Efficient inference in random fields with pairwise potentials defined on distances

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

BIWI Computer Vision

Efficient inference in random fields with pairwise potentials defined on distances

Master's Thesis

Viviana Petrescu
Department of Computer Science

Advisor: Bogdan Alexe
Supervisor: Prof. Dr. Vittorio Ferrari
Computer Vision Laboratory
Department of Information Technology and Electrical Engineering

August 31, 2011

Acknowledgements

I am really thankful to my advisor, Bogdan Alexe, whose support and guidance from the beginning to the end of my thesis enabled me to develop an understanding of the subject.

Lastly, I offer my regards to all of those who helped me in any respect during the completion of the project.

Contents

1 Introduction **3**
- 1.1 Focus of this work . . . 3
- 1.2 Thesis Organization . . . 4

2 Background **5**
- 2.1 LOCalizing objects while LEARNing their appearance . . . 5
  - 2.1.1 The Conditional Random Field model in LocLearn . . . 5
  - 2.1.2 LocLearn Performance on Pascal07 . . . 9
- 2.2 Inference in Conditional Random Fields . . . 11
- 2.3 LocLearn complexity and limitations . . . 11

3 Divide and Conquer LocLearn **13**
- 3.1 Divide and conquer LocLearn . . . 13
- 3.2 Experiments . . . 14
  - 3.2.1 Properties of the selected windows of intermediate subproblems . . . 17
  - 3.2.2 corLoc results for Divide and Conquer LocLearn . . . 21
- 3.3 Discussion . . . 24

4 LocLearn with Binary encoding of GIST descriptor **27**
- 4.1 GIST . . . 27
- 4.2 LSBC encoding of GIST . . . 28
- 4.3 LocLearn with binary encoding of GIST feature . . . 29
- 4.4 Experiments . . . 30
- 4.5 Discussion . . . 32

5 LBP inference for sparse distance matrices **35**
- 5.1 Exploiting spatial overlap for computing sparse pairwise distance matrices . . . 35
  - 5.1.1 Theoretical upper bound on appearance distance of Color Histogram descriptors . . . 37
  - 5.1.2 Color Histogram bound experimental evaluation . . . 41
- 5.2 Sparse LocLearn . . . 43
- 5.3 Experiments . . . 45
- 5.4 Discussion . . . 48

6 Overview of hybrid LocLearn algorithms **49**

7 Conclusions **51**

List of Figures

2.1 Scheme of LocLearn algorithm 8

3.1 Divide and conquer LocLearn 15
3.2 100 windows per image 20
3.3 Divide and conquer LocLearn results 22
3.4 Distribution of corLoc for 100 LocLearn problems with different candidate windows. 24

4.1 corLoc for LocLearn Binary on meta-training data 31
4.2 Localization results of Binary GIST LocLearn with 256-bit encoded GIST 32

5.1 Spatial overlap of two windows 36
5.2 Bound on appearance distance using spatial overlap 37
5.3 Bound on appearance distance using spatial overlap 38
5.4 Maximum Theoretical Upper Bound $\mathcal{B}(o, r)$ for CHIST. 42
5.5 Maximum Theoretical Upper Bound $\mathcal{B}(o)$ for CHIST 43
5.6 Probability of selecting pairwise distances among the smallest $q\%$. 46

List of Tables

2.1 corLoc results of LocLearn on Pascal07 6x2 10

3.1 Hitrate and SNRCLS for the classes in Pascal07 6x2 with 10000 sample windows per image 16
3.2 Four setups of Divide and Conquer LocLearn 16
3.3 Histogram of hitrate for 100 LocLearn subproblems 18
3.4 SNR per class for 100 LocLearn subproblems 19
3.5 corLoc results of Divide and Conquer LocLearn on Pascal07 6x2 21
3.6 Accuracy of selected windows on Pascal07 6x2 23

4.1 coLoc of Binary GIST LocLearn on meta-training dataset Pascal07 6x2 31

5.1 Sparsity in pairwise distance matrices 46
5.2 corLoc results for Sparse LocLearn and Sparse Mean LocLearn 47

6.1 Summary of the hybrid LocLearn algorithms 49

Abstract

We propose three independent approaches that address the memory and time complexity limits of the method presented in [6]. This allows us to derive efficient inference algorithms for conditional random fields (CRF) with pairwise potentials defined on distances. [6] uses a fully connected CRF for jointly localizing a new object class for weekly supervised data by selection one window (label) per image (node) containing an instance of the new object class. Our first approach uses a Divide and Conquer strategy to reduce the quadratic complexity of the CRF in the number of windows. The second approach uses a Hamming embedder [13] to reduce the memory needed to store the descriptors. Finally, the third approach exploits the correlation between the spatial overlap of two windows and the appearance distance of the window descriptors [3] to speedup the computations of the pairwise potentials matrix and also to reduce the memory by computing sparse pairwise potentials matrices. The well known loopy-belief-propagation inference algorithm is modified such that it supports the sparse matrices input. In the experimental evaluations we show the first approach to outperform [6] as it can use 100x more windows and the second approach to obtain great memory savings for just a minor loss in performance. Although this work addresses primarly the limitations of LocLearn both theoretically and experimentally, the approaches we propose here can be employed in other tasks that deal with inference in Conditional Random Fields using pairwise potentials defined on distances.

Chapter 1

Introduction

1.1 Focus of this work

In supervised learning methods for object recognition tasks, one has to manually provide many images with bounding boxes of the target class for learning a model of that specific class. In order to reduce the labeling effort, in [7] is proposed a novel algorithm for jointly localizing and learning a new model in a weakly supervised manner. The algorithm will be referred to in the following sections as LocLearn. Given a set of images from a target class, the algorithm selects a window per image likely to cover target objects. As the paper [7] mentions it, LocLearn outperforms the state-of-the-art algorithms for the task defined above on the datasets Caltech4, Pascal06 and also on the more difficult dataset Pascal07.

In LocLearn, the selection of windows is done by solving an inference problem in a fully connected Conditional Random Field with the nodes represented by images and the labels represented by candidate windows. The selected windows are obtained by minimizing an energy function which consists of unary and pairwise potentials terms. The pairwise potentials are computed between every pair of windows coming from different images and express how likely it is for two windows to contain an object of the same class. LocLearn extracts different appearance descriptors for each window. The pairwise potential between two windows according to an appearance cue is computed as the squared Euclidean distance between their appearance descriptors. Storing all the descriptors and their appearance pairwise potentials requires a large amount of memory (in the order of Gb). Often this is more than what a normal personal computer can offer. Moreover, the computation of the pairwise potentials and their use in the inference problem makes LocLearn a computationally intensive algorithm.

The complexity of LocLearn is determined by the number N of images used, the number W of candidate windows per image and by the dimensionality of the feature descriptors that are extracted for each window. The time complexity is $O(N^2W^2)$ and the memory complexity is $O(NWD + N^2W^2)$ where D is the sum of the dimensions of each appearance descriptor used ($O(N^2W^2)$ for storing the pairwise potentials, $O(NWD)$ for storing the descriptors).

In the ideal case, LocLearn searches over all possible windows in an image in order to select the optimal window that covers an object. This is infeasible due to the complexity of the problem, and therefore the search space is restricted by sampling a small number of candidate windows for each image according to an objectness measure. However, the algorithm does not use more than 100 candidate windows per image and more than 50 images per class due to limited computational resources.

This master thesis addresses the problems that restrict the current implementation of LocLearn in using a larger number of windows and/or images. We propose three approaches which extend the given LocLearn

framework with the goal of overcoming its limitations in memory and time.

In the first part, a Divide and Conquer version of LocLean is proposed for increasing the number of sampled windows to be in the order of thousands, a setting which is infeasable in LocLearn. The aim is to reduce the quadratic complexity in the number of sampled windows W which appears when solving the inference problem.

Memory poses also some restrictions, since for each candidate window, four descriptors with dimensionality varying between 216 and 4000 are extracted and stored in a preprocessing step. In the second part, another method is proposed for reducing the required memory by binary encoding the most discriminant feature GIST [1] using a Hamming embedder.

In a third approach, the computation of pairwise potentials is speeded up by computing less distances between descriptors. This results in less memory needed for storing the pairwise potentials and thus allows the use of more candidate windows per image.

Although this work addresses primarly the limitations of LocLearn both theoretically and experimentally, the approaches we propose here can be employed in other tasks that deal with inference in Conditional Random Fields using pairwise potentials defined on distances.

1.2 Thesis Organization

The next section gives a broader description of LocLearn, its current performance on Pascal07 dataset and discusses its complexity and limitations. In the following sections, we explain in detail the different versions of the LocLearn algorithm that we propose, devoting a separate section for each method. In sec. 3 more candidate windows are used in a Divide and Conquer version of LocLearn, which reduces both the time and the memory complexity of LocLearn. In sec. 4 the most discriminant feature vector GIST is binary encoded, which reduces the memory requirements for storing the features of the candidate windows. In sec. 5 we employ an algorithm to compute fewer pairwise distances between window descriptors. This reduces the computational time and the storage memory of the pairwise potentials and permits the use of more candidate windows per image. The last section summarizes the implications of our approaches which are supported by the experimental evaluation of the three hybrid LocLearn algorithms.

Chapter 2

Background

2.1 LOCalizing objects while LEARNing their appearance

In supervised learning methods for object recognition tasks, classifiers are trained on a large collection of images that come with manually annotated bounding boxes. The labeling of images is time consuming and limits the applicability of these methods. Recently more effort has been put in learning a model for a class in a *weakly supervised* fashion. In this setup, for training a model, a set of images containing one or more objects of the specific class is given with no additional information on the location of the objects.

In [7] a Conditional Random Field (CRF) model is proposed for simultaneously LOCalizing and LEARNing (LocLearn) the appearance of a new class in a weakly supervised fashion. LocLearn uses *meta-training* data to learn first a model from generic classes. The meta-training data consists of images for which the ground-truth bounding boxes covering different object classes are provided. In a next step, the model uses the images (not annotated) from the *training data* in order to adapt to a new class in an iterative process, where localization and learning stages are repeated alternatively.

Given a set of images of a new class with candidate windows, LocLearn selects a configuration of windows that are likely to cover an object of the specific class. The *Pascal criterion* states that a window w *covers* or *correctly localizes* an object, if the overlap $\frac{|w \cap gt|}{|w \cup gt|} > 0.5$, where gt is the ground-truth bounding box.

2.1.1 The Conditional Random Field model in LocLearn

A Conditional Random Field (CRF) is a probabilistic graphical model for labeling sequence data, in which each node can have multiple labels. If an edge is present between every pair of nodes, the CRF is said to be *fully connected*. Given a random variable X which represents the observed sequence and a random variable L which represents the label sequence, the CRF relaxes the independence assumptions by modeling $p(L|X)$ directly, rather than a joint distribution $p(L, X)$ as in the case of Hidden Markov Models.

The maximum a posteriori probability $p(L|X)$ of labels L given observation X is approximated by $\exp(-E(L|X))$, where $E(L|X)$ is an energy function. The *inference problem* refers to selecting a label for each node such that the energy function is minimized. This is equivalent to returning the set of labels that have the largest probability given the observation.

In LocLearn, the nodes of the CRF are represented by the set of training images $I = (I_1, ..., I_N)$ and the labels represent candidate windows. For each node/image the model chooses a label among the candidate windows. Searching over all possible windows in an image is infeasible and a smaller number of windows

are sampled according to the objectness measure defined in [2], giving candidate windows which have a greater probability of containing an object of any class. Each window comes with an *objectness score* which quantifies how likely it is to cover an object of *any* class. The higher the score, the more likely it is that the window covers an object and not pure background.

Given a set of training images $I = (I_1, ..., I_n)$ and their candidate windows (sampled from the objectness measure), LocLearn selects the best configuration of windows $L = (l_1, l_2, .., l_N)$ by minimizing an energy function $E(L|I, \theta)$, where θ represents the parameters of the model.

The posterior probability of the selected windows is defined by:

$$p\left(L|I,\theta\right) \propto \exp(-E(L|I,\theta)) = \exp\left(-\sum_{n}\rho_n\phi(l_n|I_n,\theta) - \sum_{n,m}\rho_n\rho_m\phi(l_n,l_m|I_n,I_m,\theta)\right). \quad (2.1)$$

The optimal configuration of windows L has the smallest energy, which corresponds to the greatest MAP probability. The energy function consists of unary and pairwise potentials terms which assign a cost to the selected configuration. The unary potential $\phi(l_n|I_n, \theta)$ refers only to the properties of window l_n and it has a smaller value when it is more likely that l_n covers an object. The pairwise potential $\phi(l_n, l_m|I_n, I_m, \theta)$ expresses the similarity in appearance and shape of the windows l_n and l_m which come from different images.

Each component of the energy function is reviewed below:

- Weights
  - ρ_n represents the weight of image I_n; they are initially set to uniform and later are adapted in the learning stages.
  - α are the weights of each potential term (unary or pairwise) which are learned from meta-training data.
- Unary potentials:

$$\phi(l_n|I_n,\omega) = \alpha_\omega\omega(l_n|I_n,\theta_\omega) + \sum_f \alpha_{\Upsilon_f}\Upsilon_f(l_n|I_n,\theta_{\Upsilon_f}) \quad (2.2)$$

  - ω is 1 - the probability to have any object in the window. The probability is given by the objectness score [2].
  - Υ_f defines an appearance model for every cue f. It depends on the content of the image and adds class specific information to the model.
- Pairwise potentials:

$$\phi(l_n,l_m|I_n,I_m) = \alpha_\lambda\lambda(l_n,l_m,\theta_\lambda) + \sum_f \alpha_{\Gamma_f}\Gamma_f(l_n,l_m|I_n,I_m) \quad (2.3)$$

  - λ expresses the shape similarity of the two windows. The term is not dependent on the image content and relates to the aspect ratio of the windows.
  - Γ_f expresses the appearance similarity of two windows for each cue f.

Γ_f consists of pairwise potentials defined on Euclidean distances computed between every pair of windows coming from different images. It expresses appearance similarity and its computed as the sum of squared differences:

$$\Gamma_f(l_n, l_m|I_n, I_m) = ||l_n^f(I_n) - l_m^f(I_m)||_2^2, \tag{2.4}$$

where $l_n^f(I_n)$ and $l_m^f(I_m)$ are the descriptors extracted from windows l_n and l_m corresponding to cue f. Four cues are extracted for each candidate window to express its appearance:

- Spatial Envelope for describing the overall appearance of a window (GIST) [1], of dimension 960
- Color histogram in Lab space (CHIST) quantized in $10 \times 20 \times 20$ bins, of dimension 4000
- Bag of Words of Speeded up robust features (SURF) [4] of dimension 2000; a codebook of 2000 words was obtained using k-means
- Histogram of oriented gradients (HOG) [11], of dimension 216

The focus of this work is to make inference in the pairwise potentials Γ_f efficient. In sec. 4, $l_m^{GIST}(I_m)$ is binary encoded using a Hamming embbeder which reduces the storage requirements for the GIST descriptor. In sec. 5, less $\Upsilon_f(l_n, l_m|I_n, I_m)$ terms are calculated which speeds up the time needed for computing all pairwise potentials and allows the use of more candidate windows per image.

LocLearn is an iterative algorithm, where *localization* and *learning* are performed iteratively. In the localization stage, inference in the CRF model gives us a configuration of windows. In the learning stage, the selected configuration is used for adapting the CRF model. Fig. 2.1 illustrates the localization and learning stages in LocLearn, which uses generic class information from meta-training data.

The parameters $\theta_\omega, \theta_\lambda, \theta_{\Gamma_f}$ and the weights α of the potentials in the energy function are learned during the meta-training stage. They carry only information from generic classes and are used to train a model to be able to localize objects of any class. The generic class model is given as input in the first localization step and its parameters do not adapt to the new class in the learning stages. The image responsibilities ρ and the appearance unary potentials Υ_f carry information about the specific class and are updated at every training phase. A schematic view of the flow of the algorithm is shown in Fig. 2.1.

Localization With the energy function defined, the problem of *localizing* the objects in images reduces to finding the set of windows among the candidate windows which maximizes the probability:

$$L^* = \arg\max_L p\left(L|I, \theta\right) \tag{2.5}$$

This is equivalent with a MAP labeling of the images (nodes of the CRF) with the windows that correspond to a smaller energy. Intuitively, it is expected that the energy is minimized when the selected windows cover an object of the target class. A candidate window is considered to localize a target object if the Pascal criterion is met.

The parameters $\theta_\omega, \theta_\lambda, \theta_{\Gamma_f}$ and the weights α are learned from the meta-training data and do not change throughout the localization and learning stages. When localization is done for the first time, the image responsibilities ρ and the appearance unary potentials Υ_f are set to uniform.

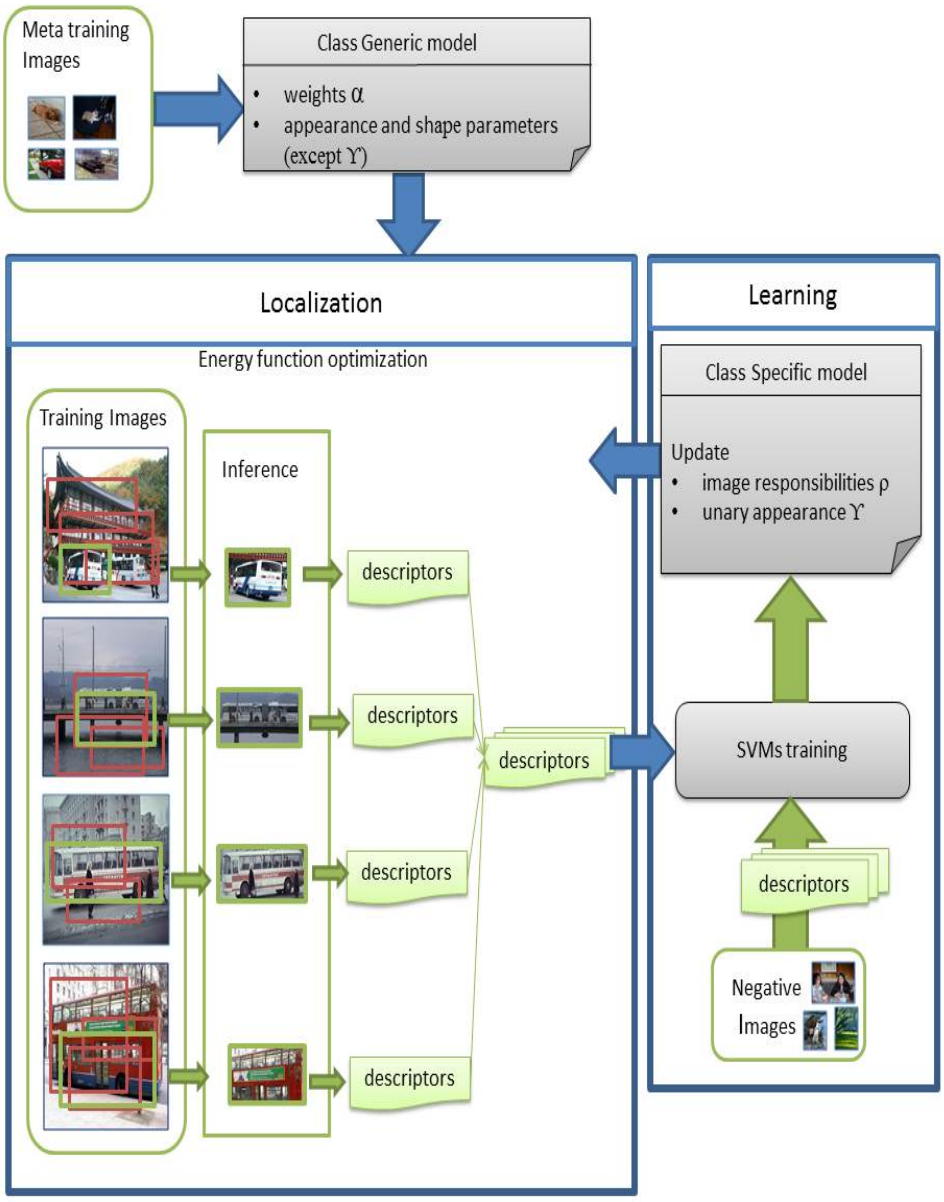


Figure 2.1: **Scheme of LocLearn algorithm.** *In a first stage, the weights α of the potentials and the parameters θ_ω, θ_λ and θ_Γ are learned from meta-training data. In the first localization step, the algorithm minimizes an energy function in order to select a configuration of windows which are likely to localize an object of any class. In the Learning stage, the corresponding appearance descriptors (for one or more cues) are used as positive training data for SVMs classifiers. Using the SVMs output, the model adapts some of its parameters to be specific for the target class. The energy function incorporates the class specific information and its optimization returns a new configuration of windows. The Localization and Learning stages are iteratively repeated.*

Learning The windows L^* selected in the localization stage are used for training a SVM classifier for each cue f. The negative set of images used in training does not contain objects of the target class. We train a SVM for cue f such that the descriptors for windows in L^* are considered positive training examples and the descriptors of windows from the negative are considered negative training examples. To make the algorithm more robust this is done multiple times, namely we use the windows from L^* which received a top k% score by the SVM to retrain the classifier. This gives more importance to windows from L^* which are more likely to cover a target object.

The class specific unary potential Υ_f for a window l_n is adapted and set to the signed distance between the SVM hyperplane and the appearance descriptor $l_n^f(I_n)$.

The image responsibilities ρ are updated such that the images which are likely to cover an object get more importance. The negative distance between a window descriptor for cue f and the hyperplane of the SVM classifier for f reflects the probability that the window comes from an image of the target class. The responsibility of an image I_n incorporates information from all cues and is set to be proportional to $\rho_n \sim \sum_f \alpha_f \Upsilon_f(l_n^*|I_n, \theta_\Gamma)$, where l_n^* is the selected window for I_n.

We note that the localization and learning stages depend on each other. A better configuration of selected windows in the localization stage leads to better SVM classifiers whose output is used for adapting the class specific model in the learning stage. In return, a better class specific model leads to better localization results.

2.1.2 LocLearn Performance on Pascal07

In this work we propose different hybrid versions of LocLearn which aim at improving the inference in CRF with respect to time or/and memory. We compare the performance of our approaches with the one of the original LocLearn framework proposed in [7]. LocLearn outperforms the state-of-the-art methods for learning a model of an unknown class in a weakly supervised manner. The performance of the algorithm is measured as the percentage of images in the training set for which the selected window correctly localizes the target object according to the Pascal criterion. This is reported as corLoc (correctly localized). Note that it can happen that an image contains more instances of objects of the target class. In this case, when computing corLoc, a selected window is considered good if it correctly localizes any of the target objects present in the image.

When the CRF model is fully adapted to the new class, LocLearn gives corLoc of 81%, 64% and 50% on the datasets Caltech4, Pascal06 6x2 and Pascal07 6x2. The best competitor [6] obtains corLoc of 55%, 45% and respectively 33% on the same datasets. LocLearn correctly localizes approximatively 43% more objects over the three datasets proving its robustness to a variety of images.

In this work, we evaluate the modified versions of LocLearn only on the Pascal07 6x2 dataset. This is a more challenging dataset which contains very difficult images. The appearance and shape characterisitcs of the objects vary a lot among images, which often depict cluttered scene. The best competitor [6] obtains a corLoc of only 33% on this dataset.

Pascal07 6x2 Meta-Training Data. The meta-training data for Pascal07 6x2 consists of a total of 799 images representing 24 class-viewpoint combinations from 6 classes (bird, car, cat, cow, dog, sheep). From the 24 classes we learn a generic class model which is able to localize objects of any class. The parameters of the generic class model are the pairwise potentials for appearance and shape, the parameters of the objectness unary potential and the weights of each term in the energy function 2.1. LocLearn uses the generic class model in the first localization step to select an initial configuration of windows for the images in the training data.

Pascal07 6x2 Training Data. The training data consists of 478 images coming from 6 classes (bus, boat, aeroplane, horse, motorbike, bicycle), each with two viewpoints, left and right. Each class-viewpoint combination has between 21 and 50 images. For each combination, the CRF model adapts the image responsibilities and the unary appearance potentials to incorporate class specific information.

We review in Table 2.1 the performance results of LocLearn on the Pascal07 6x2 training dataset as reported in [7]. In the last column we show the average corLoc over all 12 classes in Pascal07 6x2.

Table 2.1: corLoc results of LocLearn on Pascal07 6x2

Method	PASCAL07 6x2 CorLoc
LocLearn -localization only	
(a) random windows	0%
(b) no objectness score + single cue GIST	30%
(c) objectness score + single cue GIST	37%
(d) all cues	37%
LocLearn - localization and learning	
(e) objectness score + single cue GIST	40%
(f) full adaptation	50%

In all the setups (a)-(f), LocLearn uses 100 candidate windows per image and in setups (a)-(d) only the localization step is run.

- setup (a): the candidate windows are randomly sampled and none of the selected windows localize an object of the new class. This motivates the need for good candidate windows, especially since searching over all possible windows in an image is not feasible.

- setup (b): the candidate windows are sampled from the objectness function [2], but their objectness unary potentials are set to uniform. As appearance similarity cue only GIST feature is used and in almost one third of the images the windows localize an object (corLoc of 30%).

- setup (c) : this setup is similar to (b) but the appearance unary potentials are expressed by the objectness scores of the sampled windows. This increases the localization rates to 37%. At this point, LocLearn obtains better results than the best competitor [6] which on average localizes objects only in 33% of the images of a new class.

- setup (d) : setup (c) is extended to use all four cues for the appearance similarity of two windows. The performance of the algorithm does not improve from setup (c).

Some observations can be made related to the results of LocLearn (a)-(d). The performance of the algorithm depends on the characteristics of the sampled windows. If the windows are randomly sampled, LocLearn does not localize any objects. This suggests the need of an objectness function that gives us sample candidate windows which are more likely to contain an object. Furthermore, we can conclude that the GIST descriptor is very powerful in describing the holistic structure of a window and its use has a high impact in the first localization stage of LocLearn. We remind that in setups (a)-(d) LocLearn uses only class generic knowledge for localizing objects of a specific class.

In setups (e)-(f) localization and learning are done iteratively.

- setup (e) : the settings are the same as in experiment (c) but the CRF model is adapted to the specific class by alternating localization and learning stages. This increases corLoc from 37% (c) to 40% (e).

- setup (f) : the model from setup (d) is adapted to the specific class. corLoc increases from 37% (d) to 50% (f), which is a 36% improvement in performance. This shows that incorporating multiple cues helps the model in learning specific characteristics of a new class.

The best performance of LocLearn algorithm on the Pascal07 6x2 is obtained when localization and learning stages are run iteratively and when four cues are used in the class specific model. Adding multiple cues does not improve performance of localizing an object of any class, but they become important when we learn a class specific model, as the cues adapt to the particular characteristics of the class.

Throughout this work, we will refer to the setups (c), (d) for localization and setups (e), (f) for localization and learning in LocLearn. The three methods we propose in the following sections for adressing the limitatios of LocLearn add modifications to one of these setups and their corLoc performance will be compared to the corresponding corLoc from Table 2.1.

2.2 Inference in Conditional Random Fields

Inference in a fully connected CRF consists in selecting a label for each node of the CRF such that an energy function is minimized. Exact inference is a NP-hard problem and many approximation algorithms for energy minimization exist such as tree-reweighted-message passing (TRW-S) [9], Loopy Belief Propagation (LBP) [12] or graph cuts.

In the results from Table 2.1, LocLearn uses TRW-S for minimizing eq. 2.5. The algorithm TRW-S returns also a lower bound on the energy which can be used to determine the quality of the approximation. The authors of [7] mention that in their experiments, TRW-S proved to give very good approximative results for the energy function and that the global minimum of the energy was attained in 93% of the cases. A fast implementation in C++ is provided by the author of [9].

LBP does not always converge and can get stuck into loops, but proved to give good results on real world data in computer vision problems, such as stereo [5], [10] or image restoration [5]. Graph cuts give better results than LBP, but do not support more than 2 labels per node.

An exact inference has complexity $O(N^W)$, where N is the number of nodes and W is the number of labels or states in the CRF. In this work, we use only TRW-S and LBP which have a complexity of $O(iEW^2)$, where i is the number of iterations, E the number of edges and W the number of states. LocLearn uses a CRF in which the nodes are fully connected which means $E = N(N-1)/2$ edges are present. Thus, the approximation algorithms introduced above reduce the complexity from $O(N^W)$ to $O(N^2W^2)$, which is a significant improvement, but still not feasible if N or W are in the order of thousands.

2.3 LocLearn complexity and limitations

In LocLearn, selecting random candidates windows gives poor results (setup (a) in Table 2.1) and using all possible windows in an image is computational infeasible. This motivates the use of an objectness measure for sampling windows that are likely to contain an object. By sampling more windows, we expect to have more candidate windows which localize a target object and thus our model could possibly select a better configuration of windows.

Time complexity of LocLearn As mentioned in the previous section, minimizing the energy defined by eq. 2.5 can be done using an approximation algorithm in $O(N^2W^2)$, where N is the number of input images and W is the number of sampled windows per image. This makes inference computationally very intensive and even infeasible if more than 200 candidate windows are to be sampled per image. Another expensive step in LocLearn is the computation of appearance pairwise potentials defined on distances. A pairwise potential for a cue f is computed for every two windows coming from different images. There are $N(N-1)/2$ image pairs and each pair accounts for W^2 potentials, since there are W^2 window combinations. This means that the computation of all the pairwise potentials has complexity $O(N^2W^2)$.

In [7] the algorithm was run with at most 50 images per class and at most 100 windows per image. Running the problem with thousands of windows per image is infeasible in the normal settings of the algorithm due to its quadratic complexity in N and W. In sec. 3, we propose a method for reducing the time complexity $O(N^2W^2)$ in the number W of sampled windows by using a Divide and Conquer version of LocLearn.

Memory complexity of LocLearn Another limitation of the algorithm which impedes the use of many candidate windows per image is the memory needed for storing the potentials (unary and pairwise) and the descriptors. The setup of LocLearn algorithm that obtains the best performance on the Pascal07 6x2 dataset (corresponding to Table 2.1(f)), uses four descriptors: GIST, CHIST, SURF and HOG.

One candidate window requires 7176 (960+4000+2000+216) double elements for storing all its descriptors. Assuming a double requires 8 bytes as it is the case for the Matlab double numeric type, this means 7176 elem x 8 bytes = 57408 bytes $\sim$ 0.43Mb per window, which makes 43Mb for 100 windows and $\sim$17Gb for 400 images with 100 candidate windows per image (Pascal07 6x2 training dataset contains 478 images). Thus, LocLearn is limited not only because of the complexity of the inference problem and the computation of the pairwise potentials, but also because of the memory needed to store the descriptors of the windows. The memory storage complexity of the descriptors is $O(NWD)$, where D is the sum of the dimensions of the descriptors used, in our case in the order of 7000. In sec. 4 we propose a method for saving large amounts of storage requirements for the GIST features by binary encoding the descriptors using a Hamming embedder. This is a general approach, which can be applied to any type of descriptors.

The memory complexity of the unary potentials is $O(NW)$, since one unary potentials is stored for every candidate window of an image. As we mentioned previously, there are $O(N^2W^2)$ pairwise potentials. The total memory complexity for storing the potentials is $O(NW + \frac{N(N-1)}{2}W^2)$, which is dominated by the second term. For a class with N=50 images and W=100 candidate windows per image, this gives roughly 5.8Gb if the elements of the potentials are stored as doubles. Note that Pascal07 6x2 dataset consists of 12 classes and we run LocLearn also with 4 types of descriptors.

We propose in sec. 5 an approach for reducing the storage requirements of the pairwise appearance potentials by computing less pairwise distances. This allows us to use more candidate windows per image and also speeds up the time needed for their computation, which is $O(N^2W^2)$.

Chapter 3

Divide and Conquer LocLearn

3.1 Divide and conquer LocLearn

In the general setup of a Divide and Conquer algorithm, a problem is broken down recursively into more subproblems of the same type (or similar) until they are feasable or trivial to solve. The solutions of the subproblems are then combined to give a solution to the initial problem. We propose a *Divide and Conquer* approach such that the time complexity $O(N^2W^2)$ of LocLearn is reduced, allowing for more sampled windows per image. We will refer to this hybrid version of LocLearn as *D&C LocLearn*.

As it is the case with LocLearn, given a set of N images from a target class, D&C LocLearn samples W candidate windows per image from the objectness measure [2] and selects a configuration of windows which are likely to cover objects in the sense of the Pascal criterion. From each image we sample a large number W of windows, for which the problem cannot be solved in the normal settings of LocLearn due to its quadratic complexity in W. We can distinguish three major steps of the D&C LocLearn algorithm following a D&C paradigm:

- we break down the problem into subproblems that are smaller instances of the same problem.

  The windows are partioned into S disjoint subsets. Each of the S subsets is used as input for a LocLearn subproblem which uses $W_s = W/S$ candidate windows for each image.

- we solve the LocLearn subproblems.

  Each subproblem $i \in 1, 2..., S$ minimizes the energy in a fully connected CRF with the labels being the windows from the corresponding subset i.

- we combine the solutions of the subproblems to give a final solution

  The windows selected by TRW-S for each subproblem L^{sol_i}, $i \in 1, ..., S$ are combined to form the final LocLearn problem, which gives us the final configuration of windows L^f.

A mathematical formulation of D&C LocLearn is given in algorithm 1. Note that the final LocLearn subproblem can select the final windows from the set of windows obtained by applying multiple LocLearn, which should be a better set of windows than having one sample LocLearn.

The D&C algorithm is depicted in Fig. 3.1 for $N = 2$ images, $W = 10000$ sampled windows per image, $W_s = 100$ windows per subproblem and a total of $S = 100$ intermediate subproblems.

By breaking down our initial problem into multiple LocLearn subproblems, we reduce the time complexity of our algorithm from $O(W^2N^2)$ to $\frac{W}{W_s}O(W_s^2N^2) + O(W_s^2N^2)$, where W_s is the number of windows per image in a subproblem ($\frac{W}{W_s}$ intermediate problems with $O(W_s^2N^2)$, and a final subproblem with

Algorithm 1 DivideAndConquerLocLearn($(I_1, ..., I_N), N, W, W_s$)

Input: $(I_1, ..., I_N)$ = input images
N = number of images
W = total number of candidate windows sampled for one image
W_s = number of windows per image in a subproblem
Output: L^f

$S = \frac{W}{W_S}$ number of LocLearn subproblems
{sample candidate windows from objectness measure}
$l_1^{1:W}, ..., l_N^{1:W}$
$l_i^{sol_j}$ = the label selected in the i^{th} image by subproblem j
for $i = 1 \rightarrow S$ **do**
{solve intermediate subproblem i}
$L^{sol_i} = \left(l_1^{sol_i}, l_2^{sol_i}, ..., l_N^{sol_i}\right) \leftarrow LocLearn\left(l_1^{(i-1)\cdot W_s+1:i\cdot W_s}, l_2^{(i-1)\cdot W_s+1:i\cdot W_s}, ..., l_N^{(i-1)\cdot W_s+1:i\cdot W_s}\right)$
end for
{solve final subproblem}
$L^f = \left(l_1^f, l_2^f, ..., l_N^f\right) \leftarrow LocLearn\left(l_1^{sol_1,sol_2,...,sol_S}, l_2^{sol_1,sol_2,...,sol_S}, ..., l_N^{sol_1,sol_2,...,sol_S}\right)$
return L^f

$O(W_s^2N^2)$). If we keep only the dominant terms, the complexity can be expressed as $O(WW_sN^2)$. Furthermore, it can be noticed that the subproblems are independent and if they are run in parallel, D&C LocLearn algorithm can have a complexity of $O(W_s^2N^2)$. Notice that W_s is much smaller than W and reducing the complexity to $O(W_s^2N^2)$ from $O(W^2N^2)$, makes the problem feasible also for large W.

We note that D&C LocLearn has different memory requirements for storing the appearance pairwise potentials. Its memory complexity is given by $O(N^2W_s(W+W_s))$ (we have $\frac{W}{W_s}+1$ LocLearn subproblems each with a memory complexity of $O(N^2W_s^2)$). When $W_s < \frac{\sqrt{5}-1}{2}W$, the memory complexity of D&C LocLearn of $O(N^2W_s(W + W_s))$ is smaller than $O(N^2W^2)$.

The method we propose exploits the independence of the LocLearn subproblems to reduce the quadratic time complexity of LocLearn in W. This permits us to run experiments with a large number of W, which is infeasable in the normal settings of the LocLearn algorithm. For $W_s < \frac{\sqrt{5}-1}{2}W$ our new method also uses less memory for storing appearance pairwise potentials.

3.2 Experiments

We sample 10000 windows per image with the objectness measure [2], and precompute for each window four descriptors (GIST, CHIST, SURF, HOG). The *hitrate* of a set of images from a class is defined as the percentage of images for which it exists at least one sampled window that covers an object in the Pascal sense. The *Signal-to-Noise Ratio (SNR)* in an image is defined as the percentage of sampled windows which localize an object of the specific class. In the following section, we will report SNR per class (SNRCLS), which is the average SNR for the set of images from that class.

The performance of D&C LocLearn expressed as corLoc has an upper bound given by the hitrate of the sampled windows. The algorithm can not obtain better performance in terms of corLoc than the hitrate of the input set of images. In our case, the average hitrate over all 12 classes in the Pascal07 6x2 dataset with

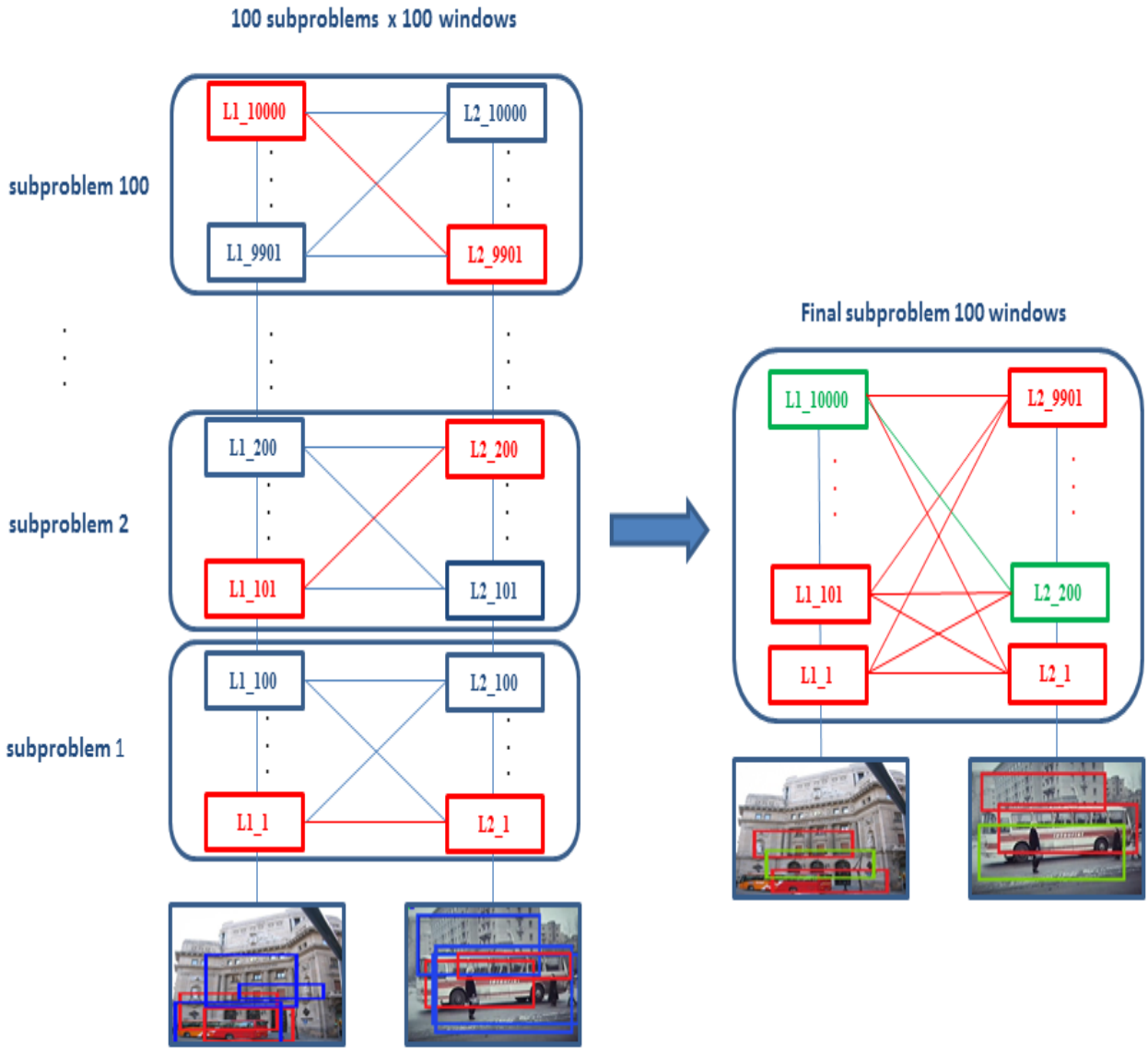


Figure 3.1: **Divide and conquer LocLearn.** *The figure illustrates a fully CRF with 2 images and 10000 candidate windows per image. The windows are split into 100 intermediate subproblems and LocLearn is run for each subproblem.* L_{n_i} *is the* i^{th} *window/label in image* n*. The solutions to the subproblems (highlighted in red) are combined in the last step to form the final LocLearn subproblem, which gives the final solution (highlighted in green).*

10000 candidate windows per image is 0.95 and SNRCLS 0.22. We present the hitrate and SNRLCS for the 12 classes in Table 3.1. We notice that there are classes for which the hitrate is 1. We also observe that the lowest hitrate obtained is 0.88 (for the class boat-left), proving the existence of very difficult images in the Pascal07 training data.

class	viewpoint	hitrate	SNRCLS
bus	left	1.00	0.19
	right	0.95	0.07
boat	left	0.88	0.28
	right	0.93	0.15
aeroplane	left	0.97	0.28
	right	1.00	0.29
horse	left	0.91	0.25
	right	0.92	0.09
bicycle	left	0.97	0.28
	right	0.97	0.17
motorbike	left	0.97	0.32
	right	0.97	0.32

Table 3.1: **Hitrate and SNRCLS for the classes in Pascal07 6x2 with 10000 windows per image sampled with the objectness function [2]**

In our setup of D&C LocLearn, following the notations from algotihm 1, we have $W = 10000$ windows in total and $W_s = 100$ windows in a subset, and thus a total of $S = 100$ subproblems. The D&C LocLearn is run with identical settings for each LocLearn subproblem, settings which correspond to versions (d) for localization and (f) for localization and learning from Table 2.1. The 100 intermediate subproblems are run in parallel and their solutions are aggregated in a final subproblem. Each subproblem (intermediate or final) can be run only until the localization stage, or localization and learning can be performed (which correspond to settings (d), and respectively (f) from Table 2.1). This results in four possible setups which will be denoted (g)-(j) D&C LocLearn.

Divide and Conquer LocLearn	
Intermediate Subproblems	Final Subproblem
localization	(g) localization
	(h) localization+learning
localization+learning	(i) localization
	(j) localization+learning

Table 3.2: Four setups of Divide and Conquer LocLearn

We review here the general setup of a LocLearn problem with setup (d) and (f) from Table 2.1. For meta-training data images from 24 classes from Pascal07 are used and for training data images from other 12 classes from Pascal07 6x2 (aeroplane, bus, boat, horse, motorbike, bicycle each with two views) are used.

In setup (d) 100 candidate windows per image are sampled using the objectness measure [2]. The energy function 2.1 incorporates unary potential ω for objectness, pairwise potentials λ for shape similarity (aspect-ratio) and pairwise potentials Γ_f for appearance similarity according to four cues (GIST, CHIST,

HOG, SURF). The weights α and the parameters of the pairwise potentials $\theta_{\Gamma_f}, \theta_\lambda$ are learned from meta-training data. We compare the performance (in terms of corLoc) of D&C LocLearn (g) which uses only the localization step in all subproblems with the performance of LocLearn setup (d). All the subproblems (intermediate or final) from D&C LocLearn have the same setup as LocLearn (g), namely each subproblem uses 100 candidate windows per image and the paramaters and weights of the potential terms are the same. We discuss the properties of the intermediate subproblems of D&C LocLearn only for setup (g).

In LocLearn (f) from Table 2.1, the setup from (d) is run with a maximum of 10 iterations in which the localization and learning are alternating. The class specific unary pairwise potentials Υ_f corresponding to each cue f and the image responsabilities ρ are adapted. Our D&C LocLearn which uses adaptation to a specific class will be compared to the results of LocLearn setup (f).

3.2.1 Properties of the selected windows of intermediate subproblems

We first look at the properties of the windows selected by the intermediate LocLearn subproblems from setup (g), when only localization is done. Fig. 3.2 shows a visualization of the 100 candidate windows/image that enter the final LocLearn subproblem in setup (g) and 100 candidate windows/image for LocLearn (d) for 6 images taken from the classes bus-left and aeroplane-left.

It can be observed from Fig. 3.2a that the solutions of the subproblems give most of the time windows covering in the Pascal sense a target object. This leads to a set of windows which enters the final subproblem, where for one image the majority of windows localize the object (colored green), or the majority of windows are not covering an object (colored red).

A selected window in an image is considered to be more accurate, the more it overlaps with the ground-truth bounding box.

Statistical properties of the windows. If we look at the statistical properties of the set of candidate windows for the final subproblem, we notice that they have greater SNRCLS and smaller hitrate than a set of windows that are simply sampled using the objectness measure.

Fig. 3.3 shows a histogram of the distribution of the hitrate for all 100 intermediate subproblems of D&C LocLearn (g). The green line represents the hitrate of the final LocLearn subproblem in (g) and the red line represents the hitrate of the windows used in LocLearn (d). The mean hitrate over all 100 intermediate subproblems and over all classes is 0.83, while the mean hitrate over all classes of the images in the final subproblem of D&C LocLearn (g) is only 0.69. The blue line represents the hitrate in a sample LocLearn problem setup (d), and its mean hitrate over all classes is 0.85. This value is close to the mean of the 100 subproblems which used in a similar way windows sampled from the ojectness measure.

It is surprising at first that the hitrate of the windows in the last subproblem, highlighted in green in Fig. 3.3, are somehow smaller than the ones used in LocLearn (d). This can be explained by looking again at Fig. 3.2. In a normal LocLearn problem, it happens relatively often that there are some images which have very few windows (1-4) hitting an object, which keeps the hitrate high. In the case of the final set of windows for the D&C LocLearn problem, most of the windows are biased to cover a certain area in the image, and sometimes this is not the one which covers an object in a Pascal sense.

It can be viewed from Fig. 3.2b and Fig. 3.2e, that there exists also images for which almost all candidate windows are hitting a target object. This makes the value of SNRCLS for the set of windows of the last subproblem to be larger. Fig. 3.4 shows a histogram of the distribution of SNRCLS over 100 intermediate subproblems in D&C LocLearn (g). The green line represents the SNRCLS of the final LocLearn subproblem in (g) and the red line represents the SNRCLS of the windows used in LocLearn (d). The mean SNRCLS over all classes of the images in the final subproblem of D&C LocLearn (g) is 0.37, which is better

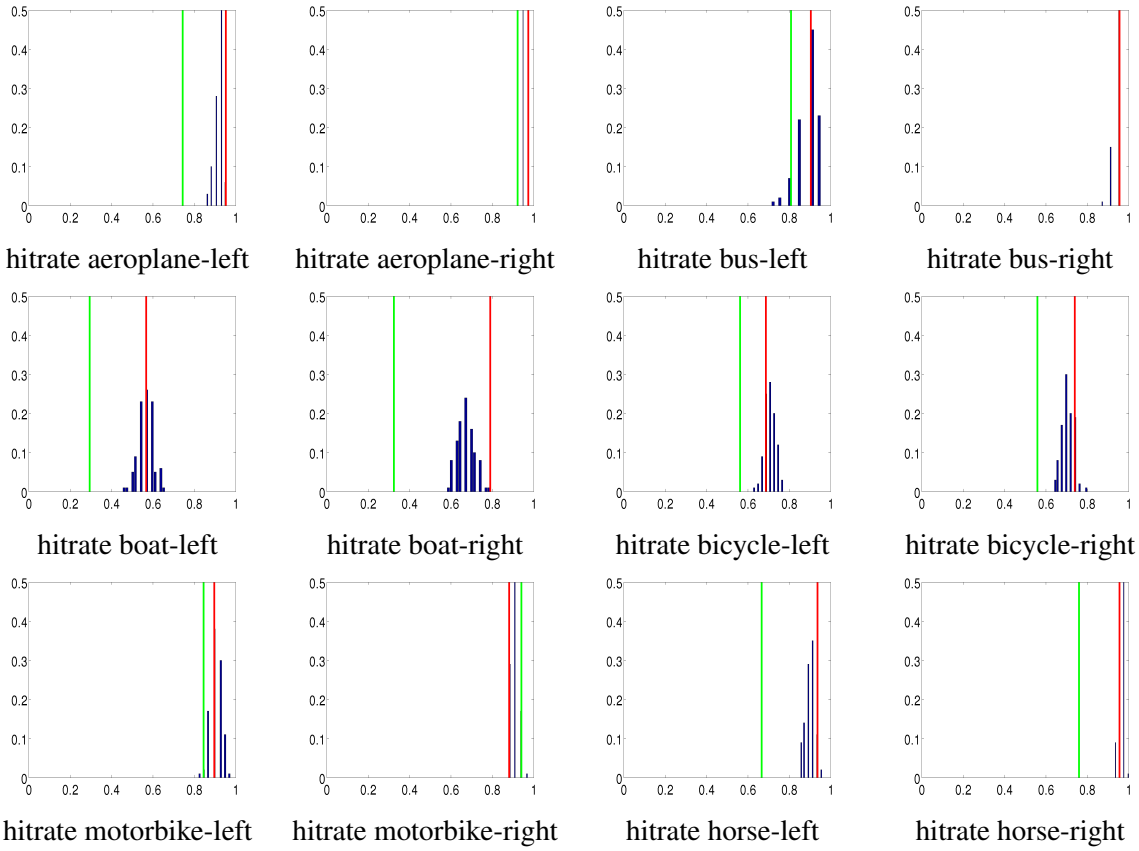


Table 3.3: **Distribution of hitrate for 100 LocLearn subproblems.** *For each class-viewpoint, a histogram (shown in blue) depicts the hitrate distribution over 100 LocLearn subproblems. With a red vertical line is highlighted the hitrate of the windows used in setup (d) from Table 2.1 and with a green vertical line is highlighted the hitrate of the windows that enter in the final subproblem of Divide and conquer LocLearn (g). The mean hitrate for the windows in setup (g) over all classes is 0.69 which is smaller than the average hitrate in LocLearn setup (d) of 0.85.*

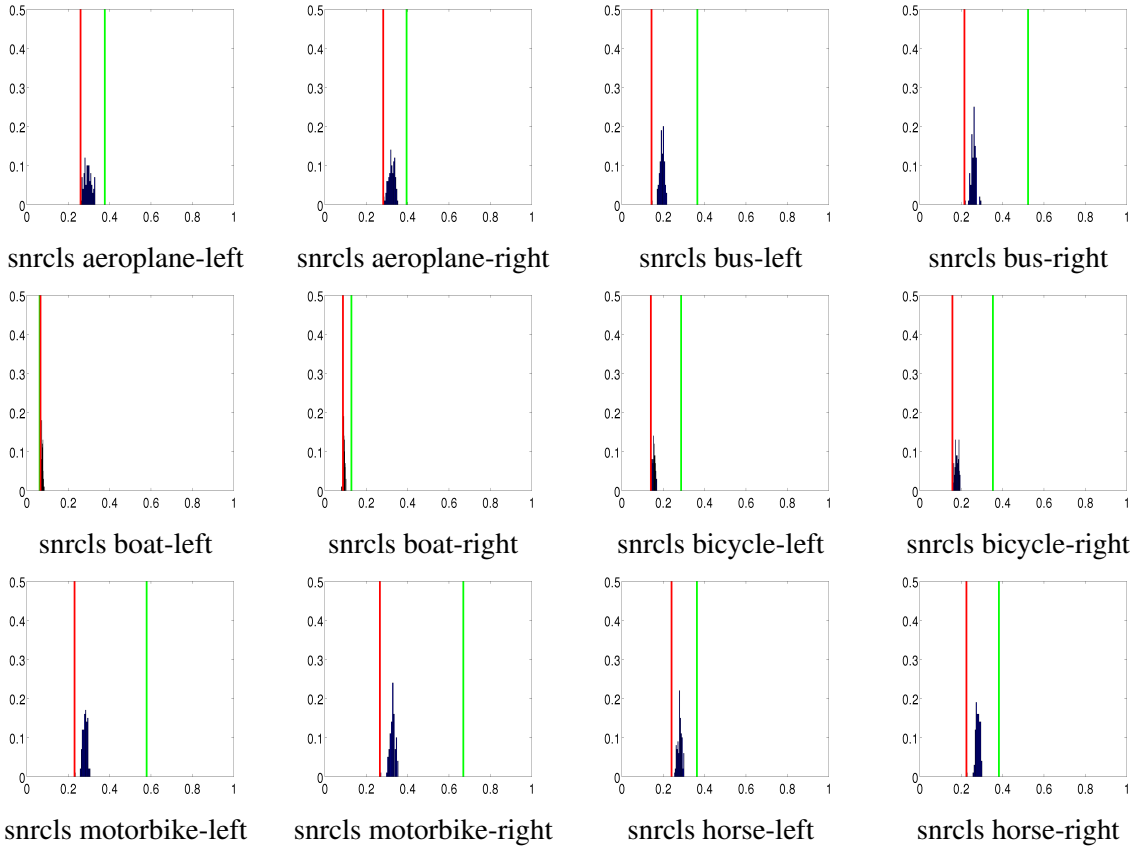


Table 3.4: **Distribution of SNR per class for 100 LocLearn subproblems.** *For each class-viewpoint, a histogram depicts the SNRCLS distribution over 100 LocLearn subproblems. With a red vertical line is highlighted the SNRCLS of the windows used in setup (d) from Table 2.1 and with a green vertical line is highlighted the SNRCLS of the windows that enter in the final subproblem of Divide and conquer LocLearn (g). The SNRCLS for the windows in setup (g) is usually bigger than the one in LocLearn setup (d).*

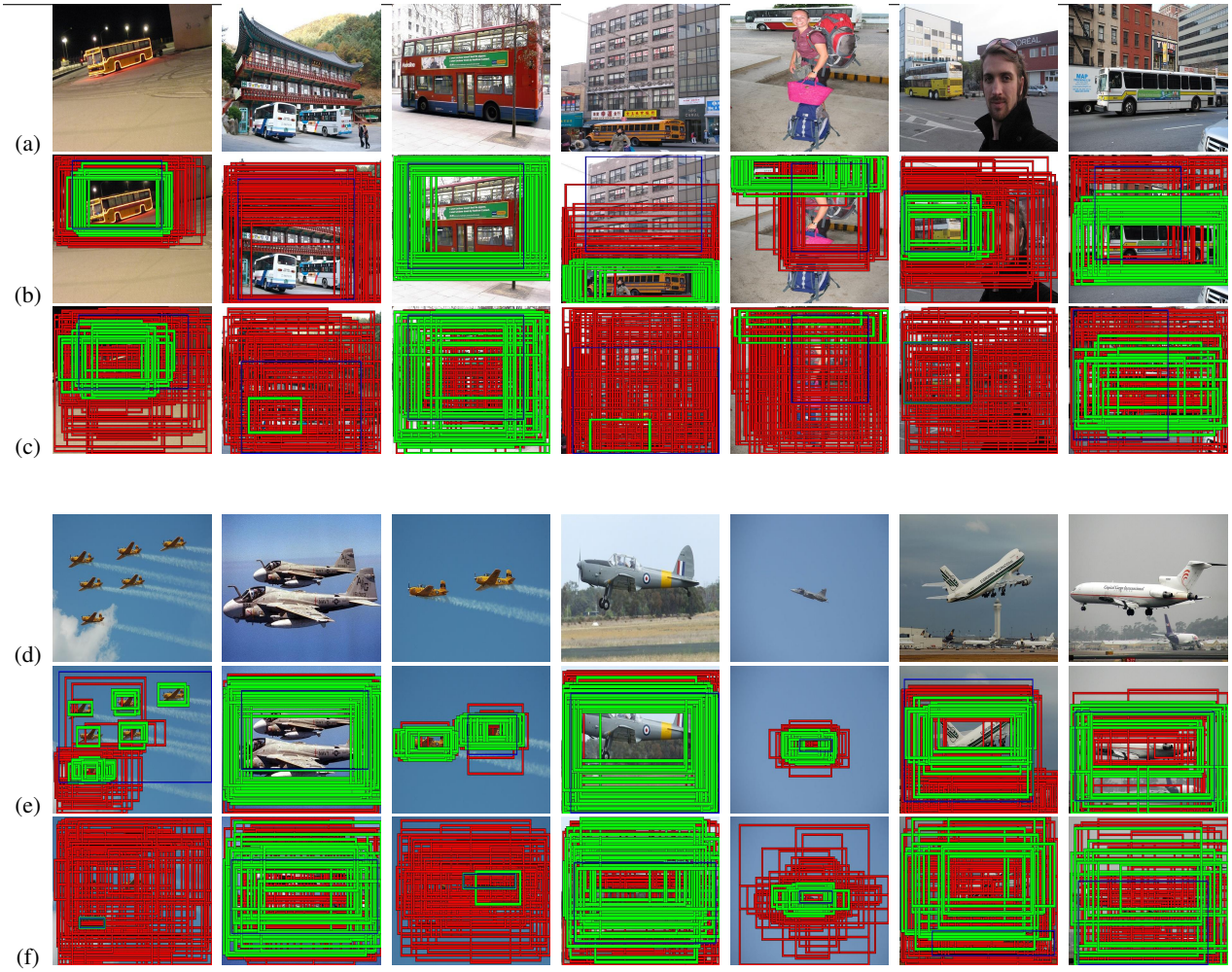


Figure 3.2: **Visualization of 100 windows per image for two classes.** *On the first row are 6 images taken from the class bus-left Pascal07 6x2. On the second row are plotted 100 windows/image. These are the selected windows from each of the 100 problems in setup (g) of D&C LocLearn, and also the ones which will be input as candidate windows to the final LocLearn subproblem. On row (c) are 100 windows/image obtained by sampling from the objectness measure [2]. Green windows cover the object according to Pascal criterion, red windows don't. The windows in c) seem more random than those in b), where most of them are concentrated around the same object or area. The last rows (d)-(f) contain similar plots but for the class aeroplane left.*

than that the mean SNRCLS over all 100 intermediate subproblems and over all classes which is 0.22. The windows that are used in LocLearn (d) have a SNRCLS of 0.19, which is close to the mean SNRCLS of 0.22 of the intermediate subproblems. This is consistent with our expectations, since LocLearn (d) and the 100 LocLearn problems use candidate windows with similar properties, which are sampled with the objectness function [2].

To summarize, the windows that enter the final subproblem have better SNRCLS but worse hitrate compared to a normal set of windows sampled with the objectness measure. The distributions of the hitrate and SNRCLS over 100 LocLearn problems show that localizing objects is more difficult for some classes

than for others. For the classes boat and bicycle, the hitrate and SNRCLS are on average smaller than for the rest of the classes. Moreover, the variance in hitrate and SNRCLS for a particular class shows that by repeating the sampling, the candidate windows can be better or worse, in which case tighter theoretical limits are imposed on the performance of corLoc.

3.2.2 corLoc results for Divide and Conquer LocLearn

In Table 3.5 are presented the results of four versions of the D&C LocLearn algorithm. Each subproblem (intermediate or final) can be run only until the localization stage (corresponding to setup (d) from Table 2.1), or localization and learning can be performed (setup (g) from Table 2.1). For D&C LocLearn, the corLoc of the final selected windows is reported in the last column.

Method		PASCAL07 6x2 corLoc
(d) LocLearn localization		37%
(f) LocLearn localization+learning		50%
Divide and conquer LocLearn		
Intermediate Subproblems	Final Subproblem	
localization	(g) localization	41%
	(h) localization+learning	45%
localization+learning	(i) localization	49%
	(j) localization+learning	51%

Table 3.5: corLoc results of Divide and Conquer LocLearn on Pascal07 6x2

We discuss the results below, depending on whether in the subproblems is performed only localization or localization and learning.

- *Localization in the intermediate subproblems*. We can see that if we run LocLearn up to the localization step in all subproblems, the corLoc obtained by D&C LocLearn in the final subproblem is of 41% (g). This outperforms the results obtained in setup (d) for which only one LocLearn problem is run up to localization (with corLoc of 37%). However, if we add the learning stage only in the last subproblem (corresponding to setup (h)), the final corLoc 45% does not outperform the best referenced LocLearn with adaptation, which is 50% (f). This is primarly due to the fact that the windows used in the last problem have different characteristics from the normal subproblem, as it was discussed in detail in the previous section. To give better performance results, the settings (e.g. the weights α of each potential term in eq. 2.1) of LocLearn in the final subproblem should be learnt again from the meta-training data.

- *Localization and Learning in the intermediate subproblems*. When learning is performed in both the intermediate subproblems and in the final subproblem, we observe a slight increase in corLoc returned by D&C LocLearn after the learning stage from 49% (i) to 51% (j). However, the final result of 51% is better than the best referenced corLoc of 50% obtained for one LocLearn problem with learning (setup (f)). The subproblems in (j) had similar setup as in (f). Intuitively, an even better improvement can be obtained by learning the parameters for the final subproblem, as we can observe that in the normal settings of a LocLearn problem, corLoc increases from 37% to 50% after the learning stage, while in the final subproblem of D&C LocLearn the performance increases less, from 41% to 45% or from 49% to 51%.

A visualization of the windows selected by the setups (d) and (g) from Table 3.5 is depicted in Fig. 3.3. Six images are selected for each class/viewpoint combination and in the images one can also see the ground-truth bounding boxes.

Figure 3.3: **Localization results for Divide and conquer LocLearn.** *Blue windows represent ground-truth, pink windows are the ones selected by LocLearn setup (d) in Table 2.1 and with green are colored the final windows selected in the localization stage by Divide and Conquer LocLearn setup (g). There are images for which the 2 algorithms return the same labels, but on average the green windows are better as the results in Table 3.5 show.*

The *accuracy* of the selected windows is computed as the *average overlap* of the ground-truth bounding boxes with the selected windows which cover an object in the Pascal sense. For a window w which localizes an object, the overlap with the ground-truth window gt is computed as $\frac{|w \cap gt|}{|w \cup gt|}$. In Table 3.6 are presented the accuracy of the selected windows for four LocLearn setups.

Methods		Accuracy	corLoc
localization	(d) LocLearn	0.67	37%
	(g) D&C LocLearn	0.69	41%
localization+learning	(f) LocLearn	0.66	50%
	(j) D&C LocLearn	0.66	51%

Table 3.6: Accuracy of selected windows on Pascal07 6x2

Although corLoc is bigger in setups (f) and (j), accuracy is a little smaller than in setups (d) and (g). This is explained by the fact that the model becomes less restrictive in the choice of selected windows, which increases the number of selected windows that localize an object, but in the same time the selected windows do not overlap so tightly with the ground-truth.

The *autoCorLoc* for two LocLearn setups is computed as the percentage of images for which the selected windows from the two setups overlap in a Pascal sense. The autoCorLoc for D&C LocLearn (g) and LocLearn (d) is 0.83. The same autoCorLoc is obtained for the setups D&C LocLearn (j) and LocLearn (f). This shows that the windows selected by D&C LocLearn are similar with the ones given by one LocLearn problem.

corLoc of intermediate subproblems The distribution of the values of corLoc obtained for 100 LocLearn subproblems after the localization stage, corresponding to setup D&C LocLearn (g) are shown in Fig. 3.2.2 for each class-viewpoint combination. With a red vertical line is highlighted the corLoc from results (d) in Table 2.1 and with a green vertical line is highlited the corLoc obtained in the final subproblem of D&C LocLearn (g).

There is a large variance of corLoc within a class for different LocLearn subproblems. If the same LocLearn setup is run several times but with different candidate windows, corLoc can vary even up to $\pm 20\%$ for a class. However, we can say that some classes have better localization results than others. Boat proves to be a difficult class, giving relatively lower corLoc values in all subproblems, while motorbike proves to be one of the easiest. The largest corLoc obtained for the classes boat-left and boat-right over 100 problems is $\sim 20\%$, which is smaller than the smallest corLoc obtained for the classes motorbike-left and motorbike-right of $\sim 50\%$.

Furthermore, it can be noticed that the selected windows in the final subproblem of D&C LocLearn are better in almost all the classes (except aeroplane left, motorbike right and horse right). This supports the idea that corLoc varies among different LocLearn problems which have same setup but different sampled windows. Moreover, using more windows leads to better results, as the algortihm has a greater set of labels to choose from.

Memory savings D&C LocLearn has different memory requirements for storing the appearance pairwise potentials than a normal LocLearn problem. Its memory complexity is given by $O(N^2 W_s(W + W_s))$, where N is the number of images, W is the number of sampled windows and W_s is the number of windows we use in a subproblem. When $W_s < \frac{\sqrt{5}-1}{2} W \sim 0.61W$, the memory complexity of D&C LocLearn is smaller than that of LocLearn of $O(N^2 W^2)$. This is the case also in our experiments, in which $W_s = 100$ and $W = 10000$. This results in saving storage memory by a factor of 99. We note that in practice D&C LocLearn always uses less memory for storing the potentials, as we normally break down our inital problem in at least two subproblems, which corresponds to $W_s \leq 0.5W$.

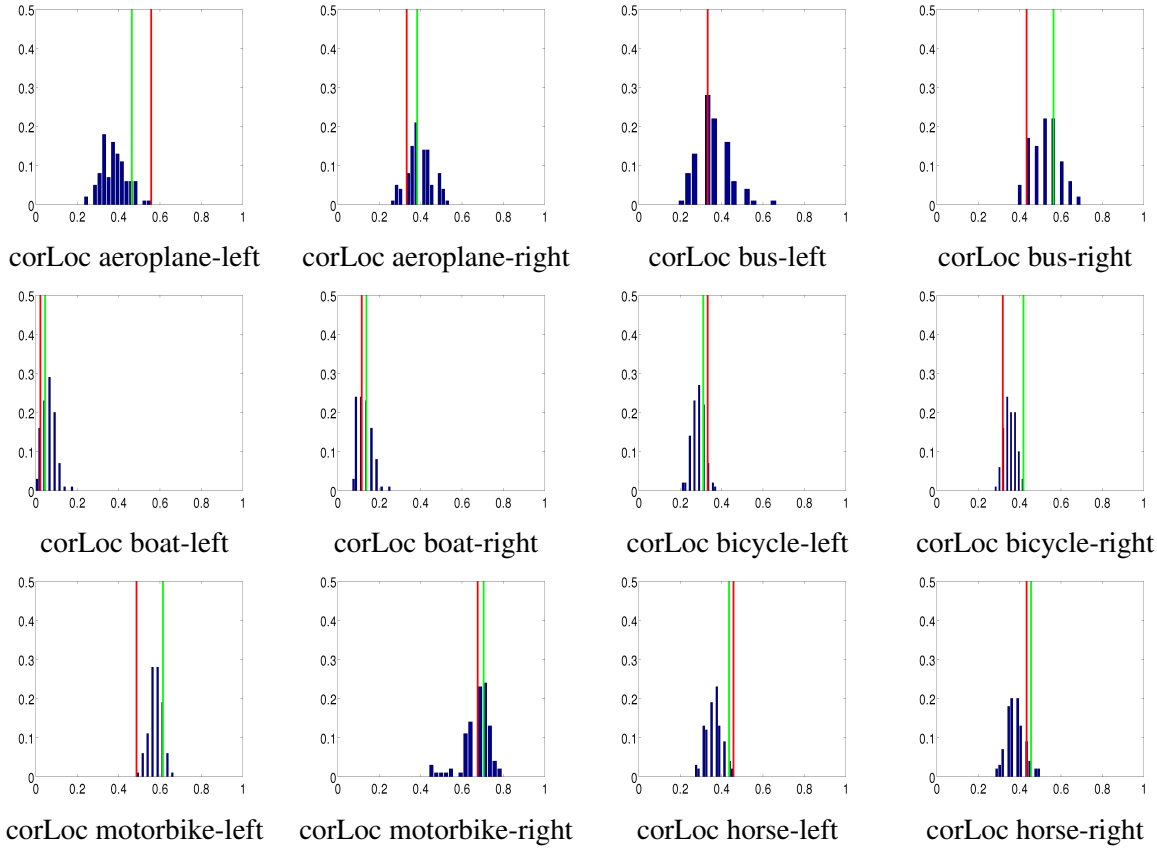


Figure 3.4: **Distribution of corLoc for 100 LocLearn problems with different candidate windows.** *With a red vertical line is highlighted the corLoc from results (d) in Table 2.1 and with a green vertical line is highlighted the corLoc obtained in the final subproblem of Divide and Conquer LocLearn. The histogram (blue) represents the distribution of the corLoc obtained in 100 intermediate subproblems of D&C LocLearn (g).*

3.3 Discussion

The performance of LocLearn in localizing objects of a new class is theoretically bounded by the hitrate of our sampled windows. The more candidate windows we have per image, the more likely it is to have one or more windows that cover an object in the Pascal sense for every image.

We propose a Divide and Conquer version of LocLearn which can use a number of candidate windows per image in the order of thousands. Using so many candidate windows per image is not feasable in the normal settings of LocLearn due to its computational complexity of $O(N^2W^2)$, where N is the number of images and W is the number of sampled windows per image. Divide and Conquer LocLearn uses the independence property of its intermediate subproblems to reduce the running time from $O(N^2W^2)$ to $O(N^2W_s^2)$, where W_s is the number of windows per subproblem. This is a big improvement since in general W_s is much smaller than W. This approach does not only save computational time, but also memory for storing the pairwise potentials. The memory complexity of D&C LocLearn is $O(N^2W_s(W + W_s))$ which in practice is always smaller than the memory requirements of a normal LocLearn problem of $O(N^2W^2)$.

The newly proposed method improves corLoc from 37% (LocLearn (c)) to 41% if only localization

is done, and from 50% (LocLearn (f)) to 51% if both localization and learning stages are performed. This shows that using more candidate windows per image helps in obtaining better localization results. Moreover, the performance can be improved even more when localization and learning are run iteratively if different parameters are used in the final problem of D&C LocLearn, in which the candidate windows have different statistical properties than a set of windows sampled with the objectness measure [7]. Namely, in many cases the majority of candidate windows are either covering an object or are biased towards another area in the image, leading to higher SNRCLS and smaller hitrate than a set of candidate windows sampled with the objectness function.

Chapter 4

LocLearn with Binary encoding of GIST descriptor

As mentioned in the introduction, the storage requirements of the feature vectors poses also constraints on the number of candidate windows per image that LocLearn can use. Because we extract different descriptors for each candidate window of every image in the input set, the memory complexity for storing the descriptors is $O(NWD)$, where N is the number of images, W is the number of candidate windows and D is the sum of the dimensionalities of the descriptors used.

We see from Table 2.1 that the GIST feature alone gives good initial candidate windows in the localization step and can be considered a crucial descriptor, since if we use all four cues or GIST alone, corLoc is the same, the other descriptors having more impact in the learning stage. Because it is a powerful descriptor, we devote our attention in the following sections to LocLearn setups which employ only GIST features. In this section, we propose a binary encoding of GIST features in order to reduce the storage requirements of the descriptors. Although we experiment only with binary encoded GIST features, same approach can be applied to arbitrary feature vectors. Our approach aims to keep the localization performance of LocLearn close to the ones reported in Table 2.1.

4.1 GIST

In [1] is proposed a feature descriptor for capturing the overall structure of a scene. It is known in the literature as GIST, which literally means the essence or the substance of a concept, action. GIST provides a representation of an image which characterizes the scene globally avoiding segmentation or edge detection schemes. The authors of [1] made a survey in which people were asked to group images that resemble in a global aspect or structure. They defined the *spatial envelope* of a scene as having at least five important properties which are reviewed here shortly:

- naturalness: man-made or natural environment
- openness: open or closed horizon
- roughness: the sizes of the elements in the scene
- expansion: the viewpoint from which the scene is observed
- ruggedness: rough or even surface

GIST encodes an image in a low dimensional space, where each dimension represents an envelope property. There exists a correlation between the envelopes and spectral information such as phase, amplitude and energy spectrum of the Discrete Fourier Transform of an image. The energy spectrum encodes general dominant patterns, the amplitude gives information about orientation and smoothness and the phase function about the position of elements in the image. Thus, the feature provides a way of modeling a complex image in a lower dimensional space which can be used as the first stage in an object detection algorithm.

Note that in our task, a GIST feature is extracted for each window, not for the whole image as in [1]. Omitting the mathematical background of extracting a GIST feature, we show only what determines the dimensionality of GIST in our implementation. Every sampled window is rescaled to a square image of size 128 x 128. Before filtering, the image is normalized by dividing by the local luminance variance. The window is further split into 4 x 4 blocks and Gabor filters are applied to each block and for each color channel. We compute oriented edge responses at three scales with 8, 8 and 4 orientations per scale giving a total of 20 Gabor filters. In our experiments, the dimensionality of the GIST feature extracted from a window is determined as $nrColorChannels \times nrBlocks \times nrFilters = 3 \times 16 \times 20 = 960$.

4.2 LSBC encoding of GIST

The storage and processing of high-dimensional data imposes large restrictions on the aplicability of many computer vision algorithms. Often the data is represented by high-dimensional feature descriptors coming from a large set of images. In order to reduce the dimensionality of the data, different methods have been proposed for *binary encoding* the high-dimensional descriptors such that the descriptors that are similar in the Euclidean space are mapped to similar binary codes in the Hamming space. The encoding algorithms aim to preserve the nearest neighbours of the high-dimensional data points in the Hamming space.

Restricted Boltzmann machines [14], locality sensitive hashing [8] and spectral hashing [15] are some of the state-of-the-art methods for computing binary codes from high dimensional data which learn the code parameters in a data dependent fashion. Unlike these methods, in [13] is proposed a Locality Sensitive Binary Codes (LSBC) Hamming embedder which is not dependent on the distribution of the input data, and thus the method can be easily applied to arbitrary feature descriptors. Furthermore, the embedding consists in a simple and fast to compute random projection. The high-dimensional descriptors are mapped to binary codes such that the value of a shift-invariant kernel is preserved.

A real valued feature vector $x \in R^D$ can be encoded as a binary vector y of length B using a LSBC mapping $\phi : R^D \rightarrow \{0,1\}^B$. The i^{th} bit of y is computed as

$$y_i = \phi_i(x) = \text{sgn}[\cos(x \cdot r_i + b_i) + t_i] \tag{4.1}$$

where $r_i \sim \text{Normal}(0, \kappa I)$, $b_i \sim \text{Unif}[0, 2\pi]$ and $t_i \sim \text{Unif}[-1, 1]$.

The normalised Hamming distance between two binary vectors y_1 and y_2 approximates a shift-invariant kernel function

$$\frac{d_H(y_1, y_2)}{B} \approx \frac{1 - K(y_1, y_2)}{2}, \tag{4.2}$$

where $K(x, y) = e^{-\kappa||x-y||^2}$. The parameters r, b and t are randomly drawn from the respective distributions, and only the parameter κ can be adapted to a specific task.

The authors of [13] give theoretical proofs that a set of N points in a R^D space can be encoded in a $\{0,1\}^{O(log(N))}$ space preserving the similarity of the distances in Euclidean and Hamming spaces with large probability. They also give a thorough proof that although the encoding is done by randomly projecting in a lower dimensional binary cube, the normalized Hamming distance between two binary strings converges

as a function of the Euclidean distance of the original vectors. Due to its convergence properties, LSBC becomes more powerful when the size of the binary codes is increased.

When more bits are used, it is more likely that we deal with bigger Hamming distances and we can increase the kernel bandwith κ to keep the approximation in eq. 4.2 more accurate. The variation of κ also influences the radius in which the search of nearest neighbors is being done.

4.3 LocLearn with binary encoding of GIST feature

In this section, we aim to reduce the memory complexity of LocLearn for storing the GIST descriptors of the candidate windows which is in $O(NWD_{GIST})$, where N is the number of images, W is the number of sampled windows per image and $D_{GIST} = 960$ is the dimensionality of the descriptor. To this end, we binary encode our GIST descriptors using the LSBC [13] Hamming embedder. This is a general approach which can be applied also to the other descriptors. Our focus is on this particular descriptor since it proved to have an important role in the localization stage of LocLearn.

We propose a modified LocLearn algorithm that uses only one cue which corresponds to a binary encoded GIST feature using LSBC [13]. This hybrid version of the LocLearn framework will be referred to in the following sections as *Binary GIST LocLearn*. As we are interested only in the localization performance of GIST, Binary GIST LocLearn will be compared to setup (c) in Table 2.1 which also uses one cue (GIST) up to the localization stage.

As mentioned in sec. 2 eq. 2.4, the sum of squared differences is used as the pairwise energy contribution of two windows with respect to a cue f:

$$\Gamma_f(l_n, l_m|I_n, I_m) = ||l_n^f(I_n) - l_m^f(I_m)||_2^2, \tag{4.3}$$

where $l_n^f(I_n)$ and $l_m^f(I_m)$ are the descriptors according to cue f.

In order to use the same weights α_{Γ_f} for the appearance pairwise potential terms in eq. 2.1 as the ones used in LocLearn (c), we need to approximate the squared Euclidean distance (eq. 4.3) as a function of the Hamming distance of the encoded features.

We want to keep the same weights as this allows for a better comparison of the algorithms Binary GIST LocLearn and LocLearn (c), which have similar setups. The only difference is the fact that the first algorithm uses binary GIST features while the second one uses normal GIST features. Moreover, using weights adapted to binary features requires more effort as they have to be learned again from the meta-training data.

We now derive an approximation of the squared Euclidean distance 4.3 by using the properties of the LSBC encoding from eq. 4.2:

$$\begin{aligned} \tfrac{2}{B} d_H(y_1, y_2) &= 1 - e^{-\kappa \frac{||x_1 - x_2||^2}{2}} \\ 1 - \tfrac{2}{B} d_H(y_1, y_2) &= e^{-\kappa \frac{||x_1 - x_2||^2}{2}} \\ -\kappa \tfrac{||x_1 - x_2||^2}{2} &= \ln(1 - \tfrac{2}{B} d_H(y_1, y_2)) \end{aligned}$$

where x_1, x_2 are GIST descriptors and y_1, y_2 their corresponding binary encoded vectors.

This simplifies to

$$||x_1 - x_2||^2 = \frac{-2}{\kappa} \ln(1 - \frac{2}{B} d_H(y_1, y_2)). \tag{4.4}$$

The similarity of two windows according to binary encoded features is now given by

$$\Gamma_f(l_n, l_m|I_n, I_m) = \frac{-2}{\kappa}\ln(1 - \frac{2}{B}d_H(l_m^f(I_m), l_m^f(I_m))), \tag{4.5}$$

where $l_n^f(I_n)$ and $l_m^f(I_m)$ are binary descriptors according to cue f. In our case the cue f corresponds to binary encoded GIST descriptors using LSBC.

The performance of Binary GIST LocLearn is now tied by the properties of the LSBC encoding, since the values of the appearance pairwise potentials are now approximated by a function which depends on the Hamming distance of the encoded features.

Note that if GIST is encoded using LSBC in B bits, the memory complexity for storing the descriptors for N images and W sampled windows per image is reduced from $O(NWD_{GIST})$ doubles (each with 64 bits) to $O(NWB)$ bits. This can mean a large saving in storage memory since usually the number of bits used for encoding a descriptor using LSBC is relatively small, up to 2048 bits [13].

4.4 Experiments

Binary GIST LocLearn setup In Binary GIST LocLearn, we keep the settings the same as in Table 2.1 (d), but instead of GIST features we use binary encoded GIST features. As meta-training and training data the 24 classes and respectively 12 classes from Pascal07 6x2 are used. Each image in the training data comes with the same set of 100 candidate windows for both LocLearn (c) and Binary GIST LocLearn.

In setup (c) 100 candidate windows per image are sampled using the objectness measure [2]. The energy function 2.1 consists of unary potential ω for objectness, pairwise potentials λ for shape similarity (aspect-ratio) and pairwise potentials Γ_f for appearance similarity according to one cue (GIST). Image responsibilities ρ and unary appearance potentials Υ_f are set to uniform and the other parameters were learned from generic classes from the meta-training data. The algorithm is run up to the localization step which uses information only from generic classes.

Our results for Binary GIST LocLearn (g) will be compared to the ones obtained in LocLearn setup (c), the only difference between the two algorithms being the fact that now the appearance pairwise potential term Γ_f uses LSBC encoded GIST features and it is approximated using 4.3.

Implementation details The parameter κ in eq. 4.1 is trained on the generic classes from the meta-training data and it was observed that $\kappa = 1$ gives the best performance.

The term inside the logarithmic function in eq. 4.3 can be negative if the normalized Hamming distance $\frac{1}{B}d_H(l_m^f(I_m), l_m^f(I_m))$ is greater than 0.5, which means that more than half of the bits of the binary descriptors differ. In this case, we can replace $1 - \frac{2}{B}d_H(l_m^f(I_m), l_m^f(I_m))$ with eps $= 2.2204e^{-16}$ value, which gives a large approximative value for the sum of squared differences. This is a reasonable approximation, since if more than half of the bits of the binary descriptors differ, it means that also the corresponding original descriptors differ a lot and the Euclidean distance between them should be large. The windows are not similar, and thus a large cost is assigned to their appearance similarity.

We try various code sizes from 2^6 to 2^{11} (powers of 2 only), and take the one that gives the best performance on the meta-training data. When code sizes of 16 or 32 bits are used, most of the values ($\sim$90%) of the pairwise distances are approximated with -2ln(eps), since very often more than half of the bits of two descriptors differ. The results of this setups are not significant, since if most of the pairwise distances

have the same values, the similarity appearance looses its importance in selecting a good configuration of windows.

Table 4.1 and Fig. 4.1 present the localization results of Binary GIST LocLearn on the meta-training classes.

nr bits	128	256	512	1024	2048
corLoc	29.62%	30.02%	28.39%	28.07%	27.08%

Table 4.1: coLoc of Binary GIST LocLearn on meta-training dataset Pascal07 6x2

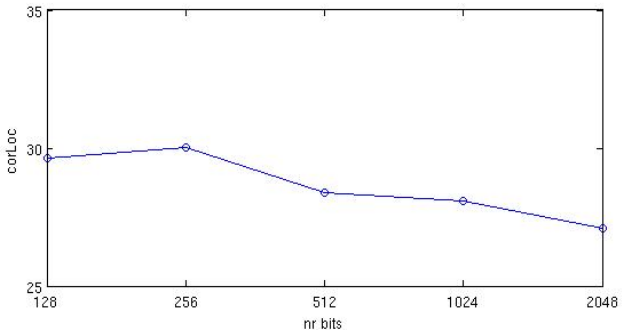


Figure 4.1: **coLoc of Binary GIST LocLearn on meta-training dataset Pascal07 6x2.** *The plot shows corLoc results for Binary GIST LocLearn with the GIST feature binary encoded using LSBC [13] on the 24 meta-training classes in Pascal07 6x2. On the horizontal axis are the number of bits used for encoding a GIST descriptor. The best results are obtained for code sizes of 256 bits and if more bits are used corLoc starts to decrease.*

If we increase the number of bits up to 256 bits, corLoc increases, afterwards it starts to decrease. A visualization of the windows selected by Binary Loclearn when more than 256 bits are used shows that the algorithm has preference for big windows. A decrease in performance when the number of bits is increased is mentioned also in the case of spectral hashing encoding but used and evaluated in a different context [13]. One of the reasons why we obtain worse performance for larger code sizes can be the fact that the kernel bandwidth κ should normally be increased when more bits are used, but in our experiments we only use $\kappa = 1$. Increasing the value of κ for large code sizes ensures that the normalised Hamming distance between two binary vectors approximates better the shift-invariant kernel function from eq. 4.2.

corLoc results of Binary GIST LocLearn using 256-bits encoding of GIST The results on the meta-training data suggest that $B = 256$ bits is the desired encoding which gives best localization results. We run Binary GIST LocLearn on the training data with the same set of candidate windows and same setup as in LocLearn (c) but with GIST encoded into a 256-bit vector using LSBC. We obtain corLoc of 36% which is very close to 37%, the corLoc obtained when LocLearn uses normal GIST features. A visualization of the windows selected by 256-bits Binary GIST LocLearn and LocLearn (c) is depicted in Fig. 4.2.

We verify the choice of B by repeating the experiments from Table 4.2 for the training data as well and observe that corLoc decreases when more than 512 bits are used. Using $B = 512$ we obtain corLoc of 36.7% on the training data, which actually suggests that a 512-bits encoding can in practice give better results.

Figure 4.2: **Localization results of Binary GIST LocLearn with 256-bit encoded GIST**. *Blue windows represent groundtruth, red windows are the ones selected by LocLearn setup (c) and with green are colored the windows selected by LocLearn with the GIST feature encoded in a 256 bit vector. Binary GIST LocLearn gives localization results comparable to LocLearn which uses GIST not encoded.*

Memory savings for storing GIST descriptors In our implementation, GIST is a 960-dimensional array of Matlab double numeric type, which uses 960×64 bits = 61440 bits. If we encode the feature in 256 bits, we use 15360/256 = 240 times less space for one descriptor. For a training dataset with 400 images and 100 sampled windows per image, this means reducing the storage requirements from 2.28Gb to only 9.76Mb.

4.5 Discussion

One of the limitations of LocLearn which does not allow the use of more candidate windows per image is the memory requirements for storing the descriptors, which has complexity $O(NWD)$, where N is the

number of images, W the number of sampled windows per image and D the sum of the dimensions of the descriptors used. We propose an approach for which the storage memory of the GIST descriptor can be reduced drastically using a binary encoding, which comes at a small cost of reducing the performance of the algorithm. This is a general method that can be employed for other descriptors as well, our focus being GIST due to its key-importance in the localization stage of LocLearn. The memory complexity for storing the descriptors is now reduced to $O(NWB)$, where B is the code size, which potentially allows the use of more candidate windows per image.

Chapter 5

LBP inference for sparse distance matrices

LocLearn is limited by the memory storage requirements for the potentials (unary and pairwise) $O(NW + \frac{N(N-1)}{2}W^2)$ terms and for the descriptors $O(NWD)$, where N is the number of images, W the number of sampled windows per image and D the sum of the dimensions of the descriptors. If we are able to save memory, we can allow the use of a larger W, which means LocLearn can select a configuration out of a greater set of windows, leading to better localization results.

The dominant term in the memory complexity comes from storing the pairwise potentials distances. As it was derived in sec. 2, there are $O(N^2W^2)$ pairwise potentials, one between each pair of windows. In this part, we propose a method for reducing the storage requirements by using sparse distance matrices that store the pairwise potentials such that we are able to run LocLearn for larger values of W. This approach does not only save memory, but also speeds up the computations because fewer elements are computed in the distance matrices. The central idea of computing less distances is to use the spatial overlap of two windows in an image and its corrrelation to their appearance distance.

5.1 Exploiting spatial overlap for computing sparse pairwise distance matrices

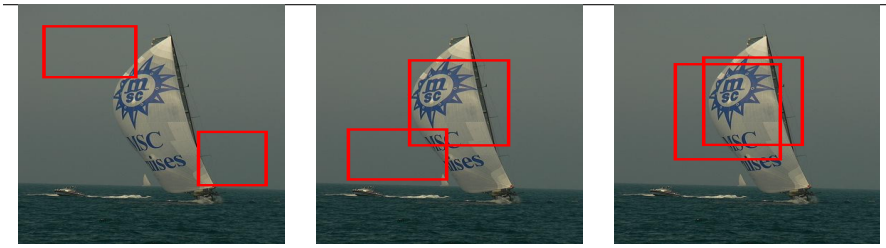
We define the *spatial overlap* of two windows w_a and w_b from an image as $o(w_a, w_b) = \frac{|w_a \cap w_a|}{|w_a^{\cup} w_b|}$, where $|w|$ is the area of a window. The overlap has values in the interval $[0, 1]$, namely it has value 0 if the windows don't have any area in common and it is 1 if the windows are identical.

The overlap o of two windows and their appearance distance d_{app} are correlated. If the overlap is very large, we expect that the window descriptors are similar and they have a small appearance distance. If the overlap is 0 or very small, we can actually not say with certainty anything about their appearance distance. In Fig. 5.1 are shown examples of images with windows having different spatial overlap.

Upper bound on appearance distance as a function of spatial overlap The spatial overlap o will be used for computing a smaller number of pairwise appearance distances between windows from *different* images. For this, we first derive an *upper bound* $\mathcal{B}$ on the appearance distance as a function of the spatial overlap

$$d_{app}(w_1^{'}, w_1^{''}) \leq \mathcal{B}(o(w_1^{'}, w_1^{''})), \tag{5.1}$$

for two windows $w_1^{'}$ and $w_1^{''}$ in an image.



a) b) c)

Figure 5.1: **Spatial overlap of two windows**
In a), the two windows have a 0 overlap, but they both contain mostly background information and their appearance distance will be very small. In b) the two windows have a small overlap, but their appearance distance will be large because their actual content is now very different. In c) the overlap is large (> 0.5) which makes more likely that their appearance distance will be small.

$\mathcal{B}$ can be either a *statistically* bound which is learned from meta-training data or an *exact* bound which is theoretically derived. While a theoretical bound is not trivial to compute for certain appearance descriptors, a statistical bound can always be estimated.

The bound $\mathcal{B}$ is used to compute all pairwise distances smaller than a threshold d_ϵ between two sets of window descriptors coming from images I_1 and I_2. Later we propose a similar task for LocLearn.

Assuming each set has cardinality W, for a window w_1 from I_1 we need to compute W distances, for each window in image I_2. Lets assume that we have computed $d_{app}(w_1, w_2^{'})$ and we want now to compute $d_{app}(w_1, w_2^{''})$, with windows $w_2^{'}, w_2^{''}$ from I_2. If $d_{app}(w_1, w_2^{'})$ is larger than the threshold and the overlap $o(w_2^{'}, w_2^{''})$ is large, using $\mathcal{B}$ we can bound $d_{app}(w_1, w_2^{''})$. If the lower bound of $d_{app}(w_1, w_2^{''})$ is greater than d_ϵ, we can discard its computation. The situation is depicted in Fig. 5.2.

Efficient computation of appearance distances less than a threshold We continue with a more theoretical formulation of the above concept which shows how the bound $\mathcal{B}$ can be used for efficiently computing appearance distances less than a threshold.

d_{app} is a metric, and we can derive a simple relation between $d_{app}(w_1, w_2^{'})$ and $d_{app}(w_1, w_2^{''})$ using the triangle inequality for bounding $d_{app}(w_1, w_2^{'})$:

$$|d_{app}(w_1, w_2^{''}) - d_{app}(w_2^{''}, w_2^{'})| \leq d_{app}(w_1, w_2^{'}) \leq |d_{app}(w_1, w_2^{''}) + d_{app}(w_2^{''}, w_2^{'})| \tag{5.2}$$

Using $d_{app}(w_2^{'}, w_2^{''}) \leq \mathcal{B}(o(w_2^{'}, w_2^{''}))$, we obtain:

$$\max(0, |d_{app}(w_1, w_2^{''}) - \mathcal{B}(o(w_2^{'}, w_2^{''}))|) \leq d_{app}(w_1, w_2^{'}) \leq \mathcal{B}(o(w_2^{'}, w_2^{''})) + d_{app}(w_1, w_2^{''}). \tag{5.3}$$

This implies that

$$d_{app}(w_1, w_2^{'}) - \mathcal{B}(o(w_2^{'}, w_2^{''})) \leq d_{app}(w_1, w_2^{''}). \tag{5.4}$$

If we know that the left hand side of the equation above is greater than d_ϵ, we can discard the computation of the distance. $\mathcal{B}$ is used to derive bounds for the appearance distance of two windows from *different* images. This is the key concept which stays at the base of the *Distance-Overlap algorithm* (DO algorithm) [3] which efficiently computes pairwise distances between two sets of descriptors lower than a threshold. Since this is not part of this work, here we have explained only the concepts on which the algorithm is based.

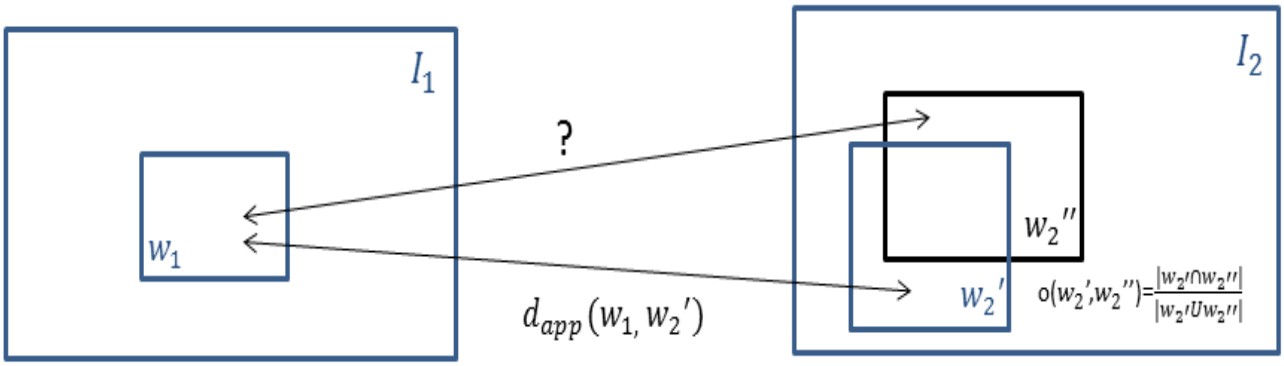


Figure 5.2: **Bound on appearance distance using spatial overlap**
If we know $d_{app}(w_1, w_2^{'})$ and $o(w_2^{'}, w_2^{''})$, we can find bounds for $d_{app}(w_1, w_2^{''})$. If we are interested in appearance distances lower than a threshold d_ϵ, we can avoid computing $d_{app}(w_1, w_2^{''})$ if its lower bound is greater than d_ϵ.

In LocLearn, appearance distances $d_{app}(w_1, w_2)$ appear as pairwise potentials $\Gamma_f(w_1, w_2|I_1, I_2)$ for a cue f. The pairwise potentials are terms of an energy function 2.1 that we want to minimize. We expect that only potentials with small values are of interest to us, and we can try to avoid the computation of those that are greater than a given threshold using DO algorithm. Note that calculating the overlap of two windows is an $O(1)$ operation and we can avoid recomputing it for the same pair of windows, by precomputing a spatial overlap table for the set of candidate windows of each image.

A theoretical bound $\mathcal{B}$ is difficult to compute for some type of descriptors, in which case it can be computed statistically from training data. A bound approximated statistically can in practice be tighter than a theoretical bound. For appearance descriptors based on histograms, an exact theoretical bound is easier to compute. In the next section, we derive a theoretical bound for the distance of CHIST descriptors of two windows as a function of their overlap. The reasoning in deriving $\mathcal{B}$ for CHIST can be used for other descriptors based on histograms.

5.1.1 Theoretical upper bound on appearance distance of Color Histogram descriptors

We derive an upper bound for the square Euclidean distance of two normalized CHIST descriptors as a function of their overlap and ratio $r = \frac{|w_1 \setminus w_2|}{|w_1|}$. The normalized CHIST of a window w quantized into C color bins is a C-dimensional vector

$$\text{CHIST}(w) = \frac{1}{|w|}(h_1^w, h_2^w, .., h_C^w), \tag{5.5}$$

where $|w|$ is the area of the window (the total number of pixels) and h_i^w is the number of pixels from w whose color falls into bin i. We assume that the colors of the windows are first quantized, such that only C colors are used and in the next sections we will use the term color to refer to the quantized color c of a pixel, $c \in 1, ..., C$.

To keep the notation simple, we denote the square Euclidean distance $d^2_{\text{CHIST}}(w_1, w_2)$ for two windows

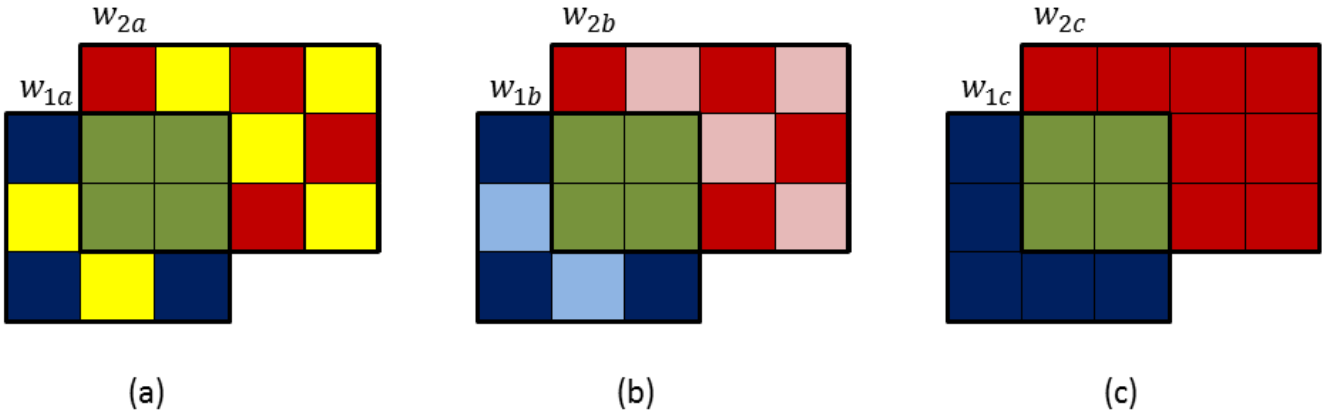


Figure 5.3: **Bound on appearance distance using spatial overlap**
We have the relation $d(w_{1a}, w_{2a}) < d(w_{1b}, w_{2b}) < d(w_{1c}, w_{2c})$. Case b) illustrates the limit case when the upper bound $\mathcal{B} = d(w_{1b}, w_{2b})$ for two windows w_1 and w_2 with the given size and common overlap. Case c) illustrates a case when $\mathcal{B}$ has the maximum possible value.

w_1, w_2 in an image with $d^2(w_1, w_2)$. The distance is defined as

$$d^2(w_1, w_2) = \sum_{c=1}^{C} \left(\frac{h_c^{w_2}}{|w_2|} - \frac{h_c^{w_1}}{|w_1|} \right)^2 \tag{5.6}$$

Properties of bound $\mathcal{B}$ for CHIST We first make important observations about the properties of $\mathcal{B}$ for normalized CHIST, which are explained through three example.

Fig. 5.3 depicts three different color configurations of two windows w_1 and w_2 for a given $o(w_1, w_2)$. Each square represents a pixel, thus the sizes of the windows w_1 and w_2 are 3x3 and respectively 3x4. In the example we consider 6 quantized colors (dark-blue, light-blue, dark-red, light-red, yellow, green). Using this representation, in the three cases from Fig. 5.3 we have the following appearance distances:

- CHIST$(w_{1a}) = (\frac{3}{9}, 0, 0, 0, \frac{2}{9}, \frac{4}{9})$
  CHIST$(w_{2a}) = (0, 0, \frac{4}{12}, 0, \frac{4}{12}, \frac{4}{12})$
  $d^2(w_{1a}, w_{2a}) = (\frac{3}{9})^2 + 0 + (\frac{4}{12})^2 + 0 + (\frac{2}{9} - \frac{4}{12})^2 + (\frac{4}{9} - \frac{4}{12})^2$

- CHIST$(w_{1b}) = (\frac{3}{9}, \frac{2}{9}, 0, 0, 0, \frac{4}{9})$
  CHIST$(w_{2b}) = (0, 0, \frac{4}{12}, \frac{4}{12}, 0, \frac{4}{12})$
  $d^2(w_{1b}, w_{2b}) = (\frac{3}{9})^2 + (\frac{2}{9})^2 + (\frac{4}{12})^2 + (\frac{4}{12})^2 + 0 + (\frac{4}{9} - \frac{4}{12})^2$
  Since $(\frac{2}{9} - \frac{4}{12})^2 \leq (\frac{2}{9})^2 + (\frac{4}{12})^2$ we have that $d(w_{1a}, w_{2a}) \leq d(w_{1b}, w_{2b})$. This suggests that the bound increases when the windows don't have colors in common outside their intersetion area.

- CHIST$(w_{1c}) = (\frac{5}{9}, 0, 0, 0, 0, \frac{4}{9})$
  CHIST$(w_{2c}) = (0, 0, \frac{8}{12}, 0, 0, \frac{4}{12})$
  $d^2(w_{1c}, w_{2c}) = (\frac{5}{9})^2 + 0 + (\frac{8}{12})^2 + 0 + 0 + (\frac{4}{9} - \frac{4}{12})^2$
  Since $(\frac{3}{9})^2 + (\frac{2}{9})^2 \leq (\frac{5}{9})^2$ and $(\frac{4}{12})^2 + (\frac{4}{12})^2 \leq (\frac{8}{12})^2$, we have that $d(w_{1b}, w_{2b}) \leq d(w_{1c}, w_{2c})$. The bound increases when the colors in $w_{1c} \setminus w_{2c}$, $w_{2c} \setminus w_{1c}$ are spread over fewer bins. In this case $\mathcal{B}$

reaches its maximum value, which corresponds to a configuration when every window has only two colors, with one color in common in their intersection area.

To summarize, given $o(w_1, w_2)$, the bound $\mathcal{B}(o(w_1, w_2))$ is represented by the case when the pixels from $w_1 \setminus w_2$ and $w_2 \setminus w_1$ do not have any quantized color in common. The bound increases when the colors of the pixels from $w_1 \setminus w_2$ fall in a fewer number of bins (fewer colors) and similarly for $w_2 \setminus w_1$. The bound $B(o(w_1, w_2)) \in [0, \mathcal{B}_{max}]$ reaches its maximum value $\mathcal{B}_{max}$ when the histogram h^{w_1} contains only two colors $c_{w_1 \setminus w_2}, c_{w_1 \cap w_2}$ and histogram h^{w_2} contains only two colors $c_{w_2/w_1}, c_{w_1 \cap w_2}$. The color $c_{w_1 \cap w_2}$ is the color of the pixels from $w_1 \cap w_2$. Fig. 5.3c represents the case when the maximum bound $\mathcal{B}_{max}$ is attained, which corresponds to a value of $\mathcal{B}_{max} = (\frac{|w_1 \setminus w_2|}{|w_1}|)^2 + (\frac{|w_2 \setminus w_1|}{|w_2|})^2 + (\frac{|w_1| \cap |w_2|}{|w_1|} - \frac{|w_1 \cap w_2|}{|w_2|})^2$. The maximum numerical value of $\mathcal{B}_{max}$ is 2, which is obtained if the two windows are single colored and they don t intersect.

We continue with the derivation of the upper bound for the square Euclidean distance $d^2(w_1, w_2)$ as a function of the overlap o and of the ratio r, where

$$o = \frac{|w_1 \cap w_2|}{|w_1 \cap w_2| + |w_1 \setminus w_2| + |w_2 \setminus w_1|}, 0 \leq o \leq 1, \tag{5.7}$$

and

$$r = \frac{|w_1 \cap w_2|}{|w_1|}, 0 \leq r \leq 1. \tag{5.8}$$

In our derivation we make use of the observations above and also note that the elements of a normalized CHIST of a window w have the following properties $\sum_{c=1}^{C} h_c^w = |w|$ and $0 \leq h_c^w, \forall c \in 1, ..., C$.

Using $h_c^{w_1} = h_c^{w_1 \cap w_2} + h_c^{w_1 \setminus w_2}$ and $h_c^{w_2} = h_c^{w_1 \cap w_2} + h_c^{w_2 \setminus w_1}$ in definition 5.6 and expanding the terms we obtain

$$\begin{aligned}
d^2(w_1, w_2) &= \sum_{c=1}^{C} \left(\frac{h_b^{w_2 \setminus w_1} + h_b^{w_2 \cap w_1}}{|w_2|} - \frac{h_b^{w_1 \setminus w_2} + h_b^{w_2 \cap w_1}}{|w_1|} \right)^2 \\
&= \sum_{c=1}^{C} \left(\frac{h_b^{w_2 \setminus w_1}}{|w_2|} - \frac{h_b^{w_1 \setminus w_2}}{|w_1|} + h_b^{w_2 \cap w_1} \left(\frac{1}{|w_2|} - \frac{1}{|w_1|} \right) \right)^2 \\
&= \sum_{c=1}^{C} \left[\left(\frac{h_b^{w_2 \setminus w_1}}{|w_2|} \right)^2 + \left(\frac{h_b^{w_1 \setminus w_2}}{|w_1|} \right)^2 + \left(h_b^{w_2 \cap w_1} \right)^2 \left(\frac{1}{|w_2|} - \frac{1}{|w_1|} \right)^2 \right] \\
&- 2 \sum_{c=1}^{C} h_c^{w_2 \setminus w_1} h_c^{w_1 \setminus w_2} \frac{1}{|w_1||w_2|} + 2 \sum_{c=1}^{C} h_c^{w_2 \setminus w_1} h_c^{w_1 \cap w_2} \frac{1}{|w_2|} \left(\frac{1}{|w_2|} - \frac{1}{|w_1|} \right) \\
&+ 2 \sum_{c=1}^{C} h_c^{w_1 \setminus w_2} h_c^{w_1 \cap w_2} \frac{1}{|w_1|} \left(\frac{1}{|w_1|} - \frac{1}{|w_2|} \right).
\end{aligned}$$

We derive $\sum_{c=1}^{C} \left(h_c^{w_2 \setminus w_1} \right)^2 \leq \left(\sum_{c=1}^{C} h_c^{w_2 \setminus w_1} \right)^2 = |w_2 \setminus w_1|^2$ and similar relations for $\sum_{c=1}^{C} \left(h_c^{w_1 \setminus w_2} \right)^2$ and $\sum_{c=1}^{C} \left(h_c^{w_2 \cap w_1} \right)^2$ using the Cauchy[1] inequality. Equality holds when all the colors are concentrated in one bin, namely when $w_1/w_2, w_2/w_1$ and $w_1 \cap w_2$ are single colored (only one element in their corresponding histogram is non zero). If we also renounce at the term $\sum_{c=1}^{C} -h_c^{w_2 \setminus w_1} h_c^{w_2 \setminus w_1} \frac{1}{|w_1||w_2|}$ since it is always negative, we can derive an initial upper bound as

$$d^2(w_1, w_2) \leq \left(\frac{|w_2 \setminus w_1|}{|w_2|} \right)^2 + \left(\frac{|w_1 \setminus w_2|}{|w_1|} \right)^2 + |w_1 \cap w_2|^2 \left(\frac{1}{|w_2|} - \frac{1}{w_1} \right)^2 +$$
$$+ 2 \sum_{c=1}^{C} \left[h_c^{w_2 \setminus w_1} h_c^{w_1 \cap w_2} \frac{1}{|w_2|} \left(\frac{1}{|w_2|} - \frac{1}{|w_1|} \right) + h_c^{w_1 \setminus w_2} h_c^{w_1 \cap w_2} \frac{1}{|w_1|} \left(\frac{1}{|w_1|} - \frac{1}{|w_2|} \right) \right].$$

By denoting with A the sum of the first three terms on the right hand side, we can rewrite it as:

[1] $\sum_{i=1}^{N} x_i y_i \leq \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i$, for $0 \leq x_i, 0 \leq y_i$

$$d^2(w_1,w_2) \leq A + 2\sum_{c=1}^{C}\left[h_c^{w_2\setminus w_1}h_c^{w_1\cap w_2}\frac{1}{|w_2|}\left(\frac{1}{|w_2|}-\frac{1}{|w_1|}\right)+h_c^{w_1\setminus w_2}h_c^{w_1\cap w_2}\frac{1}{|w_1|}\left(\frac{1}{|w_1|}-\frac{1}{|w_2|}\right)\right]. \tag{5.9}$$

We can use the following relations in order to express A only in terms of o and r:

- $|w_1 \setminus w_2| = |w_1| - |w_1 \cap w_2| = |w_1|(1-r)$
- $|w_2| = |w_1|\frac{r-o(1-r)}{o}$, derived from $o = \frac{r|w_1|}{|w_2|+|w_1\setminus w_2|} = \frac{r|w_1|}{|w_2|+|w_1|(1-r)}$
- $|w_2 \setminus w_1| = |w_2| - |w_1 \cap w_2| = |w_1|\left(\frac{r-o(1-r)}{o} - r\right) = |w_1|\frac{r-o}{o}$.

Thus, we obtain: $A = \left(\frac{|w_2\setminus w_1|}{|w_2|}\right)^2 + \left(\frac{|w_1\setminus w_2|}{|w_1|}\right)^2 + |w_1\cap w_2|^2\left(\frac{1}{|w_2|}-\frac{1}{|w_1|}\right)^2 = |w_1|^2\left(\frac{r-o}{o}\frac{o}{r-o(1-r)}\right)^2\frac{1}{|w_1|^2}$ $+\frac{|w_1|^2(1-r)^2}{|w_1|^2} + r^2|w_1|^2\frac{1}{|w_1|^2}\left(\frac{2o-r-or}{r-o(1-r)}\right)^2 = \left(\frac{r-o}{r-o(1-r)}\right)^2 + (1-r)^2 + r^2\left(\frac{2o-r-or}{r-o(1-r)}\right)^2$.

Using again Cauchy inequality we derive

$$\sum_{c=1}^{C} h_c^{w_2\setminus w_1}h_c^{w_1\cap w_2} \leq \left(\sum_{c=1}^{C}h_c^{w_2\setminus w_1}\right)\left(\sum_{c=1}^{C}h_c^{w_1\cap w_2}\right) = |w_2\setminus w_1||w_2\cap w_1|$$

and in a similar way we obtain $\sum_{c=1}^{C} h_c^{w_1\setminus w_2}h_c^{w_1\cap w_2} \leq |w_1\setminus w_2||w_2\cap w_1|$. In the first relation, equality holds when w_1/w_2 and $w_1\cap w_2$ are single colored and their colors are different. A similar observation holds for w_2/w_1 and $w_1\cap w_2$. This setting corresponds to the maximum value of the bound $\mathcal{B}_{max}$ as it was illustrated in Fig. 5.3c.

Now we consider two cases:

- $|w_1| \leq |w_2| => \frac{1}{|w_2|} - \frac{1}{|w_1|} \leq 0$
- $|w_1| > |w_2| => \frac{1}{|w_2|} - \frac{1}{|w_1|} > 0$

In each of the cases, we renounce to the corresponding negative term from inequality 5.9 and we obtain:

- $|w_1| \leq |w_2|$

  $d^2(w_1,w_2) \leq A + 2\sum_{c=1}^{C} h_c^{w_1\setminus w_2}h_c^{w_1\cap w_2}\frac{1}{|w_1|}\left(\frac{1}{|w_1|}-\frac{1}{|w_2|}\right)$

  $\leq A + 2\frac{|w_1\setminus w_2||w_1\cap w_2|}{|w_1|}\left(\frac{1}{|w_1|}-\frac{1}{|w_2|}\right)$

  $= A + 2\frac{|w_1|r|w_1|(1-r)}{|w_1|}\frac{1}{|w_1|}\left(1-\frac{o}{r-o(1-r)}\right) = A + 2r(1-r)\frac{r-2o+or}{r-o(1-r)}$

  $= (1-r)^2 + \frac{1}{(r-o(1-r))^2}\left[(r-o)^2 + r^2(2o-r-ro)^2 + 2r(1-r)(r-o(1-r))(r-2o+or)\right]$

  The final bound is:

  $d^2(w_1,w_2) \leq (1-r)^2 + \frac{1}{(r-o(1-r))^2}\left[(r-o)^2 + r(r-2o+or)\left(r^2(-o-1)+r(2o+2)-2o\right)\right]$.

- $|w_1| > |w_2|$

  $d^2(w_1, w_2) \leq A + 2\sum_{c=1}^{C} h_c^{w_2 \setminus w_1} h_c^{w_1 \cap w_2} \frac{1}{|w_2|} \left(\frac{1}{|w_2|} - \frac{1}{|w_1|} \right)$

  $\leq A + 2\frac{|w_2 \setminus w_1||w_1 \cap w_2|}{|w_2|} \left(\frac{1}{|w_2|} - \frac{1}{|w_1|} \right) = A + 2\frac{|w_1|(r-o)}{o} \frac{r|w_1|}{|w_1|} \frac{o}{r-o(1-r)} \left(\frac{o}{r-o(1-r)} - 1 \right) \frac{1}{|w_1|} =$

  $= A + 2\frac{r(r-o)(2o-r-or)}{(r-o(1-r))^2}$

  $= (1-r)^2 + \frac{1}{(r-o(1-r))^2} \left[(r-o)^2 + r(2o-r-or)(r(2o-r-or) + 2r - 2o) \right]$

  The final bound is:

  $d^2(w_1, w_2) \leq (1-r)^2 + \frac{1}{(r-o(1-r))^2} \left[(r-o)^2 - r(r-2o+or)(r^2(-o-1) + r(2o+2) - 2o) \right]$.

To summarize, we have derived a theoretical upper bound $\mathcal{B}(o, r)$ for the square Euclidean distance $d^2(w_1, w_2)$ of two normalized CHIST descriptors , which is equal to

- $(1-r)^2 + \frac{1}{(r-o(1-r))^2} \left[(r-o)^2 + r(r-2o+or) \left[r^2(-o-1) + r(2o+2) - 2o \right] \right]$, if $|w_1| \leq |w_2|$

- $(1-r)^2 + \frac{1}{(r-o(1-r))^2} \left[(r-o)^2 - r(r-2o+or)(r^2(-o-1) + r(2o+2) - 2o) \right]$, if $|w_1| > |w_2|$,

with $r = \frac{|w_1 \cap w_2|}{|w_1|}$ and o the spatial overlap of the windows. In the following section we evaluate the theoretical bound $\mathcal{B}(o, r)$ experimentally.

5.1.2 Color Histogram bound experimental evaluation

In order to experimentally verify our theoretical upper bound, we compute $\mathcal{B}(o, r)$ for pairs of windows taken from real images. We use images taken from the classes bus, bicycle and horse from Pascal07 6x2. From each image, we sample pairs of windows and compute a set $\mathcal{S}$ of $(o, r, \mathcal{B}(o, r))$ triplets. The set $\mathcal{S}$ contains approximatively 8 million pairs of windows. We quantize the values of o and r into 100 values $o_1, o_2, ..., o_{100}$ and $r_1, r_2, ..., r_{100}$, and since $0 \leq o, r \leq 1$ this corresponds to a step size of 0.01.

In Fig. 5.4 we plot for each pair of quantized values (o_i, r_j) with $i, j \in 1...100$, the maximum theoretical bound found among our set $\mathcal{S}$ of triplets, namely $\mathcal{B}(o_i, r_j) = \max(\mathcal{B}(o, r))$, with $o \in o_i$, $r \in r_j$ and $(o, r, \mathcal{B}(o, r)) \in \mathcal{S}$. Since $o \leq r$, for the pairs (o_i, r_j) with $r_j < o_i$ nothing is plotted.

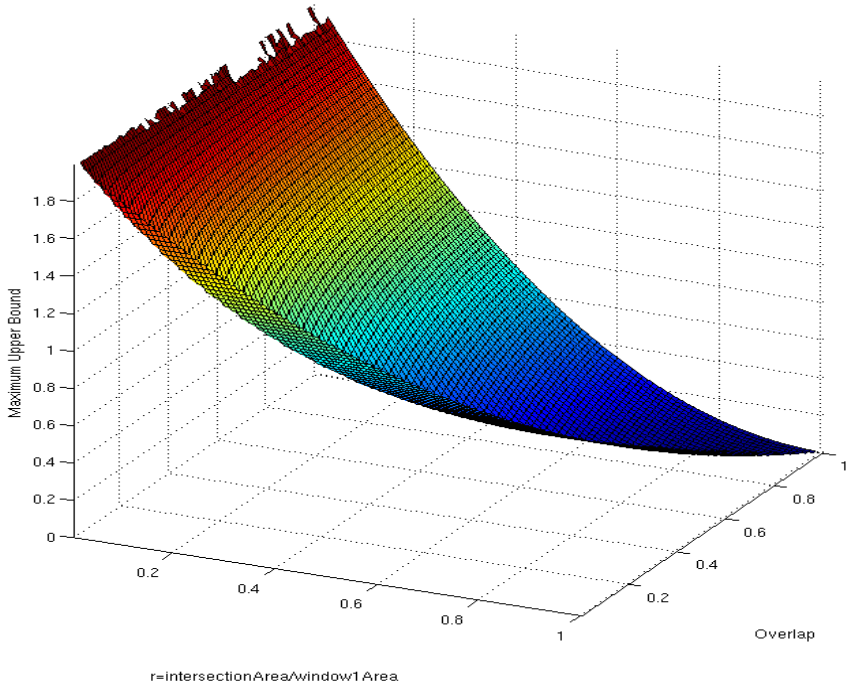


Figure 5.4: **Maximum Theoretical Upper Bound $\mathcal{B}(o, r)$ for CHIST.** *We obtain a set of $(o, r, \mathcal{B}(o, r))$ triplets from millions of pairs of windows. Both o and r are quantised into 100 values. For each quantised value (o_i, r_j), we plot the maximum bound $\mathcal{B}(o, r)$ from our set for which o is quantised into o_i and r into r_j.*

In Fig. 5.5 we renounce to the dependency of $\mathcal{B}$ on r and make a 2D plot relating only $\mathcal{B}$ and o. The maximum bound is computed now as $\mathcal{B}(o_i) = \max_r(\mathcal{B}(o_i, r))$, for $o_i \in \{0.01, 0.02, ..., 1\}$ and $(o_i, r_i, \mathcal{B}(o_i, r_i)) \in \mathcal{S}$. We note that the maximum upper bound is 2, which corresponds to the case when two windows don't overlap and are monocolored, each with a different color. The graph shows that the maximum bound decays rapidly with increasing overlap. This confirms our intuition presented before, that for a larger overlap, the appearance distance of two windows decreases. In this case, the bound becomes tighter.

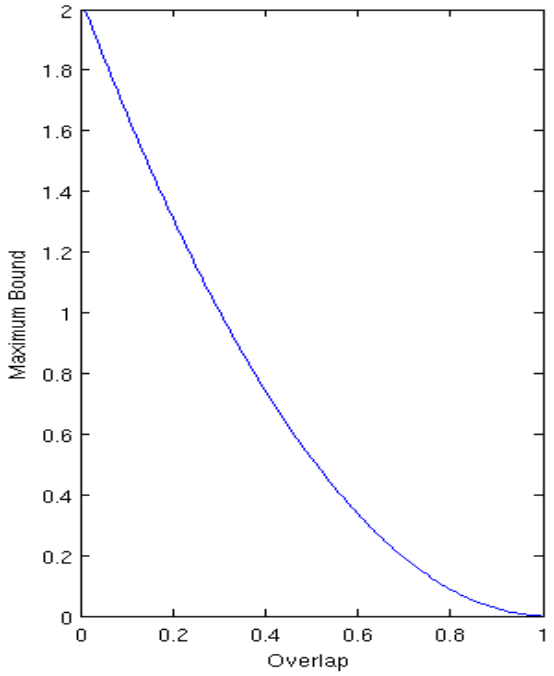


Figure 5.5: **Maximum Theoretical Upper Bound $\mathcal{B}(o)$ for CHIST.** *The spatial overlap o is quantized into 100 bin values. For each quantized value o_i, we plot the maximum bound $\mathcal{B}(o_i)$ from the set $\mathcal{S}$.*

5.2 Sparse LocLearn

Given a set of images from a new class, LocLearn selects a window per image likely to cover target objects. The selection of windows is done by minimizing the energy function defined in eq. 2.1. The energy consists of unary potentials and pairwise potentials. In particular, we expect that the appearance pairwise potentials for the selected configuration of windows are among the top $k\%$ smallest values of all pairwise potentials, for some value k. We therefore propose a speeded up version of LocLearn in which we compute only the smallest top $k\%$ appearance pairwise distances, leading to sparse distance matrices. The aim is to reduce both the time complexity for computation and the memory complexity of appearance pairwise potentials, which are in $O(W^2N^2)$, for N images and W sampled windows per image. This allows to run LocLearn with more candidate windows per image, but unlike Divide and Conquer LocLearn method from sec. 3, all the windows are used in the *same* LocLearn problem.

Sparse pairwise appearance distances Between every pair of images I_m, I_n from the training set, each with W sampled windows, we store the appearance pairwise potentials in a matrix $P^{mn;f}$ of size $W \times W$ such that the element $p_{ij}^{mn;f}$ represents the pairwise potential for appearance cue f between the i^{th} window in I_n and the j^{th} window in I_m. We remind that for cue f, $p_{ij}^{mn;f} = \Gamma_f(l_i^f(I_n), l_j^f(I_m)) = ||l_i^f(I_n) - l_j^f(I_m)||_2^2$ where $l_i^f(I_m)$ and $l_j^f(I_m)$ are the descriptors of window i in I_m and window j in I_m according to cue f. We denote with $l^{f,1:W}(I_n)$ and $l^{f,1:W}(I_m)$ the set of descriptors extracted from the sampled windows of the images I_n and I_m.

We now employ DO algorithm for computing all pairwise distances less than a threshold d_ϵ between the sets of descriptors $l^{f,1:W}(I_n)$ and $l^{f,1:W}(I_m)$. The DO algorithm returns a sparse representation of each $P^{mn;f}$ matrix by exploiting the correlation between the spatial overlap and the appearance distance of two

windows. For every two sets of descriptors we determine the threshold d_ϵ by an initial sampling of a few pairs. We compute only 1% randomly selected elements of $P^{mn;f}$ and set d_ϵ to be the 10%-quantile of the computed appearance distances.

Most of the pairwise distances in $P^{mn;f}$ returned by the DO algorithm are smaller than a threshold d_ϵ. The percentage of sparse elements in a distance matrix varies with the number of windows W used and also with the randomness of the sampling when computing the threshold. Large memory savings can result from using sparse matrices, because instead of storing the full $W \times W$ matrices they efficiently keep in memory only the elements of interest in the form of triplets (i, j, p_{ij}^{mn}).

This can be done for all matrices $P^{mn;f}$, namely for every pair of image and for each cue f, with $f \in$ GIST, CHIST, HOG, SURF which reduces the storage requirements of the pairwise appearance potentials. We experiment only with the more powerful GIST feature since if LocLearn is run up to the localization stage, using only GIST or all four cues, the performance is the same. Since we use only one cue, in the following sections we denote $P^{mn;f}$ with P^{mn}, the pairwise distance matrices of GIST descriptors. We compare the performance of our algorithm with the one of LocLearn (c) from Table 2.1, which also uses only GIST features.

Inference using sparse matrices The C++ implementation of the energy minimization algorithm TRW-S from [9] requires full pairwise distance matrices. In this setup, we use a Matlab implementation of the Loopy Belief Propagation[12] adapted for sparse matrices. LBP was implemented as a message-passing algorithm where the messages are represented by probabilities. The input to LBP is a sparse matrix but inside the message passing routine we set some values for the missing data. We therefore convert the unary and pairwise potentials into probabilities such that a potential term with a small cost corresponds to a large probability. We exploit eq. 2.1 which gives us an approximation of the posterior probability of a configuration of windows as a function of potentials.

The energy cost of an appearance pairwise potential according to cue f for any windows l_i from image I_n and l_j from I_m is $\Gamma_f(l_i^f(In), l_j^f(Im))$ and can be transformed into a probability similarly as in eq.2.1 using $p(l_i{}^f, l_j^f|In, (Im)) = e^{-\Gamma_f(l_i^f(I_n), l_j^f(I_m))}$. The probability $p(l_i|I_n, \omega)$ corresponding to the unary potential term for objectness $\omega(l_i|I_n, \omega)$ can be approximated in a similar way using $e^{-\omega(l_i|I_n,\omega)}$. However, the objectness score [2] of a window already gives us a value in the interval $[0, 1]$ which represents the probability that the window covers an object of any class. Thus we use this score to approximate $p(l_i|I_n, \omega)$.

The pairwise distances that are not computed by the sparse algorithm are bigger than our threshold, and thus in our approximation they should correspond to small probability values. We propose two algorithms, depending on how we approximate the pairwise distances that are not computed. We approximate all elements $p_{ij}^{mn} = \Gamma_f(l_n^f(In), l_m^f(Im)), \forall i, j \in 1..W$ from a matrix P^{mn} that are not computed, with the same distance $\mathcal{D}$. We look at two posibilities:

- $\mathcal{D} = \infty$, corresponding to a probability of $\mathcal{P} = e^{-\infty} = 0$ in LBP.
- $\mathcal{D} = \mathcal{M}$, where $\mathcal{M}$ is the mean of the elements in P^{mn} computed using the DO algorithm for a threshold d_ϵ. The motivation for using the mean is the fact that it minimizes the sum of squared errors between the actual value of the distances p_{ij}^{mn} that were not computed and their approximation. This corresponds to a probability of $\mathcal{P} = e^{-\mathcal{M}}$ in LBP.

For a given sparse P^{mn} matrix, even if we approximate in the LBP inference the unknown probability values with $\mathcal{P} = 0$, we don't have to store the 0 elements since they can be read on the fly. When we use

$\mathcal{P} = e^{-\mathcal{M}}$, we have to store for each pairwise matrix P^{mn}, the mean distance of its computed elements. This will add a $O(\frac{N(N-1)}{2})$ storage requirements, since there are $O(\frac{N(N-1)}{2})$ pairwise appearance matrices.

We propose therefore two different versions of LocLearn with sparse pairwise appearance distances for GIST which uses LBP inference. One version of the algorithm uses 0 values for approximating the probabilities of the unknown elements and will be referred to in the following sections as *Sparse LocLearn*. The other version of LocLearn uses the mean of the computed elements in a pairwise matrix to approximate the missing probabilities and will be referred to as *Sparse Mean LocLearn*.

Both algorithms are compared to LocLearn (c) from Table 2.1. The major differences between these methods consist in the fact that our proposed algorithms use sparse pairwise appearance distances for GIST features and the optimization of the energy function 2.1 uses LBP, while LocLearn (c) uses full pairwise distance matrices for GIST and the energy is optimized using TRW-S inference.

5.3 Experiments

We run Sparse LocLearn and Sparse Mean LocLearn on the training data from Pascal07 6x2 comprising of 6 classes, each with 2 viewpoints. The settings of the proposed algorithms that are not mentioned, are considered to be the same as in LocLearn (c). By employing the DO algorithm, we obtain sparse pairwise distances for GIST which reduce the memory storage requirements. This allows to sample more candidate windows per image, namely we experiment with $W \in \{100, 500, 1000\}$.

LocLearn which incorporates only unary potentials gives corLoc of 32% on Pascal07 6x2 training data. By adding to the model the pairwise potentials appearance corresponding to GIST, corLoc is increased to 37%. This proves the usefulness of using appearance similarity in our model.

Thresholding DO algorithm uses a threshold d_ϵ for computing only the pairwise distances in P^{mn} which are smaller than d_ϵ. In order to see if computing only the top $k\%$ of the distances is meaningful for our problem, we look to see if the pairwise distances corresponding to the selected windows by LocLearn are among the smallest distances in their corresponding pairwise matrix. For a pairwise matrix P^{mn} corresponding to images I_m and I_n, if the selected windows by LocLearn have indices i and respectively j, we look at the *rank* of the element p_{ij}^{mn} in P^{mn}. In other words, we make a sorted array of the elements in P^{mn} and look at the position of p_{ij}^{mn}. We compute the rank q as the ratio of the position of the element in the sorted array over the total number of elements in the array given by $W \times W$. This expresses the q-quantile of the distances in P^{mn}.

We run LocLearn (c) on the meta-training data classes of Pascal07 6x2 and we compute for each distance matrix P^{mn} the ranks of the p_{ij}^{mn}, where i, j are the selected windows by the algorithm for the images I_n and I_m respectively. In Fig.5.3 we plot the probability that the windows selected by the algorithm correspond to elements in pairwise matrices that are among the smallest $q\%$ elements.

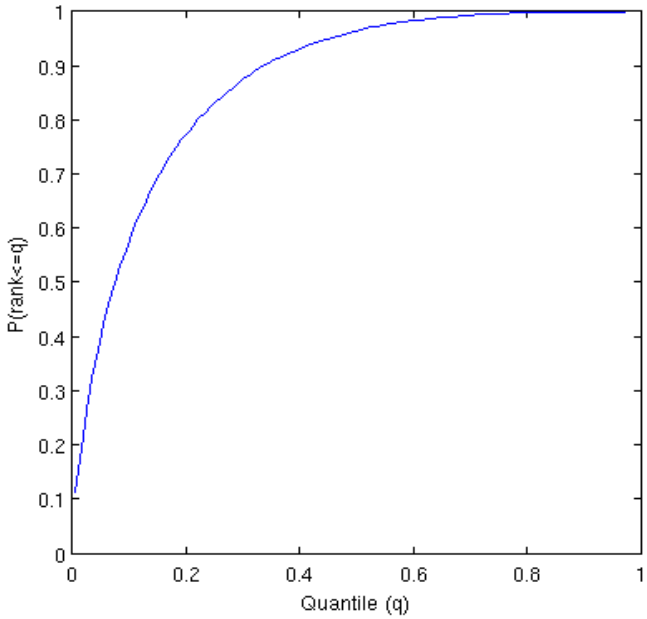


Figure 5.6: **Probability of selecting pairwise distances among the smallest q%.** *The plot uses information from the selected windows of LocLearn (c) on the meta-training classes.* $P(rank < q)$ *represents the probability that the pairwise potentials which enter in the minimum energy returned by LocLearn are among the smallest* $q\%$ *elements in their corresponding matrices.*

The plot shows that with probability of roughly 0.60, the pairwise potentials which enter in the minimum energy returned by LocLearn are in the top 10%, with probability of 0.75 are in the top 20%, with probability 0.85 are in the top 30% and with probability 0.95 are in the top 50%. This gives us an idea of what to expect when we use sparse appearance matrices with more than 50% of elements that are not computed. However, the plot uses only data obtained from LocLearn (c), which uses only 100 sample windows per image, while we use also setups with $W = 500$ and $W = 1000$.

Sparsity of pairwise distance matrices For all values of W, we compute using DO algorithm [3] sparse distance matrices P^{mn} of size $W \times W$. We compute the *sparsity* of a matrix as the percentage of elements from the matrix that are not computed. The sparsity of a matrix varies with the number W of windows used and with the threshold used (which we obtain statistically). The more windows we used, the sparser the matrix becomes.

Nr sampled windows W	100	500	1000
Sparsity of distance matrices	48.8%	71.8%	77.3%

Table 5.1: **Sparsity in pairwise distance matrices.**
Using sparse algorithm with statistically determined threshold for each matrix, we obtain sparser matrices as W *is increased. The percentages represent the average sparsity over all the appearance matrices for GIST computed with DO algorithm [3].*

The reduction in memory for a class which has N images and W windows per class is of almost 50% for $W = 100$, of 70% for $W = 500$ and of 77% for $W = 1000$. The memory complexity of the pairwise potentials is of $O(\frac{N(N-1)}{2}W^2)$. To give a rough idea what this actually means, we give some examples of the memory requirements for a class with N = 40 images and distance matrices stored in the Matlab double numerical type. For W = 100 this means a reduction from roughly 59.5 Mb to 30.46 Mb, for W = 500 from 1.45 Gb to 0.40 Gb and for W = 1000 from 5.81 Gb to 1.29 Gb. The training data of Pascal07 6x2 consists of 12 classes, each having between 21 and 50 images.

Nr sampled windows W		100	500	1000
corLoc	Sparse LocLearn (k)	31%	28%	17%
	Sparse Mean LocLearn (l)	32%	30%	29%*
	LocLearn (c)	37%	-	-

Table 5.2: **corLoc results for Sparse LocLearn and Sparse Mean LocLearn.**
The table presents the performance results of LocLearn with sparse pairwise distances obtained using DO algorithm. The newly proposed methods can be run with more candidate windows per image since the sparse matrices reduce memory requirements. The performance of Sparse LocLearn and Sparse Mean LocLearn is smaller than the referenced LocLearn (c) from Table 2.1 which has a similar setup.

corLoc results For all values of W, Sparse LocLearn (k) and Sparse Mean LocLearn (l) give worse localization results than LocLearn (c), which is our targeted reference algorithm when only GIST features are used. The corLoc values are in the range $28\% - 31\%$ even when W was increased to 500 or 1000. The performance of Sparse Mean LocLearn is slightly better than Sparse LocLearn, which can be explained by the fact that using the mean of the computed distances to approximate the sparse elements is a better approximation in the mean squared error sense, than simply using infinity.

If we rerun the experiment from LocLearn setup (k) or (l), we obtain different corLoc results, but still in the same range of $28\% - 31\%$. The variations appear because we use a different threshold d_ϵ for each pairwise matrix, which is computed by randomly sampling pairwise distances from the matrix.

Another parameter which influences our results is the maximum number of iterations for running LBP when it does not converge. This is set by default to $4 \times N$. TRW-S usually performs 10 iterations and its implementation in C++ is faster than the LBP implementation in Matlab that we use. We also run LocLearn (c) (with TRW-S) with $W = 100$ and pairwise distance matrices obtained with DO algorithm. Since TRW-S requires full matrices as input we fill in the missing values in the matrix with the mean of the computed distances. We obtain for this setup a corLoc of 30% which shows that our poor results are not a result of the optimization algorithm that we use, TRW-S or LBP, but of the approximation in the sparse appearance pairwise distances.

When $W = 1000$, for LocLearn (k), due to the large number of probabilities with 0 value that are used in the LBP inference, the algorithm has a poor performance as all the windows receive the same probability in the end. The selected windows are then randomly chosen among the candidates, which explain the low localization result of 17%. For LocLearn (l) with $W = 1000$, we run our algorithm only on 8 classes out of 12 and obtained a corLoc of 29%. For the same 8 classes, LocLearn (c) obtains a corLoc of 33%. For the other 4 classes (horse left/right, bicycle left/right) we were not able to run LocLearn (k) due to memory limit restrictions since they had more images than the other classes.

From Fig. 5.3 we see that the pairwise potentials that enter in the final optimized energy are not always among the smallest in their corresponding matrix. Increasing W, we increase the sparsity of the matrices.

This eliminates some possible configurations which might have been selected by the optimization algorithm, and thus we obtain worse localization results.

5.4 Discussion

We propose hybrid versions of the LocLearn framework which reduce the memory requirements for storing the pairwise appearance potentials for GIST and thus it allows us to run LocLearn with more candidate windows per image. The savings in memory that we obtain are bigger than 50%.

We compute only the smallest pairwise distance between two sets of descriptors coming from different images using DO algorithm.

The algorithm uses the correlation between the spatial overlap and appearance distance of window descriptors. For this, we first derive an upper bound $\mathcal{B}$ on the appearance distance as a function of the overlap. The bound can be either exact or estimated statistically from training data. We derive an exact theoretical bound for normalised CHIST which can provide intuitions on how to compute theoretical bounds for other histogram based descriptors.

We obtain large savings in memory by using sparse pairwise distance matrices at the cost of a worse localization performance. Our sparse matrices contain only the top $k\%$ smallest elements from the full distance matrix. However, the pairwise potentials which enter in the final energy returned by the optimization algorithm are not always among the smallest. LocLearn looses from the benefit of using a lager number of sample windows per image because this leads to sparse appearance distance matrices which do not model properly the dependencies in the data.

Chapter 6

Overview of hybrid LocLearn algorithms

LocLearn is limited by the number N of images and the number W of sampled windows per image. In the previous sections, we proposed three hybrid versions which build on top of the LocLearn framework with the aim of reducing its time and/or memory complexity. Table 6.1 summarizes the complexities and performance of each algorithm relative to LocLearn. The memory complexity is expressed in doubles (64 bits). We review here the notations from Table 6.1:

- D is the sum of the dimensionalities of the descriptors used.
- W_s is the number of windows in a subproblem of Divide and Conquer LocLearn.
- B is the number of bits used for encoding a descriptor.
- $s\%$ is the average sparsity of a pairwise distance matrix obtained using DO algorithm [3].

Method	Memory complexity unary + pairwise potentials	Time complexity pairwise potentials + inference	Performance
LocLearn	$O(NWD) + O(N^2W^2)$	$O(N^2W^2) + O(N^2W^2)$	reference
Divide and Conquer LocLearn	$O(NWD) + O(N^2W_s(W + W_s))$	$O(N^2W_s^2) + O(N^2W_s^2)$	+
Binary GIST LocLearn	$O(NW\frac{B}{64}) + O(N^2W^2)$	$O(N^2W^2) + O(N^2W^2)$	-
Sparse/Mean Sparse LocLearn	$O(NWD) + O((1 - s\%)N^2W^2))$	$O((1 - s\%)N^2W^2) + O(N^2W^2)$	- -

Table 6.1: **Summary of the hybrid LocLearn algorithms.**
The table presents the time and memory complexities of the hybrid LocLearn algorithms. Their performance (in terms of corLoc) is reported relative to that of normal LocLearn. + means the performance increases, - it slightly decreases, - - it decreases more (see text for details).

The memory complexity is determined by the storage requirements of the unary and pairwise potentials of the energy function 2.1. The time complexity is determined by the computation of the pairwise potentials and the inference problem in the localization step of LocLearn. In Table 6.1 we do not include the time complexity of the unary potentials. We compute the unary potentials in a preprocessing step which in practice can be very expensive depending on the cues we are using. The complexity is of $O(NWC)$, where C is the time needed to compute the descriptors for one window.

We review below the problems that each hybrid algorithm tackles.

- **Divide and Conquer LocLearn** - Following a D&C paradigm and making use of the independence of the subproblems that can be run in parallel, we reduce the time complexity of LocLearn and its memory requirements of the pairwise potentials. Moreover, D&C LocLearn with W sampled windows per image and W_s windows in a subproblem performs better than LocLearn with W_s windows per image.

- **Binary GIST LocLearn** - The algorithm reduces the memory storage of the unary potentials at a cost of a slightly decrease in performance. The unary potentials use features encoded in B bits. The time complexity is the same as that of LocLearn.

- **Sparse/Mean Sparse LocLearn** - The algorithm uses Loopy Belief Propagation for sparse distance matrices obtained using DO algorithm [3]. The new approach reduces both the computation and the memory requirements of the pairwise potentials. However, this comes at a cost of a large decrease in performance.

Chapter 7

Conclusions

We proposed three methods for addressing the problems that restrict LocLearn in using more input images and/or more candidate windows per image, with the ultimate goal of increasing its performance by processing more images and/or windows. While the initial algorithm did not use more than 100 candidate windows per image and more than 50 images per class due to its complexity, we were able to run experiments also with 500, 1000 and even 10000 candidate windows per image. The novel methods modify the LocLearn framework with the aim of reducing the storage memory of the appearance potentials (unary or pairwise) or the time complexity.

In a first approach, following a Divide and Conquer paradigm, we were able to run LocLearn with a number of candidate windows which is in the order of thousands, a setting which is infeasable in a normal LocLearn setup. By running the intermediate subproblems in parallel, we reduced the quadratic time complexity in the number of windows per image. Due to the large number of candidate windows our algorithm can choose from, we outperformed LocLearn [7] on the Pascal07 6x2 dataset. Another advantage of D&C LocLearn is that it uses less memory for storing the pairwise potentials, since it computes pairwise distances only between windows from the same subproblem.

In a second approach, we aimed at reducing the storage requirements of the unary pairwise potentials for GIST. This was done by binary encoding the features using a LSBC Hamming embedder, such that descriptors that are similar in the Euclidean space map to similar binary codewords in the Hamming space. We therefore modified the computation of the appearance distances in the LocLearn framework to account for the use of binary descriptors. Although we experimented only with GIST descriptors, the embedding does not depend on the distribution of data and can be applied to arbitrary features. The use of binary encoded GIST features accounted only for a small loss in performance.

In a third approach, we reduced both the computation time and the memory requirements of the appearance pairwise potentials by using sparse distance matrices for GIST. This allowed us to run LocLearn with 100, 500 and 1000 candidate windows per image. We modified Loopy Belief Propagation algorithm to deal with sparse matrices, which were efficiently computed using the DO algorithm [3]. The central idea of the algorithm [3] is to exploit the correlation between the spatial overlap and the appearance distance of windows. Although the use of sparse matrices allowed us to use more candidate windows, the performance of the algorithm decreased more than desired.

For the last two methods, we experimented only with the GIST features due to its discriminative characteristics in the localization stage of LocLearn, but the algorithms can be used with arbitrary features.

All our methods reduce the memory requirements of the potentials (unary or pairwise) by at least half of that of LocLearn. D&C LocLearn reduces the quadratic time complexity in the number of windows

and Sparse/Sparse Mean LocLearn reduces the computation time of the pairwise potentials. Due to their modularity, any of the above algorithms can be combined to contribute to further reductions in memory or time complexity. For instance, we can obtain memory savings for both unary and pairwise potentials if we use D&C LocLearn with binary encoded features.

The problems that we addressed in this work are related mostly with the potentials term of the energy function 2.1. In practice, any CRF model which incorporates unary or pairwise potentials defined on appearance distances can benefit from the above approaches.

Bibliography

[1] Oliva A. and A Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. In *ICVJ*, 2001.

[2] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR 2010*, June 2010.

[3] B. Alexe, V. Petrescu, and V. Ferrari. Exploiting spatial overlap to efficiently compute appearance distances between image windows. In *NIPS, In Press*, 2011.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc J. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[5] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11), 2001.

[6] Ondrej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *CVPR*, 2007.

[7] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *ECCV*, volume 6314 of *LNCS*, pages 452–466. Springer, September 2010.

[8] Piotr Indyk and Milan Ruzic. Near-optimal sparse recovery in the l1 norm. In *FOCS*, pages 199–207, 2008.

[9] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, 2006.

[10] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV (3)*, pages 82–96, 2002.

[11] Dalal N. and Triggs B. Histogram of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[12] J. Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. 1982.

[13] Lazebnik S. Raginsky M. Locality sensitive binary codes from shift-invariant kernels. 2009.

[14] Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.

[15] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.