and $V_\beta = \{V_i = \beta, \forall i \in V\}$ those assigned to β. Then, graph cuts compute the optimal classes assignation for the graph $H_c = (V_c, E_c)$, where $V_c = V_\alpha \cup V_\beta \cup \{s, t\}$ and $E_c = \{e_{ij} \in E \mid (i = \alpha) \cap (j = \beta)\} \cup \{e_{s \to k}, e_{k \to t}, \forall k \in (V_\alpha \cup V_\beta)\}$. In this case, a node connected to the source s in the minimum cut is classified as belonging to the class α, and to β otherwise. A change in the assignation of a node in the minimum cut with respect to its previous one produces an α-β swap move. The algorithm ends when no swap moves increasing the likelihood can be performed.

α-*expansions.* This method iterates over the classes α in L performing α-expansions, i.e. changing the class assigned to a node from $L_{\bar{\alpha}} \in L - \alpha$ to the class α. Thus, for each class, graph cuts are used to compute the minimum cut of the graph

$H_c = (V_c, E_c)$, where in this case $V_c = \{V_{\bar{\alpha}}, s, t\}$ is the set of nodes not assigned to the class α plus the source s and the sink t, and $E_c = \{e_{ij} \in E \mid (\boldsymbol{y}_i \neq \alpha) \cap (\boldsymbol{y}_j \neq \alpha)\} \cup \{e_{s \to k}, e_{k \to t}, \forall k \in V_{\bar{\alpha}}\}$. The nodes connected to the source s in the minimum cut produce an α-expansion, i.e. they replace their assigned class by α, while those linked to the sink t keep their initial class. This process is repeated until no α-expansion can increase the current expected likelihood.

Message passing methods. The message passing approach, also called Belief Propagation (BP) or max-product [22], is based on the exchange of statistical information among related nodes. This is performed by passing messages from node y_i to node y_j, denoted as $m_{ij}(y_j)$, indicating the belief of node y_i about the belonging class of node y_j. These messages are computed in the following way:

$$m_{ij}^t = U(y_i, x_i, \boldsymbol{\omega}) I(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) \prod_{y_k \in N_H(y_i) \setminus y_j} m_{ki}(y_i) \tag{9}$$

where $N_H(y_i) \setminus y_j$ is the set of neighbors of y_i in the graph H less y_j, and t is an iteration counter. Thus, the BP algorithm keeps sending messages between nodes following a certain message scheduling until the graph is calibrated, i.e. the messages exchanged between nodes are the same in two consecutive algorithm iterations. Once calibrated, the belief of each node is computed as:

$$b(y_i) = \kappa U(y_i, x_i, \boldsymbol{\omega}) \prod_{y_j \in N_H(y_i)} m_{ji} \tag{10}$$

being κ a normalization component so the beliefs for node y_i sum to 1. Then, each node y_i is assigned to the class with the highest belief value in $b(y_i)$.

In the case of tree-structured graphs, such a message updating rule yields the optimal maximum. On the other hand, when it is applied to graphs with loops it adopts the name of *Loopy Belief Propagation* (LBP), and it is able to approximate a solution with a reasonable success. Next, we briefly describe the *Tree-Based Reparametrization* message passing algorithm (TRP) [21].

Tree-Based Reparametrization. This method pursuits a more global exchange of statistical information, not only between related nodes, aiming to reach a faster calibration even in cases where traditional BP methods fail. For that, a set of trees $T = \{T_1, .., T_t\}$ are spanned over the original graph $H = \{V, E\}$ in such a way that every node in V belongs to (at least) one tree.

Once the set of trees T is obtained, the algorithm iteratively selects a tree and calibrates it, keeping fixed all the messages from the variables out of the tree. The calibration of a tree can lead to the *miscalibration* of other trees, so the algorithm has to be repeated until global calibration is reached. Once calibrated, the inference results are obtained in the same way as the original LBP algorithm. The interested reader can refer to [21] for more detail.

X

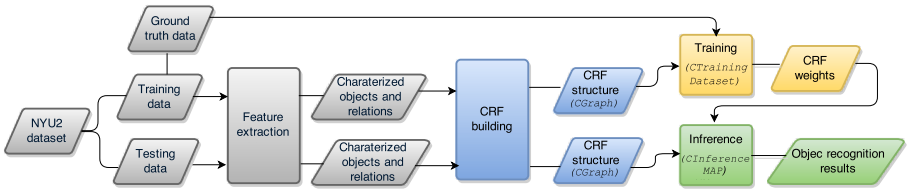

Fig. 3. Processing pipeline when training and exploiting a CRF model using the UPGMpp library. Colored shapes are processes and data handled by the library (see figure 2), while gray shapes are problem specific and, therefore, defined by the user.

4 Contextual Object Recognition Using the UPGMpp Library

This section shows the flexibility and usability of the UPGMpp library when applied to the scene object recognition problem exploiting contextual information. First, we introduce the NYU2 [19] dataset which has been used to test the library (section 4.1). Then, we describe the processes needed for training and testing (performing inference) Conditional Random Fields (CRFs) employing the UPGMpp library (section 4.2). Finally, the recognition results yielded by the different inference methods within UPGMpp are shown (section 4.3), including performance information, which support the suitability of the application of the presented library for the scene object recognition problem.

4.1 The NYU2 dataset

In this work we employ RGB-D images from the NYU2 dataset, which contains a total of 1,449 densely labeled pairs of intensity and depth data. The dataset has been widely used in the literature (e.g. [8]) due to its challenging cluttered scenes from commercial and residential buildings. In this work, we have used 208 scenes belonging to rooms typically found in houses, namely: bedrooms, bathrooms, kitchens and living rooms, containing 1,692 objects that belong to 22 objects classes, e.g. cabinet, counter, bottle, toilet, sofa, lamp, clothes, etc.

4.2 CRFs creation and operation with UPGMpp

Figure 3 shows the general processing pipeline when training a CRF model and exploiting it to perform object recognition. The colored shapes in the image comprise the processes and data provided/managed by the UPGMpp library, and the involved package (as before, blue represents the *base* package, yellow the *training* one, and green the *inference* package). On the other hand, the gray shapes are problem-specific, so they have to be defined by the user. We have instantiated this pipeline for the case of the NYU2 dataset, although it can be replaced by any other.

```
1  CGraph CRFgraph; // Conditional Random Field structure
2
3  Eigen::VectorXd node1Features(X); // Features of the object x6
4  node1Features << 4.11, 0.02, 0.22, 0.70, 0.89, 0.84, 0.17, 1.15, 78.59;
5
6  Eigen::VectorXd node2Features(X); // Features of the object x7
7  node2Features << 0.53, 0.02, 0.02, 0.83, 0.90, 0.87, 0.03, 0.40, 106.84;
8
9  CNodePtr y6 ( new CNode( object, node1Features ) ); // Create nodes
10 CNodePtr y7 ( new CNode( object, node2Features ) );
11
12 Eigen::VectorXd edgeFeatures(X); // Features of the contextual relation
13 edgeFeatures << 3.58, 0.13, 1, 1;
14
15 CEdgePtr edge_y6_y7 ( new CEdge( y6, y7, edgeBetweenObjects, edgeFeatures ) );
16
17 CRFgraph.addNode( y6 ); // Add nodes and edge to the graph
18 CRFgraph.addNode( y7 );
19 CRFgraph.addEdge( edge_y6_y7 );
20
21 // Keep on inserting nodes (objects) and edges (contextual relations)
22 // ...
23
24 // Perform ICM inference over the built CRF graph structure
25 CICMInferenceMAP ICM;
26 std::map<size_t,size_t> resultsMAP; // Map of results <ID_node,category>
27 ICM.infer( CRFGrpah, resultsMAP );  // Infer the recognition results
```

Fig. 4. A simple example of the use of the UPGMpp library.

The NYU2 dataset has been split into training and testing scenes which have to be processed in order to extract the features of the objects appearing in them and their relationships. These features are defined by the user, and in this work we have used the following object/node ones: orientation, planarity, linearity, minimum, maximum and centroid heights from the floor, volume, area of its biggest face, and hue variation, while the chosen contextual/edge features have been: difference of orientation, vertical distance, *is on* relation[2], and a bias value that states the compatibility of the related object classes. The extracted features from the training scenes are used to build their respective CRF representations (instances of `CGraph`), which together with ground truth information are inserted into an instance of the `CTrainingDataset`. Then, the selected training method computes the vectors of weights of the CRF model. In UPGMpp these weights are stored within the node and edge types (instances from `CNodeType` and `CEdgeType` respectively), so all the CRF graphs employing these node and edge types share the same vectors of weights.

On the other hand, for each scene into the testing data, its CRF structure is built according to the features shown by their constituent objects. Figure 4 lines 1-22 shows a code snippet where two scene objects/nodes are created and characterized (concretely, x_6 and x_7 from figure 1), as well as their contextual relation/edge, and then inserted into a CRF structure. Notice that both nodes share the same node type, `object`, and the used edge type is defined as `edgeBetweenObjects`. The same process is repeated for all the objects and rela-

[2] This feature takes the value 1 if an object is placed on the other one, and 0 otherwise.

tions appearing in the scene. Finally, the chosen inference process over the CRF structure gives the object recognition results, which are obtained for every scene within the testing data. As an illustrative example, figure 4 lines 25-27 shows the definition of an ICM inference object, and its use to get the recognition results for a given CRF structure.

4.3 Contextual Object Recognition results

This section shows the results of applying the UPGMpp to the NYU2 dataset excerpt, as well as the computational time required for training and inference. This outcomes come from a 4-random-fold cross-validation, i.e. the 208 scenes were randomly divided into 4 folds with equal size, then three out of the four folds were used to train a CRF model, while the remaining fold was used to evaluate its performance. This process is repeated a total of 100 times, and the results are computed as the average of all the evaluations. The training of the CRF models has been done through the optimization of the pseudolikelihood function.

Table 1. Scene object recognition results employing the different inference methods within the UPGMpp library with/without contextual information. It is also shown their mean execution time as well as the execution time in the worst cases (in ms.).

Method	ICM	Greedy	α-expansions	α-β swaps	LBP	TRP
Objects	65.71%	65.71%	65.71%	65.71%	65.71%	65.71%
Objects+context	68.37%	68.60%	68.99%	66.72%	**71.45%**	71.16%
Mean ex. time	**0.46**	2.92	7.78	37.39	2.16	11.05
Max ex. time	**4.85**	26.30	26.73	181.26	10.80	130.67

Table 1 shows the results yielded by the different inference methods. Note that all the methods yielded the same outcome when considering *only* the features of the objects themselves. In this case, only nodes are added to the CRF graph structure, and all the methods chose the class assignation that maximizes the unary factor for each node (recall equation 3). On the other hand, when contextual information is considered, the performance increases in all of the cases. It can be seen how the outcome of the Greedy method is slightly better than the ICM one, and similar to the α-expansions, being the α-β swaps the method with *worse* results. In contrast, LBP and TRP shows the better figures, improving the recognition results in more than a $\sim 5\%$ with respect to only using the objects' features (with no contextual information). Regarding the execution time consumed by these methods, the average ranges from the 0.46ms. of the ICM method up to the 37.39ms. of the α-β-swaps[3].

Despite of the results achieved by the inference methods in these tests, their general performance is affected by a number of factors, e.g. the features used to

[3] These figures were obtained using an Intel®Core™i5 3330 microprocessor at 3GHz and 8 GB DDR3 RAM memory at 1.6 GHz.

model the problem, the training method employed, or the domain at hand. Thus, for a different application, the performance of each individual method should be tested in order to employ the one giving the better results.

Regarding the time spent training the CRF models, its average over the 100 executions is 585.46 seconds. Notice that the training process has to be performed only once, and the resulting CRF model can then be used to recognize objects within any scene.

5 Conclusions and Future Work

This paper has presented the Undirected Probabilistic Graphical Models in C++ library (UPGMpp), a software library for dealing with the scene object recognition problem exploiting contextual information. A description of the main software packages of UPGMpp has been detailed, with especial emphasis on the implemented probabilistic inference algorithms, giving a practical idea about the library features and capabilities. This work also contributes with the application of the UPGMpp library to a use-case to both: train CRF models, and obtain object recognition results through the execution of a number of inference processes. The challenging NYU2 dataset is used to train and test the CRF models within the use-case, proving the virtues of the library, which is publicly available under a GNU General Public License at `http://mapir.isa.uma.es/work/upgmpp-library`.

Some additional features regarding the performance of the UPGMpp library are currently under work. For example, some parts could greatly reduce their execution time with the utilization of multi-core parallelization mechanisms, like OpenMP. Support for GPUs using CUDA and/or OpenCL could be also advantageous in that sense. We also plan to include visualization tools for PGM graphs, as well as sampling techniques to draw samples from the probability distribution defined by a PGM. We welcome any contribution to the UPGMpp library from the computer vision community.

Acknowledgements

This work has been funded by the Spanish grant program FPU-MICINN 2010 and the Spanish projects “TAROTH: New developments toward a robot at home” (Ref. DPI2011-25483) and “PROMOVE: Advances in mobile robotics for promoting independent life of elders” (Ref. DPI2014-55826-R).

References

1. Anand, A., Koppula, H.S., Joachims, T., Saxena, A.: Contextually guided semantic labeling and search for three-dimensional point clouds. In the International Journal of Robotics Research 32(1), 19–34 (Jan 2013)
2. Besag, J.: On the statistical analysis of dirty pictures. Journal of the Royal Statistical Society. Series B (Methodological) 48(3), 259–302 (1986)

3. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. Pattern Analysis and Machine Intelligence, IEEE Transactions on 23(11), 1222–1239 (Nov 2001)
4. D.M. Greig, B.P., Seheult, A.: Exact maximum a posteriori estimation for binary images. J. of the Royal Statistical Society. Series B 51, 271–279 (1989)
5. Galleguillos, C., Belongie, S.: Context based object categorization: A critical survey. Computer Vision and Image Understanding 114(6), 712–722 (Jun 2010)
6. Guennebaud, G., Jacob, B., et al.: Eigen v3. http://eigen.tuxfamily.org (2010)
7. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
8. Lin, D., Fidler, S., Urtasun, R.: Holistic scene understanding for 3d object detection with rgbd cameras. IEEE Int. Conf. on Computer Vision 0, 1417–1424 (2013)
9. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: An empirical study. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence. pp. 467–475. UAI'99 (1999)
10. N. Okazaki, J.N.: libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). `http://www.chokkan.org/software/liblbfgs/` (2015), [Online; accessed 20-April-2015]
11. Okazaki, N.: Crfsuite: a fast implementation of conditional random fields (crfs), `http://www.chokkan.org/software/crfsuite/`, [Online; accessed 28-April-2015]
12. Ren, X., Bo, L., Fox, D.: Rgb-(d) scene labeling: Features and algorithms. In: IEEE Conf. on Computer Vision and Pattern Recognition. pp. 2759–2766 (2012)
13. Ruiz-Sarmiento, J.R., Galindo, C., González-Jiménez, J.: Mobile robot object recognition through the synergy of probabilistic graphical models and semantic knowledge. In: European Conf. on Art. Int. Workshop on Cognitive Robotics (2014)
14. Ruiz-Sarmiento, J.R., Galindo, C., González-Jiménez, J.: Exploiting semantic knowledge for robot object recognition. In: Knowledge-Based Systems (2015)
15. Ruiz-Sarmiento, J.R., Galindo, C., González-Jiménez, J.: OLT: A Toolkit for Object Labeling Applied to Robotic RGB-D Datasets. In: European Conference on Mobile Robots (ECMR) (2015)
16. Ruiz-Sarmiento, J.R., Galindo, C., González-Jiménez, J.: Scene Object Recognition for Mobile Robots through Semantic Knowledge and Probabilistic Graphical Models (2015), submitted
17. Schling, B.: The Boost C++ Libraries. XML Press (2011)
18. Schmidt, M.: UGM. `http://www.cs.ubc.ca/~schmidtm/Software/UGM.html` (2015), [Online; accessed 28-April-2015]
19. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor Segmentation and Support Inference from RGBD Images. In: Proc. of the 12th European Conference on Computer Vision (ECCV 2012). pp. 746–760 (2012)
20. Valentin, J., Sengupta, S., Warrell, J., Shahrokni, A., Torr, P.: Mesh based semantic modelling for indoor and outdoor scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2013). pp. 2067–2074 (2013)
21. Wainwright, M., Jaakkola, T., Willsky, A.: Tree-based reparameterization framework for analysis of sum-product and related algorithms. IEEE Transactions on Information Theory 49(5), 1120–1146 (May 2003)
22. Weiss, Y., Freeman, W.T.: On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. IEEE Trans. Inf. Theor. 47(2), 736–744 (Sep 2006)
23. Xiong, X., Huber, D.: Using context to create semantic 3d models of indoor environments. In: In Proceedings of the British Machine Vision Conference (BMVC 2010). pp. 45.1–11 (2010)