Propagation Kernels for Partially Labeled Graphs

Marion Neumann MARION.NEUMANN@IAIS.FRAUNHOFER.DE
Knowledge Discovery Department, Fraunhofer IAIS,
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Roman Garnett RGARNETT@CS.CMU.EDU
Robotics Institute, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213, United States

Plinio Moreno PLINIO@ISR.IST.UTL.PT
Electrical & Computer Engineering Department, Instituto Superior Técnico,
Av. Rovisco Pais, 1049-001 Lisboa, Portugal

Novi Patricia, Kristian Kersting FIRSTNAME.LASTNAME@IAIS.FRAUNHOFER.DE
Knowledge Discovery Department, Fraunhofer IAIS,
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

Learning from complex data is becoming increasingly important, and graph kernels have recently evolved into a rapidly developing branch of learning on structured data. However, previously proposed kernels rely on having discrete node label information. Propagation kernels leverage the power of continuous node label distributions as graph features and hence, enhance traditional graph kernels to efficiently handle partially labeled graphs in a principled manner. Utilizing locality-sensitive hashing, propagation kernels are able to outperform state-of-the-art graph kernels in terms of runtime without loss in prediction accuracy. This paper investigates the power of propagation kernels to classify partially labeled images and to tackle the challenging problem of retrieving similar object views in robotic grasping.

1. Introduction

For attribute-valued data, sophisticated kernel approaches have been widely and successfully studied. Nowadays, however, data often is complex and highly structured. Structured data is commonly represented by graphs, which capture relations among entities, but also naturally model the structure of whole objects. Real-world examples are proteins or molecules in bioinformatics, image scenes in computer vision, text documents in natural language processing, and object and scene models in robotics. Learning in such domains and in turn developing meaningful kernels to take the structure of these data into account is becoming more and more important. For example, one of the basic skills for a robot autonomous grasping is to select the appropriate grasping point for an object. Learning those grasping points from different types of features extracted from a single image or from more complex 3D reconstructions is an important and challenging problem in vision based robotic grasping (Montesano & Lopes, 2012; Moreno et al., 2011; Bohg & Kragic, 2010). In this paper, we tackle the task of retrieving similar views of objects represented as graphs derived from 3D point clouds, cf. Figure 1 for an example, in order to efficiently learn grasping points for unseen object views.

Determining whether or not a point is *graspable* involves setting up time-consuming and expensive experiments on a real robot. Hence, it is costly or even impossible to acquire complete label information. Nevertheless, matching regions of graspable points from similar (partially labeled) known objects in a given database to a new, unseen object view is a compelling

Appearing in *Proceedings of the Workshop on Mining and Learning with Graphs (MLG-2012)*, Edinburgh, Scotland, UK, 2012.

idea in order to efficiently learn where to grasp an object. The first step in tackling the complex task of learning robotic grasping is determining the similarity among objects represented as partially labeled graphs by graph kernels. Existing graph kernels developed within the graph mining community can be categorized mainly into four classes: graph kernels based on walks (Gärtner et al., 2003; Vishwanathan et al., 2010) and paths (Borgwardt & Kriegel, 2005), graph kernels based on limited-size subgraphs (Shervashidze et al., 2009), graph kernels based on subtree patterns (Ramon & Gärtner, 2003), and graph kernels based on structure propagation (Shervashidze et al., 2011). Whereas efficient kernel computations such as presented in (Vishwanathan et al., 2010) are able to compare unlabeled graphs efficiently, Shervashidze *et al.* (Shervashidze et al., 2011) specifically consider efficient comparisons of large, labeled graphs. Unfortunately, these existing graph kernels are either not computationally feasible for online analysis or rely on unlabeled or fully labeled graphs. As most of them can only handle discretely labeled, unweighted graphs in an efficient and principled manner, we introduce *propagation kernels* (Neumann et al., 2012) which leverage the power of continuous node-level features derived from propagated label information, i.e. node label distributions, in the graphs and naturally handle partially labeled and weighted graphs as the ones modeling objects to be grasped by a robot.

Triggered by previously introduced kernels on probabilistic models (Jaakkola & Haussler, 1998; Tsuda et al., 2002), propagation kernels exploit distributions from propagation schemes like label propagation (LP) and in general enhance existing graph kernel frameworks to handle continuous, vector-valued node attributes. In order to efficiently retrieve the similarity among node label distributions propagation kernels leverage randomization techniques from the theoretical computer science community by defining locality-sensitive hash (LSH) functions to create distance-preserving signatures for each continuous node label distribution.

To summarize, the main contribution of this work is the investigation of the power of propagation kernels to efficiently deal with partially labeled graphs. Specifically, we consider two problems on real-world data: semantic image classification and similar object retrieval in robotic grasping.

We proceed as follows. After reviewing the family of propagation kernels, we will briefly describe locality-sensitive hashing for handling vector-valued node label distributions. Then, we present experimental results for a graph classification task for partially labeled graphs on state-of-the-art image datasets. Before concluding, we examine the robustness of propagation kernels with respect to missing labels and the most similar retrieved object views in a robotic grasping scenario.

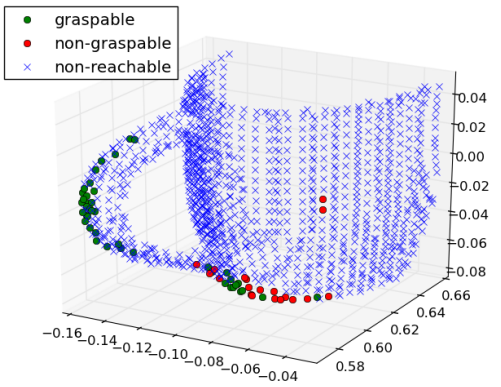


Figure 1. **3D Point Cloud Data** The figure shows an example object view of a cup with 1230 data points with 43 graspable, 19 non-graspable and 1168 non-reachable points. From the raw point cloud data of each object view a k-minimum spanning tree graph is constructed with one node per 3D point. The labels for each node are either graspable, non-graspable or non-reachable.

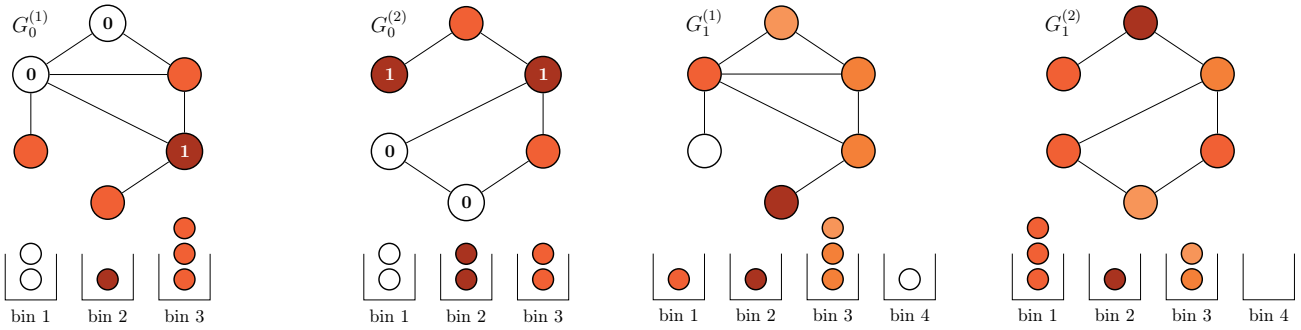
2. Propagation Kernels

In the following, we review the family of *propagation kernels* and several instances thereof based on propagating label information. The main insight that empowers propagation kernels is that the intermediate node label distributions, e.g., the iterative distribution updates of a label propagation scheme, capture both label *and* structure information of the graphs. Hence, the graph features, i.e. kernel inputs, are based on the counts of similar distributions among the respective graphs' nodes during these updates.

2.1. General Definition

Here we will define a similarity measure (specifically a positive semidefinite covariance function) $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ among graph instances $G^{(i)} \in \mathcal{X}$. The input space $\mathcal{X}$ is a family of graphs $G^{(i)} = (V^{(i)}, E^{(i)}, L^{(i)})$, where V_i is a set of nodes, $L^{(i)}$ are the corresponding node label distributions,[1] and $E^{(i)}$ is the set of edges in graph $G^{(i)}$. Further, graph i has n_i nodes and each node in $V^{(i)}$ is endowed with one of k true labels Specifically, these node labels do not have to be known for all nodes. Propagation kernels can naturally be computed

[1]Note that $L^{(i)}$ could also represent continuous, vector-valued node attributes, however, in this paper we focus on label distributions.



(a) Initial label distributions and linear base kernel value for $t = 0$

(b) Updated label distributions and linear base kernel value for $t = 1$

Figure 2. **Propagation Kernel Computations** Propagation kernel computations for two graphs $G_t^{(1)}$ and $G_t^{(2)}$ with binary node labels using one iteration of label propagation, Eq.(8), as distribution update. Node label distributions are decoded by color, white means $\ell_{0,j}^{(i)} = [1, 0]$ and dark red stands for $\ell_{0,j}^{(i)} = [0, 1]$, the initial distributions for unlabeled nodes (light red) are $\ell_{0,j}^{(i)} = [1/2, 1/2]$. Panel (a) shows the initial distributions, bins, and respective kernel computation and panel (b) depicts distributions, bins, features and linear base kernel for $t = 1$.

for partially labeled graphs as the features are only built upon the node label distributions, which will be initialized uniformly for unknown node labels. Note that, observed node labels are represented by a trivial delta distribution.

Propagation kernels are defined by applying the following iterative procedure $T + 1$ times, from time $t = 0$ to T, beginning with an initial set of graphs $\{G_0^{(i)}\}$ with label distribution initialized as above.

Step 1: count common node label distributions. First, we generate feature vectors $\phi(G_t^{(i)})$ for each graph by counting common label distributions induced over the nodes among the respective graphs. Therefore, each node in each graph is placed into one of a number of "bins," each one collecting similar label distributions, and these vectors count the nodes in each bin for each graph. The exact details of this procedure are given below, in Section 2.2.

Step 2: calculate current kernel contribution. Given these vectors, for each pair of graphs $G^{(i)}$ and $G^{(j)}$, we calculate

$$k(G_t^{(i)}, G_t^{(j)}) = \langle \phi(G_t^{(i)}), \phi(G_t^{(j)}) \rangle, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is an arbitrary base kernel. This value will be an additive contribution to the final kernel value between these graphs. Hence, the T-iteration propagation kernel between two graphs $G^{(i)}$ and $G^{(j)}$ is defined as

$$K_T(G^{(i)}, G^{(j)}) = \sum_{t=0}^{T} k(G_t^{(i)}, G_t^{(j)}). \tag{2}$$

Step 3: propagate node label distributions. Finally, we apply an iterative update scheme for the node label distributions

$$L_t^{(i)} \to L_{t+1}^{(i)}, \tag{3}$$

e.g. label propagation. These new label distributions replace those in the current set of graphs, and we continue with Step 1. The exact choice for this update results in different propagation kernels; examples are provided in Section 2.4. An illustrative example of propagation kernel computations for $t = 0$ and $t = 1$ for two graphs is shown in Figure 2.

2.2. Distribution-based Graph Features

The main ingredient of propagation kernels is the way distribution-based graph features are generated. Let $\ell_{t,j}^{(i)}$ be the j-th row of $L_t^{(i)}$ and $\mathcal{L} = \bigcup_i^N \bigcup_j^{n_i} \{\ell_{t,j}^{(i)}\}$ be the set of all uniquely occurring label distributions on the nodes of all graphs. The family of propagation kernels is characterized by generating graph features by counting similar node label distributions of the respective graphs. This will be captured by a function f mapping from the space of distributions $\mathbb{R}^k$ into the space of standard basis vectors $E_{k'} = \{e_1, ..., e_{k'}\}$ with

$k' = |\mathcal{L}| \leq n$, where n is the number of nodes for all graphs $n = \sum_{i=1}^{N} n_i$. We now define

$$\phi(G_t^{(i)}) = \sum_{j=1}^{n_i} f(\ell_{t,j}^{(i)}). \tag{4}$$

As the node label distributions $\ell_{t,j}^{(i)}$ are $k-$dimensional *continuous* vectors the cardinality of $\mathcal{L}$ might in fact be equal to the total number of nodes n in the whole graph database. This, however, means that the derived features are not meaningful as, in this case, we never get the same count feature for any two similarly distributed nodes and the kernel value for any two graphs as defined in Eq. (1) is always zero. To ensure the acquisition of meaningful features we leverage *quantization* (Gersho & Gray, 1991). Hence, the mapping f is replaced by $q\colon \mathbb{R}^k \to E_{k''}$, where q is a quantization function such that $k'' \ll |\mathcal{L}| \leq n$. Note, that deriving a quantization function for distributions involves considering distance metrics for distributions such as Hellinger or total variation (TV) distance. Below, we approach the quantization by defining locality-sensitive hash functions with respect to these metrics.

2.3. Locality-Sensitive Hashing for Propagation Kernels

Now, we briefly describe the quantization approach for implementing propagation kernels on graphs with node label distributions. The key insight here is, that the used quantization functions should "probably" assign points "close enough" to each other in a given metric space to the same bin. This approach is known as locality-sensitive hashing (Datar & Indyk, 2004). Each node-label vector is considered as an element of the space of discrete probability distributions on k items equipped with an appropriate probability metric.

Let $\mathcal{X}$ be a metric space with metric $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and let $\mathcal{Y} = \{1, 2, \ldots, k'\}$. Let $\theta > 0$ be a threshold, $c > 1$ be an approximation factor, and $p_1, p_2 \in (0, 1)$ be the given success probabilities. A set of functions $\mathcal{H}$ from $\mathcal{X}$ to $\mathcal{Y}$ is called a $(\theta, c\theta, p_1, p_2)$-*locality sensitive hash* (LSH) if for any function $h \in \mathcal{H}$ chosen uniformly at random, and for any two points $x, x' \in \mathcal{X}$, we have that

- if $d(x, x') < \theta$, then $\Pr(h(x) = h(x')) > p_1$, and
- if $d(x, x') > c\theta$, then $\Pr(h(x) = h(x')) < p_2$.

It is known that we can construct LSH families for L^p spaces with $p \in (0, 2]$ (Datar & Indyk, 2004). Let V be a real-valued random variable. V is called p-stable if for any $\{x_1, x_2, \ldots, x_d\}$, $x_i \in \mathbb{R}$ and independently sampled $v_1, v_2, \ldots, v_d$, we have

$$\sum x_i v_i \sim \|x_i\|_p V.$$

Explicit p-stable distributions are known for some p; for example, the standard Cauchy distribution is 1-stable, and the standard normal distribution is 2-stable. Given the ability to sample from a p-stable distribution V, we may define a LSH $\mathcal{H}$ on $\mathbb{R}^d$ with the L^p metric (Datar & Indyk, 2004). An element h of $\mathcal{H}$ is specified by three parameters: a width $w \in \mathbb{R}^+$, a d-dimensional vector $\mathbf{v}$ whose entries are independent samples of V, and $b \in [0, w]$, drawn from $\mathcal{U}[0, w]$, and defined as

$$h(\mathbf{x}; w, \mathbf{v}, b) = \left\lfloor \frac{\mathbf{v}^\top \mathbf{x} + b}{w} \right\rfloor. \tag{5}$$

We may now consider $h(\cdot)$ to be a function mapping our label distributions to integer-valued bins, where similar distributions end up in the same bin. If we number the non-empty integer bins occupied by all the nodes in all graphs from 1 to k'', then we may define the function f in Eq. (4) by $f(\cdot) = u \circ h(\cdot)$, where $u : \mathbb{N} \to E_{k''}$ and k'' denotes the number of non-empty bins.

Note, that we may use more than one hyperplane to decrease the probability of collision , i.e., more than one random vector $\mathbf{v}$ is chosen and the hash maps to more than one integer. For propagation kernels we only choose one hyperplane, as we effectively have T hyperplanes for the whole kernel computation. Hence, the probability of a hash conflict is reduced over the iterations.

As we are concerned with the space of discrete probability distributions on k elements, endowed with a probability metric d, we specifically consider the *total variation* (TV) and *Hellinger* (H) distances:

$$d_{\mathrm{TV}}(p, q) = {}^1\!/_2 \sum_i |p_i - q_i|$$

$$d_{\mathrm{H}}(p, q) = \left({}^1\!/_2 \sum_i \left(\sqrt{p_i} - \sqrt{q_i}\right)^2 \right)^{1/2}.$$

The total variation distance is simply half the L^1 metric, and the Hellinger distance is also a scaled version of the L^2 metric after applying the map $p \mapsto \sqrt{p}$. We may therefore create a locality-sensitive hash family for d_{TV} by direct application of (5), and create a locality-sensitive hash family for d_{H} by applying (5) after applying the square root map to our label distributions. These are the quantization schemes applied in our experiments.

Table 1. Average accuracy (and standard deviation) on 10 different sets of partially labeled images for *label propagation kernel* using TV distance ($K_{\text{LP}+\text{TV}}$), and for the *WL-subtree kernel* with unlabeled nodes treated as additional label K_{WL} and with hard labels derived from converged LP ($\text{LP} + K_{\text{WL}}$).

		labels missing			
dataset	method	20%	40%	60%	80%
MSRC9	$K_{\text{LP}+\text{TV}}$	**90.0** (1.2)	**88.7** (1.0)	**86.6** (1.3)	**80.4** (1.8)
	$\text{LP} + K_{\text{WL}}$	**90.0** (0.6)	87.9 (1.9)	83.2 (2.0)	77.9 (3.1)
	K_{WL}	89.2 (1.5)	88.1 (1.5)	85.7 (1.9)	78.5 (2.7)
MSRC21	$K_{\text{LP}+\text{TV}}$	**86.9** (0.8)	**84.7** (1.0)	**79.5** (0.9)	**69.3** (1.1)
	$\text{LP} + K_{\text{WL}}$	85.8 (0.6)	81.5 (0.8)	74.5 (1.0)	64.0 (1.2)
	K_{WL}	85.4 (1.3)	81.9 (1.2)	76.0 (0.8)	63.7 (1.3)

2.4. Instances of Propagation Kernels

So far, we introduced the general family of propagation kernels. Specific choices of label update schemes, cf. Eq. (3), result in different instances of the propagation kernel family. In particular, we define the *diffusion graph kernel* and the *label propagation kernel*.

Diffusion Graph Kernel: For the diffusion graph kernels we use the following update for the node label distributions $L_t^{(i)} \rightarrow L_{t+1}^{(i)}$. Given the adjacency matrix $A^{(i)}$ of graph $G^{(i)}$ label diffusion on each node is defined as

$$L_{t+1}^{(i)} \leftarrow T^{(i)} L_t^{(i)}, \tag{6}$$

where $T^{(i)}$ is the transition matrix, i.e., the row-normalized adjacency matrix $T^{(i)} = (D^{(i)})^{-1} A^{(i)}$, where $D^{(i)}$ is the diagonal degree matrix with $D_{aa}^{(i)} = \sum_b A_{ab}^{(i)}$.

Label Propagation Kernel: The label distribution update for the label propagation kernel differs in the fact, that before each iteration of label diffusion the labels of the originally labeled nodes are *pushed back* (Zhu & Ghahramani, 2002). Let

$$L_0^{(i)} = \left[L_{0,[labeled]}^{(i)}, L_{0,[unlabeled]}^{(i)} \right]^\top \tag{7}$$

be the original labels of graph $G^{(i)}$, where the distributions in $L_{0,[labeled]}^{(i)}$ represent hard labels and those in $L_{0,[unlabeled]}^{(i)}$ are initialized by a uniform label distribution. Then label propagation is defined by

$$\begin{aligned} L_{t,[labeled]}^{(i)} &\leftarrow L_{0,[labeled]}^{(i)}, \\ L_{t+1}^{(i)} &\leftarrow T^{(i)} L_t^{(i)}. \end{aligned} \tag{8}$$

3. Empirical Evaluation

Our intention here is twofold. First, we show results for partially labeled graphs for graph classification on semantic image datasets. Second, we introduce a graph representation for object views derived from 3D point clouds and investigate the power of propagation kernels for the task of retrieving similar graphs, i.e. object views, in a vision based robotic grasping scenario. To this aim, we implemented propagation kernels in Matlab. All experiments were conducted on an Apple Mac Pro workstation with two 2.26 GHz quad-core Intel Xeon "Gainestown" processors (model E5520) and 28 GB of RAM. We set the bin-width parameter $w = 10^{-5}$; all results were fairly insensitive to the exact choice of w.

3.1. Graph Classification

We compare classification accuracy and runtime for several instances of propagation kernels: the *diffusion graph kernel* and the *label propagation kernel* with total variation distance and the WL-subtree kernel. We choose the WL-subtree kernel for comparisons as it is currently the most accurate and efficient graph kernel (Shervashidze et al., 2011).

3.1.1. Experimental Setup and Datasets

The classification performance is evaluated by running C-SVM classifications using libSVM,[2] in a 10-fold cross-validation setup. The real-world image datasets MSRC 9-class and MSRC 21-class[3] are state-of-the-art datasets in semantic image processing. For our experiments we derived two datasets MSRC9 and MSRC21. Each image is represented by a conditional Markov

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[3] http://research.microsoft.com/en-us/projects/ObjectClassRecognition/

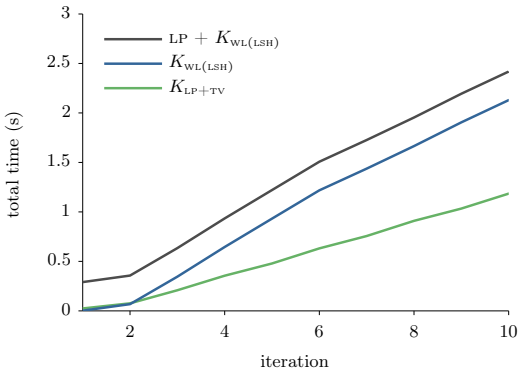


Figure 3. **Runtime for Partially Labeled MSRC21** Average time in seconds over 10 different instances of the MSRC21 dataset with 50% labeled nodes for kernel iterations T from 0 to 10. We compare the *WL-subtree kernel* with unlabeled nodes treated as additional label ($K_{WL(LSH)}$), and with hard labels derived from converged LP distributions (LP + $K_{WL(LSH)}$), and the *label propagation kernel* with TV distance (K_{LP+TV}).

random field graph. The nodes of each graph are derived by oversegmenting the images using the quick shift algorithm[4] with an average of 40 superpixels per graph. Hence, each node represents one superpixel and the semantic (ground-truth) node labels are derived by taking the mode ground-truth label of all pixels in the corresponding segment. Note, that the number of nodes varies from graph to graph.

3.1.2. Results

To assess the predictive performance of propagation kernels on partially labeled graphs, we ran the following experiments 10 times. We randomly removed 20–80% of the labels in MSRC9 and MSRC21 and computed cross-validation accuracies and standard deviations. Because the WL-subtree kernel was not designed for partially labeled graphs, we compare the *label propagation kernel* to two variants: one where we treat unlabeled nodes as an additional label "u" (K_{WL}) and another where we use hard labels derived from running label propagation until convergence (LP + K_{WL}). The results are shown in Table 1. For larger fractions of missing labels K_{LP+TV} obviously outperforms the baseline methods.

We also compared the runtime of propagation kernels using label propagation to the WL-subtree kernel K_{WL} on the MSRC21 dataset with partially labeled graphs. We again compare K_{LP+TV} with K_{WL} and LP + K_{WL}, where we implemented two variants of K_{WL}, $K_{WL(LSH)}$ and $K_{WL(REF)}$, where the former is a structure propagation kernel implementation of the WL-subtree kernel leveraging LSH (Neumann et al., 2012) and $K_{WL(REF)}$ is the original implementation introduced in (Shervashidze et al., 2011). The results are summarized in Figure 3. Note that $K_{WL(REF)}$ is over 36 times slower than K_{LP+TV} and omitted in the figure. These results clearly indicate that propagation kernels have attractive scalability properties for large datasets.

[4] http://www.vlfeat.org/overview/quickshift.html

3.2. Robotic Grasping

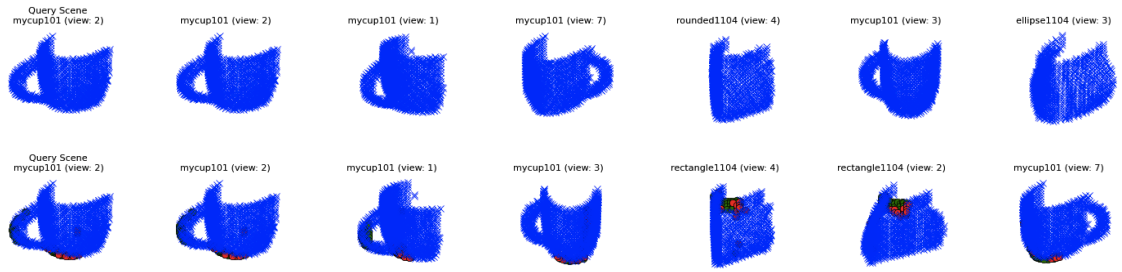
Now, we investigate the power of propagation kernels for the task of retrieving similar object views in robotic grasping, where object views are modeled by partially labeled graphs. We visually examine the most similar object views retrieved by our kernel and compare the robustness of the *diffusion graph kernel* and the WL-subtree kernel w.r.t. missing labels.

3.2.1. Graph Representation of Object Views

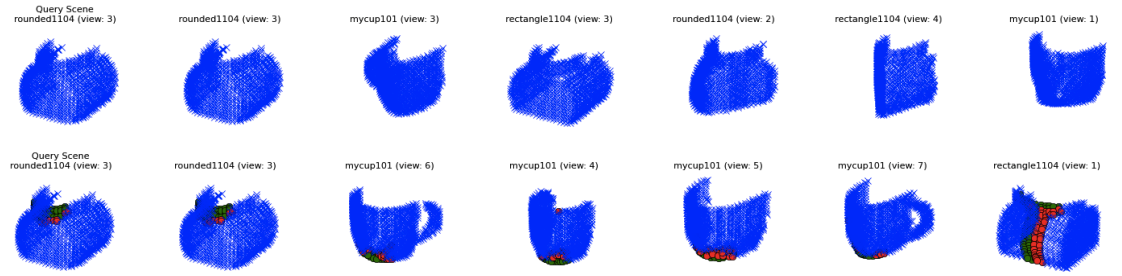
We consider laser range data of objects for robot grasping (Moreno et al., 2011), see Figure 1 for an example. The dataset consists of 8 objects, as for instance different cups and glasses. And the final task for the robot is to successfully grasp the objects. For each object we have between 1-8 views represented by 3D point clouds. From this data we derived 32 graphs, one per object view, by building the k-minimum spanning tree graph ($k = 5$) and assigning an edge weight reflecting the tangent plane orientations of its incident nodes. The weight for edge (i, j) between 2 nodes is given by $w_{i,j} = 1 - |\mathbf{n}_i \cdot \mathbf{n}_j|$, where $\mathbf{n}_i$ is the normal of point i. In total the data considered consists of 26 073 nodes. The nodes have 3 classes *graspable*, *non-graspable,* and *non-reachable*. Note, that every reachable point is either graspable or non-graspable. In the considered learning scenario, we only assume partial knowledge for node labels of reachable points, since the information whether a point is reachable can be easily gained by applying collision detection. Whereas, gathering information whether or not a point is *graspable* involves setting up time-consuming and expensive experiments on a real robot performing several trials per point in order to get reliable labels.

3.2.2. Retrieving Similar Objects

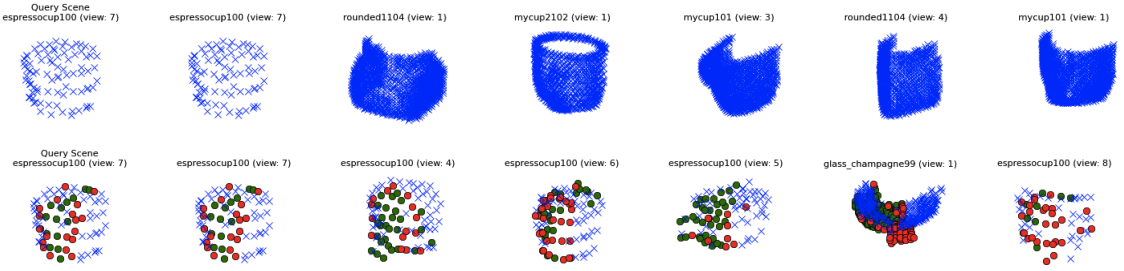
Figure 4 shows the top 6 retrieved object views for three different example query objects. All results were obtained by turning the *diffusion graph kernel* with Hellinger distance K_{DIFF+H} of height 4 into a correlation matrix. These results show, that propagation kernels on our graph representation of 3D point clouds give meaningful results and label information is indeed rel-



(a) Query object: MYCUP101 (VIEW 2)



(b) Query object: ROUNDED1104 (VIEW 3)



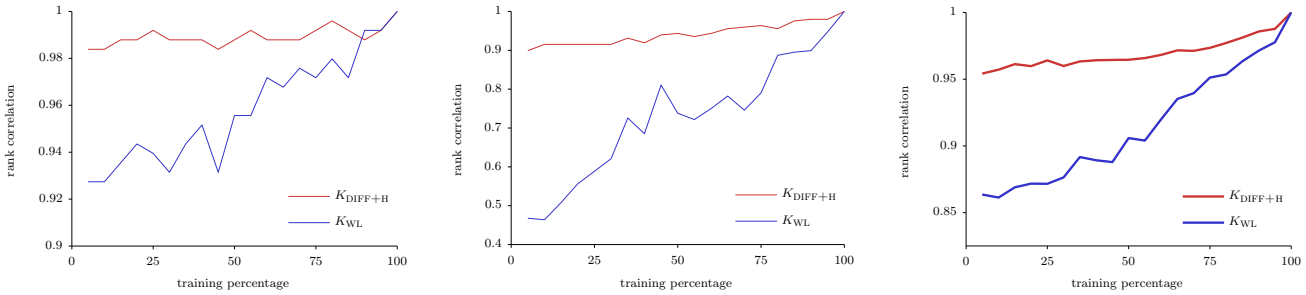
(c) Query object: ESPRESSOCUP100 (VIEW 7)

Figure 4. **Similar Objects** Query objects (first column) and top 6 most similar object views for unlabeled (first row) and fully labeled graphs (second row) applying the *diffusion graph kernel* with Hellinger distance $K_{\text{DIFF+H}}$ of height 4.

evant for the retrieval of similar objects. We also ran an experiment to test the robustness of kernels with respect to missing node labels. We first calculated K_{WL} and $K_{\text{DIFF+H}}$ on the fully labeled ground-truth graphs. We then removed (uniformly at random) from 10% to 90% of the labels of the reachable nodes (at 10% increments) and calculated the kernels again. For the K_{WL} kernel, we filled in missing labels with a placeholder. For each partially labeled dataset, we calculated the Spearman rank coefficient between the resulting kernel values for the respective query objects and the database versus the ground-truth kernel values. The hope is that this rank coefficient is high even with many missing labels. The results in Figure 5 indicate that our kernel is ordering the database objects correctly according to the kernel despite missing labels. Our kernel is robust across the entire range of missing values and exhaustively outperforms the WL-subtree kernel.

4. Conclusions and Future Work

Propagation kernels count common distributions induced in each iteration of running inference on the nodes of two graphs. They leverage locality-sensitive hashing to do so efficiently. This allows propagation kernels to efficiently deal with partially labeled graphs. Experimental results clearly show an improvement over state-of-the-art graph kernels in terms of quality and runtime. Further, we demonstrated that propagation kernels, can be utilized to augment the learning process of graspable points in vision based robotic grasping, The most exciting opportunity for future work is to examine the potential for *active learning* that has been opened up by our fast and robust kernel. Applications like the robot grasping problem could benefit greatly from actively selecting potentially graspable points, and our kernel seems to be promising for this task.



(a) ROUNDED1104 (VIEW 3) (b) ESPRESSOCUP100 (VIEW 7) (c) Average over all objects

Figure 5. **Rank correlation w.r.t. Partial Labels** The rank correlation between the ground-truth kernel values (using all labels) and the kernel values derived using partial node labels, for $K_{\text{DIFF+H}}$ and K_{WL}. Each plot shows the progression for a different object: (a) ROUNDED1104, (b) ESPRESSOCUP100, and (c) the average over all objects in the database.

Acknowledgments

We thank VisLab/IST Lisbon for the point cloud data. This work was partly supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract FP7-248258-First-MM.

References

Bohg, J. and Kragic, D. Learning grasping points with shape context. *Robotics and Autonomous Systems*, 58(4):362–377, 2010.

Borgwardt, K.M. and Kriegel, H.-P. Shortest-path kernels on graphs. In *Proc. of International Conference on Data Mining (ICDM-2005)*, pp. 74–81, 2005.

Datar, M. and Indyk, P. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG-2004)*, pp. 253–262, 2004.

Gärtner, T., Flach, P. A., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *Proc. of Computational Learning Theory and Kernel Machines (COLT-2003)*, pp. 129–143, 2003.

Gersho, A. and Gray, R. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991. ISBN 0-7923-9181-0.

Jaakkola, T. and Haussler, D. Exploiting generative models in discriminative classifiers. In *Proc. of Neural Information Processing Systems (NIPS-1998)*, pp. 487–493, 1998.

Montesano, L. and Lopes, M. Active learning of visual descriptors for grasping using non-parametric smoothed beta distributions. *Robotics and Autonomous Systems*, 60(3):452–462, 2012.

Moreno, P., Hornstein, J., and Santos-Victor, J. Learning to grasp from point clouds. Technical report, Vislab-TR001/2011, Dept. of Electrical and Computers Eng., Instituto Superior Técnico, 2011.

Neumann, M., Patricia, N., Garnett, R., and Kersting, K. Efficient Graph Kernels by Randomization. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD-2012)*, 2012. To appear.

Ramon, J. and Gärtner, T. Expressivity versus efficiency of graph kernels. In *Proceedings of the 1st International Workshop on Mining Graphs, Trees and Sequences*, pp. 65–74, 2003.

Shervashidze, N., Vishwanathan, S.V.N., Petri, T., Mehlhorn, K., and Borgwardt, K.M. Efficient graphlet kernels for large graph comparison. *Journal of Machine Learning Research - Proceedings Track*, 5:488–495, 2009.

Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., and Borgwardt, K.M. Weisfeiler–Lehman Graph Kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

Tsuda, K., Kawanabe, M., Rätsch, G., Sonnenburg, S., and Müller, K.-R. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10): 2397–2414, 2002.

Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R.I., and Borgwardt, K.M. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02-107, Carnegie Mellon University, 2002.