# Robust Duplicate Detection of 2D and 3D Objects

Peter Vajda, Ivan Ivanov, Lutz Goldmann, Jong-Seok Lee and Touradj Ebrahimi

Multimedia Signal Processing Group – MMSPG

Institute of Electrical Engineering – IEL

Ecole Polytechnique Fédérale de Lausanne – EPFL

CH-1015 Lausanne, Switzerland

{peter.vajda, ivan.ivanov, lutz.goldmann, jong-seok.lee, touradj.ebrahimi}@epfl.ch

## Abstract

In this paper, we analyze our graph-based approach for 2D and 3D object duplicate detection in still images. A graph model is used to represent the 3D spatial information of the object based on the features extracted from training images so that an explicit and complex 3D object modeling is avoided. Therefore, improved performance can be achieved in comparison to existing methods in terms of both robustness and computational complexity. Different limitations of our approach are analyzed by evaluating performance with respect to the number of training images and calculation of optimal parameters in a number of applications. Furthermore, effectiveness of our object duplicate detection algorithm is measured over different object classes. Our method is shown to be robust in detecting the same objects even when images with objects are taken from very different viewpoints or distances.

Keywords: object duplicate detection, graph, SIFT, retrieval, computer vision

## Introduction

With the technological evolution of digital acquisition and storage technologies, millions of images and video sequences are captured every day and shared in online services such as Facebook, Flickr, and Picasa. As keyword-based indexing is very time consuming and inefficient due to linguistic and semantic ambiguities, content-based image and video retrieval systems have been proposed (Vajda, Dufaux, Minh, & Ebrahimi, 2009), which search and retrieve documents based on visual features. Within such systems, a query document is compared to all the documents in the database by making use of content-based features extracted from it. However, since the features are extracted from images which contain two-dimensional projections of three-dimensional scenes, the features may change significantly depending on the view point. Thus, systems often fail to retrieve relevant content in response to some queries.

In general, content-based image retrieval can utilize different representations for describing the image content, including global descriptors, feature points, or regions. Recently, interest has turned towards higher-level representations such as objects. Given a query image containing an object, an image retrieval system can perform two tasks: object recognition or object duplicate detection. Object recognition aims at finding all the instances of a certain object class (such as cars, or shoes), while object duplicate detection represents a more specific task of finding only a particular sample of that object class (such as "red Citroen C3 car" or "white Converse sneakers"). Figure 1 illustrates the relationship between object duplicate detection and object recognition problems. Therefore, within a complete system object recognition is usually applied first to detect a relevant class of objects (e.g. faces, cars) and then object duplicate detection is used to find a specific instance of that object class. Our object duplicate detection system is able to fulfill both tasks together.

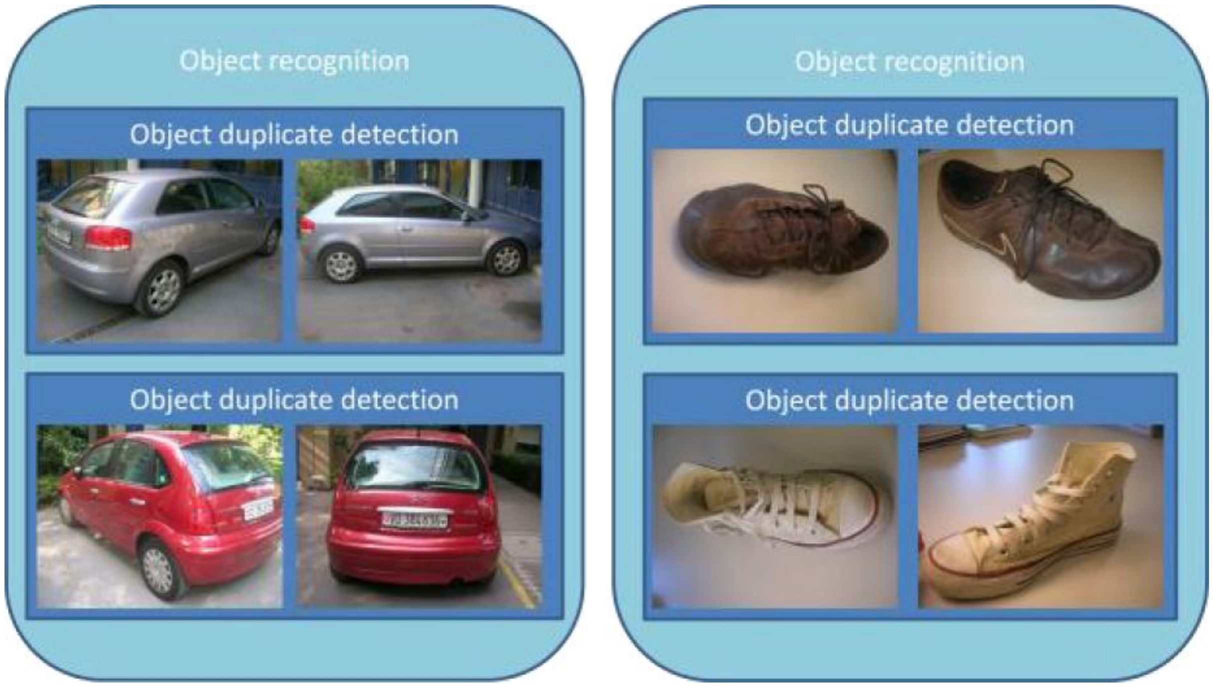
Object recognition
Object duplicate detection
Object duplicate detection
Object recognition
Object duplicate detection
Object duplicate detection

*Figure 1: Illustration of relationship between object recognition and object duplicate detection. While the former groups objects into different classes such as cars and shoes, the latter distinguishes between specific shoes or cars.*

In this paper, we are focusing on the object duplicate detection task. The general goal is to detect the presence of a target object in a set of images based on an object model created from a small set of training images. Duplicate objects may vary in their perspective, have different sizes, or be modified versions of the original object after minor manipulations, which do not change their identity. Therefore, object duplicate detection should be robust to changes in position, size, view, illumination, and partial occlusions.

A large number of applications can benefit from object duplicate detection. For example, in the popular photo sharing websites, untagged images can be automatically annotated based on the detection of the same objects from a smaller set of images with associated tags. Also, object duplicate detection may be used to search a specific object in a large collection, such as a suspect car in a video surveillance database. Moreover, when a user takes a picture of an object with his/her mobile phone, additional information about the object can be retrieved from the web, such as the price of a product, or the name and location of a monument.

In this paper, we analyze an earlier proposed graph-based approach (Vajda, Dufaux, Minh, & Ebrahimi, 2009) for 3D object duplicate detection in still images. This approach combines the efficiency of a bag of words model with the accuracy of a part-based model, which are described in the Related work section, i.e. we make an attempt towards 3D modeling, while keeping the efficiency of 2D processing. A graph model is used to represent the 3D spatial information of the object based on the features extracted from the training images so that we can avoid explicitly making a complex 3D object model. Therefore, improved performance can be achieved in comparison to existing methods in terms of robustness and computational complexity. Another advantage of our method is that it requires only a small number of training images in order to build a robust model for the target object. Several images from different views are needed to create 3D model. However in our approach, just few common features are necessary to link spatial graphs from different views; therefore fewer training images are needed for the model creation. The method is evaluated through a comprehensive set of experiments, in which an in-depth analysis of its advantages and limitations is performed and optimal parameters are derived from such an analysis. A comparison with two representative methods shows its considerable performance improvement.

The remaining sections of this paper are organized as follows. We introduce related work in the next section. Then, we describe our approach for object duplicate detection in more detail. Next, experiments and results are shown. Finally, we conclude the paper with a summary and perspectives for future work.

## Related work

### *Feature extraction*

The first important step of object duplicate detection is to extract salient features from given images. Features for object duplicate detection can be grouped into global and local. Global features usually describe an object as a whole, while local features are extracted from special parts of the object, such as salient regions. One of the best global features is the Histogram of Oriented Gradient (HOG) (Felzenszwalb, Mcallester, & Ramanan, 2008). Global features usually require an exhaustive search of the whole image over various scales and sizes to localize the target object. Thus, it is more time consuming when compared to a search using local features, which are typically scale and rotation invariant. The extraction of these local features usually consists of two steps. First, the salient regions are detected in a way robust to geometric transformations. Second, a region description for each of the detected region is generated to make them distinguishable. Reliable region detectors robust to illumination and viewpoint changes consider affine covariant regions (Mikolajczyk, et al., 2005), known to be scale, rotation and translation invariant. Among the most commonly used region descriptors is the Scale Invariant Feature Transform (SIFT) (Lowe, 2004), which is based on an approximation of the human perception. A faster version of SIFT descriptor with comparable accuracy, called Speeded Up Robust Features (SURF), is proposed in (Bay, Tuytelaars, & Gool, 2006).

## *Object representation*

Regarding object representation, one can generally distinguish between spatial and non-spatial approaches. The latter does not consider any spatial information with respect to the object and its individual parts, such as the Bag of Words (BoW) model, which is based on a histogram of local features (Fei-Fei & Perona, 2005). Zhang et al. (Zhang, Marszalek, Lazebnik, & Schmid, 2007) presented a comparative study on different local features on texture and object recognition tasks based on global histogram of features. This method gives a robust, simple, and efficient solution for recognition without considering the spatial information of the object. The advantage of our object representation over the BoW model is that spatial information from the object is considered using a graph representation. However, the graph modeling adds to the complexity.

On the other hand, spatial models such as the Part-Based Model also consider the spatial information of objects for improved performance. Connections between different parts of an object can be achieved using different structures. A star structure has been used to represent objects based on HOG features (Felzenszwalb, Mcallester, & Ramanan, 2008). Another common way for considering spatial relationships between individual parts is to employ graph models often referred to as structural pattern recognition (Neuhaus & Bunke, 2006). These graph matching approaches have been successfully applied to handwriting, character, and contour-line recognition. A generative model based on graph matching is the Random Attributed Relational Graph (RARG), which is able to capture the structural and appearance characteristics of parts extracted from objects (Zhang & Chang, 2006). However these methods are more suitable for object recognition and they need several training images for training the object model.

Most of the object representations consider the objects in the 2D image space only. However, since real-world objects are inherently 3D a higher performance can be achieved using 3D models. However, the creation of complete 3D models requires a large number of images from all possible angles, which may not be available in real applications. However, some interesting solutions have been proposed for multi-view retrieval of objects from a set of images or video. An approach described in (Sivic, Schaffalitzky, & Zisserman, Object level grouping for video shots, 2006) uses tracking to retrieve different views of the same object and to group video shots based on object appearance. The model is then used to recognize objects more reliably. In (Rothganger, Lazebnik, Schmid, & Ponce, 2004) a full 3D model of the object is used for the detection of objects and model creation from video sequences. Our approach makes an attempt towards 3D modeling, while keeping the efficiency of 2D processing, using a graph model to represent the 3D spatial information so that we can avoid explicitly making a complex 3D object model.

## *Object duplicate detection*

Typically, any object duplicate detection method contains the following tasks: feature extraction, object representation, similarity measurement and searching tasks. In this section we review representative object duplicate detection methods based on the previously described tasks.

Local features are used for object duplicate detection in (Lowe, 2004). The General Hough Transformation is then applied for object localization. Furthermore, pose is estimated using the RANSAC algorithm. Our object duplicates detection method is based on this algorithm and the detection accuracy is improved by using our spatial graph matching method. Our method is also extended by considering more training images. Therefore, 3D objects can be detected with more accuracy.

In (Sivic & Zisserman, 2006), descriptors are extracted from local affine-invariant regions and quantized into visual words, reducing the noise sensitivity of the matching. Inverted files are used to match the video frames to a query object and retrieve those which are likely to contain the same object. However, this work considers only 2D objects, such as posters, signs, ties, and does not take into account real 3D objects. In this paper an analysis of this method for real 3D objects is provided.

An extension (Sivic, Schaffalitzky, & Zisserman, 2006) of this approach uses key-point tracking to retrieve different views of the same object and to group video shots based on the objects appearance. The tracked object is then used as an implicit representation of the 3D structure of the objects to improve the reliability of the object duplicate detection. This method has proven to be more effective than a query with a single image, but it requires that all the relevant aspects of the desired object are present in the query shot, which limits its applicability.

A useful application of object duplicate detection is presented in (Gool, Breitenstein, Gammeter, Grabner, & Quack, 2009). Images coming from webcams on large database are automatically annotated with bounding boxes on object level. They also deal with searching in large database.

Detecting buildings in a large database is presented in (Philbin, Chum, Isard, Sivic, & Zisserman, 2007). The BoW method is applied for preselecting images from the large database and an efficient spatial verification is considered for further analysis. The database contains up to one million images. To resolve the problem of large database they use a forest of 8 randomized k-d trees as a data structure for storing and searching features.

## Object duplicate detection algorithm

In this section, the 3D object duplicate detection algorithm, which was first introduced in (Vajda, Dufaux, Minh, & Ebrahimi, 2009), will be described in more detail. The goal of this algorithm is to detect the presence of target objects and to predict their location in a set of images based on an object model created from training images. In this method, SIFT features are used to increase the detection speed in large databases thanks to existence of efficient search methods for local features. We target 3D object duplicate detection by using a graph model, which imposes spatial constraints between features and between the different viewpoints of the whole object to improve the accuracy of the object duplicate detection.

Since objects are considered as 3D objects, in principle more than one training image is needed to create a reliable 3D model for an object. However, as the spatial graph model is only an approximation of a 3D model with multiple 2D models (views) linked with each other, fewer training images (including only one image) are sufficient for model creation. In fact, we will show in the experimental section that only very few images are needed to create a reliable model for 3D object detection.

The system architecture is shown in Figure 2. In the training phase, an object model is created from a set of images containing this object. In the testing phase, this object model is used to detect objects and to predict their locations and sizes in test images. Each of the two phases starts with the same feature extraction step, which is shown separately. In the following sections, the feature extraction, the training, and the testing phases are explained in detail.

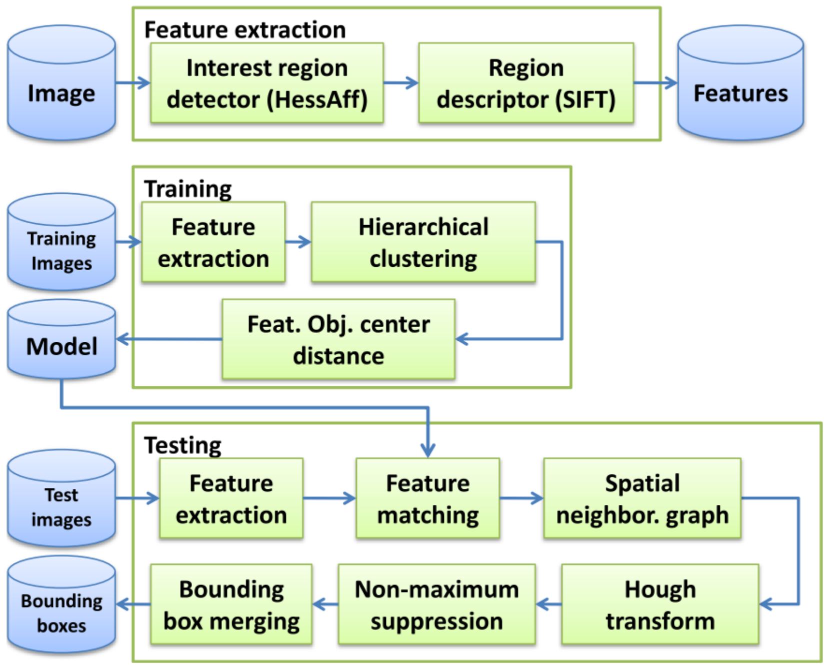

*Figure 2: Overview of the object duplicate detection approach with the individual training and testing stages, and the commonly used feature extraction step.*

## *Feature extraction*

There are mainly two kinds of features: global and local. Global features represent the whole object and usually require an exhaustive search of the whole image over various scales and sizes to localize the target object. Thus, it is more time consuming when compared to a search using local features, which are typically scale and rotation invariant.

The extraction of these local features usually consists of two steps. First, the salient regions are detected in a way robust to geometric transformations. Second, a region description for each of the detected region is generated to make them distinguishable. In our case, these steps are also separated as shown in the pseudo code of the feature extraction:

```
F = Feature extraction(i:image)
IR := Hessian affine detector (i);
    IR_i:(p:position, o:orientation, s:scale), interest region on image i.
F := SIFT(IR);
    where F_i: (IR_i, f:feature descriptor), feature.
```

1. Sparse local features are used to resolve the object localization problem more efficiently. *Interesting regions* are extracted using the Hessian affine detector (Mikolajczyk & Schmid,

2002), as it has been shown to outperform other detectors (Mikolajczyk, et al., 2005) dealing with scale and view point changes. Based on a combination of corner points detected by an Hessian matrix, multi-scale analysis through the Gaussian scale-space, and affine normalization using an iterative affine shape adaptation algorithm, the Hessian affine detector finds regions which are stable across different scales, rotations, translations, as well as changes in illumination and viewpoint. The position, scale and orientation are computed for each of the regions and will be used within the graph model.

2. To *describe* the detected regions, features based on the Scale Invariant Feature Transform (SIFT) (Lowe, 2004) are extracted, as they remain robust to arbitrary changes in viewpoints. The idea of this algorithm is to approximate the human visual perception mechanism through features that share similar properties with the neurons in the inferior temporal cortex used for object recognition in primate vision systems (Serre, Kouh, Cadieu, Knoblich, Kreiman, & Poggio, 2005). High contrast candidate points and edge response points are processed for each region and dominant orientations are considered as in the inferior temporal cortex. These steps ensure that region descriptors are more stable for matching.

## *Training*

During the training phase, a set of training images containing the target object from different views is processed, and a model for the object is constructed. The training images correspond to a single object filling up the whole field of view. Therefore, we assume that the object is positioned around the center of training images, whose boundaries are used as the bounding box of that object. In the following, each step of the training phase is described. The pseudo code of the training phase is the following:

```
Model = Training(I:images)
F := Feature extraction (I);
Storage := Hierarchical clustering(F);
Model := Spatial neighborhood graph creation(F);
        where Model:(G:Graph, A:Graph attributes)
        G:(V:feature nodes, E: edges), edges are between the close features in spatial domain.
        A(E): (d:distance, o:orientation), attributes of the edges.
        A(V):(p:position, o:orientation, s:scale), relative position, orientation and scale to the
        object center and object scale.
```

1. *Features are extracted* from the training images, as described in the previous subsection.
2. *Hierarchical clustering* is applied to the features, to group them based on their similarity. This improves the efficiency of the feature matching by adopting a fast approximation for the nearest neighbor search. More specifically, hierarchical k-means clustering (MacQueen, 1967) is used to derive the vocabulary tree, similar to that described in (Nister & Stewenius, 2006). It is a balanced tree whose inner nodes correspond to the feature of the cluster centers derived from all its children. The balanced tree is a tree where there are no big differences between the depths of the leaves. The computational complexity of finding a nearest neighbor for a given feature in this tree structure is significantly less than an exhaustive nearest neighbor search. However, it is important to mention that this approximation may occasionally cause erroneous matches, which are discarded by the further validation process.
3. Finally, the *spatial graph model is constructed* from the features and their positions. The nodes of the graph are the features of the training images. Each node also stores the scale, and orientation of the corresponding region of the feature and the relative position from the object center, normalized by the size of the object. The edges of the graph are the spatial nearest neighbors of two features. The attributes of edges are the distance and orientation of the neighboring nodes. These attributes are important for the matching step in the testing phase.

## Testing

In the testing phase, the graph model derived from a small set of training images is used to detect similar objects in other test images and to predict their bounding boxes. The whole testing phase is illustrated in Figure 3 and its individual steps will be described in more detail below. The pseudo code of the training phase is the following:

```
Bounding boxes = Testing(i:image, Model)
F := Feature extraction (i);
Matches_feature := Feature matching(F, Model.G.V) ∩ Feature matching(Model.G.V, F);
        Matches:{(f_query:feature, f_model:feature)} one to one matching using Storage.
Query := Spatial neighborhood graph creation(F);
         Query:(G:Graph, A:Graph attributes)
         G:(V:feature nodes, E: edges), edges are between the close features in spatial domain.
         A(E): (d:distance, o:orientation), attributes of the edges.
         A(V):(p:position, o:orientation, s:scale)
Matches_graph := Graph matching(Query, Model, Matches_feature)
```

```
        $Matches_{graph}$ = {($V_{query}$x$V_{model}$:feature matches, $E_{query}$x$E_{model}$:edge matches)}
Bounding boxes := General Hough Transformation(Query, Model, $Matches_{graph}$)
        Bounding boxes = {(c:coordinates, $F_{query}$:Features)}
Bounding boxes := Non-maximum Supression(Bounding boxes)
Bounding boxes := Bounding box merging(Bounding boxes, $Matches_{graph}$)
```

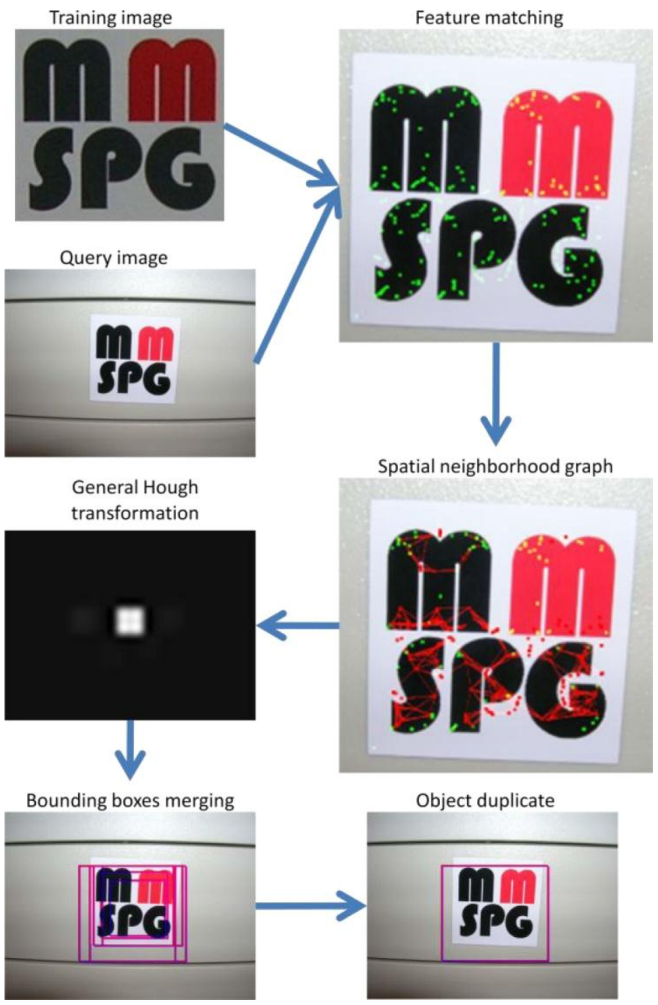

*Figure 3: Illustration of the different steps of the testing stage with individual feature matching, spatial graph matching and bounding box estimation.*

1. *Features are extracted* from the query image, as described before, which results in several robust region descriptors.
2. These features are matched to those in the graph model derived from the training images using a one to one nearest neighbor *matching* which is illustrated in Figure 4. First, for every feature from the test images the nearest neighbors in the graph model are determined based on the

Euclidean distance. Then, for each feature from the graph model the best matching feature in the test image is searched back. If it leads to the original feature in the test image then this match is a one to one match, otherwise the matching features are not corresponding to each other and they are discarded. Furthermore, matches with a distance larger than a predefined threshold $T_d$ are also discarded. This procedure ensures the selection of very reliable matching features. Due to the tree representation of features as described before, the complexity of this matching step is $O(N \log N)$, where $N$ is related to the number of features.

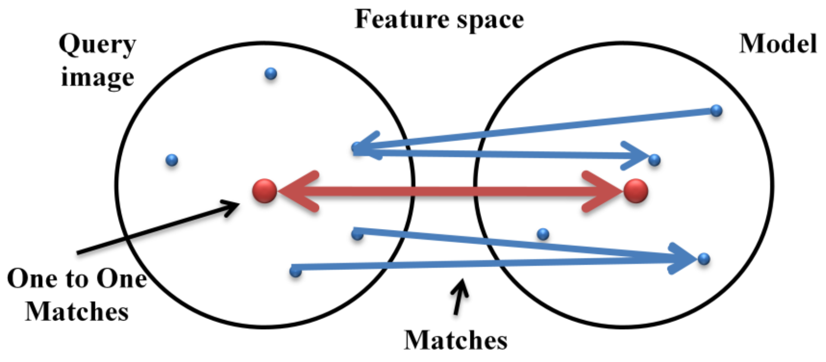

*Figure 4: One to one feature matching.*

3. A *spatial graph is constructed* from the features and their positions from the query image applying the same method as the graph creation in the training phase. Graph matching is applied between the graph model derived from training images and the graph created from the query image. Figure 5 shows the graph matching step. The graph contains spatial information from the potentially matching objects, hence making the algorithm more robust. The previous feature-matching step creates the connections between these two graphs, as shown with red and blue lines in Figure 5. First, on the graphs we match the edges. To avoid wrong connections, only those edges which have similar properties relative to the object size are matched, as shown with a red line in Figure 5. An edge on the query graph is matched with an edge on the model graph, if the normalized scales of both end of the edge are similar. More precisely, the edges are matched if the ratio between normalized scales is between $T_R$ and $1/T_R$, where the threshold $T_R$ is set manually. The scale values are obtained in the feature extraction step as described in subsection Feature extraction. The normalized scale is the scale of the feature divided by the scale of the matched feature. Second, the number of edges that go out from each node is examined and only the nodes having at least $T_{outdeg}$ edges are accepted.

This produces a rather robust graph of objects. An expected property of this graph is that, ideally, nodes of a given edge should not be part of two different objects (i.e. each edge is a part of only one object). However, it is possible to have more than one graph per object. With this method, several mismatched features can be discarded, thanks to the spatial position in the object and the normalized scale of features.

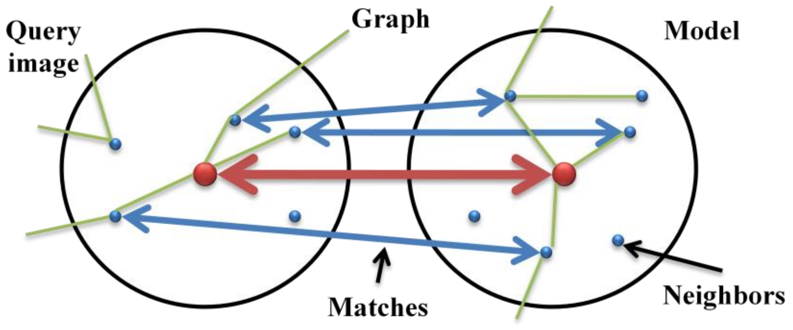

*Figure 5: Spatial neighborhood graph matching between the model graph and the query graph. Red line shows finally matched features.*

4. To estimate the bounding boxes of the detected objects, the *General Hough Transform* is applied on the nodes of the matched graph (Ballard, 1981). Each node in this graph votes for the center and the size of the bounding box in the query image, using the orientation and scale of the extracted features. The weight of each vote corresponds to the number of edges connected to the node. This creates a histogram of the object's center and its scale. Then, the local maxima are searched for in the obtained histogram. A general threshold $T$ is then applied on these local maxima.
5. As the above described procedure may return several bounding boxes with spatial overlaps, *non-maximum suppression* is applied to discard duplicate bounding boxes (Neubeck & Van Gool, 2006). This method leads to good results when using one training image; however if different views of the same object in training images are used, then more separate bounding boxes can be obtained for that object.
6. Bounding boxes obtained due to the different views of an object in training images are *merged based on graph intersection* as shown in Figure 6. This is done by considering the number of edges of the graphs intersecting two bounding boxes. The ratio between the number of intersected edges and the number of edges in both bounding boxes are threshold by a parameter $T_{bb}$. Thus, if a sufficient number of edges intersect, the two bounding boxes are merged

(Warshall, 1962). Finally, each bounding box represents a target object in the test image. This graph based method solves the problem of multi-view images in which separate bounding boxes may be obtained for the same object. Even if more than one target object is present in the test image, our algorithm still successfully detects them, as each object produces a separate graph.

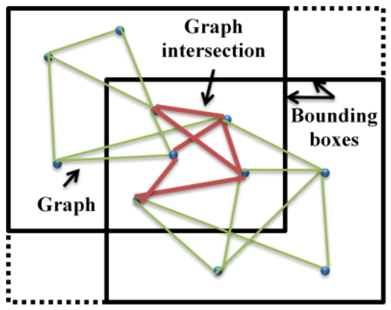

*Figure 6: Bounding box merging based on graph intersection.*

## Experimental setup

In this section the evaluation environments are assessed.

The parameter settings of our algorithm are set based on several experiments and heuristics. The number of clusters in the vocabulary tree was set to 128, as this provided a good tradeoff between complexity and accuracy. For the feature matching, the distance threshold was selected to be $T_d = 10^5$. In the spatial neighborhood graph $T_{outdeg} = 4$, to accept robust matching features. The normalized scale ratio was set to $T_R = 2$. Finally, the threshold $T_{bb} = 0.1$ was chosen to merge bounding boxes. The optimal detection threshold $T$ for the algorithm was derived based on various experiments as summarized below.

The considered dataset is described in subsection Database. Finally in subsection Evaluation the evaluation methodologies are presented.

### *Database*

A dataset including 850 images was created in order to evaluate the object duplicate detection method described in this paper. It consists of ten 3D and five 2D object classes as shown in Figure 7: bag, bicycle, body, face, shoes, stone, can, car, building, motor, poster, logo, newspaper, book and workbook.

Each of the 15 classes contains at least 3 samples. Figure 8 provides 3 images for 2 selected classes: building and shoes. As it can be seen from these samples, images with a large variety of view points and sizes are included in each class.

Angles and relative size of image point of views are calculated for each image in 3D dataset for further analysis on full, 360° duplicate detection as shown in Results section.

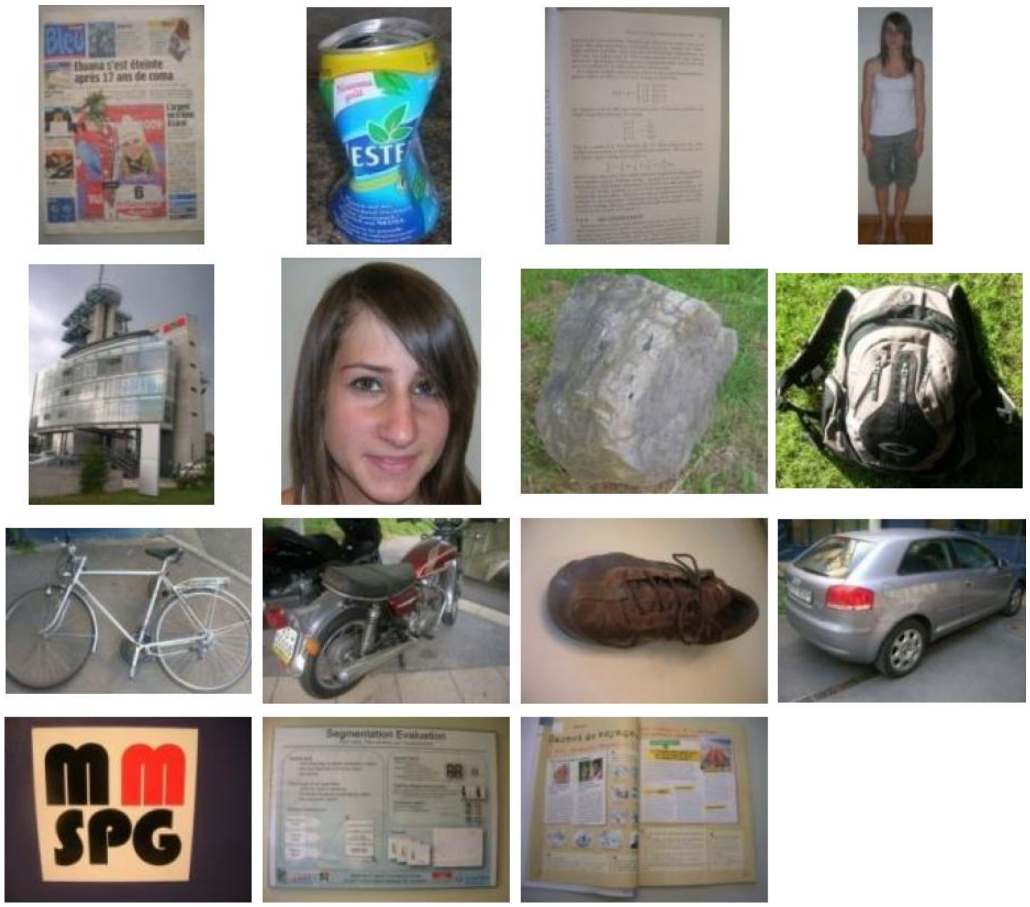

*Figure 7: Samples of the different 2D and 3D object classes within the dataset used in evaluations.*

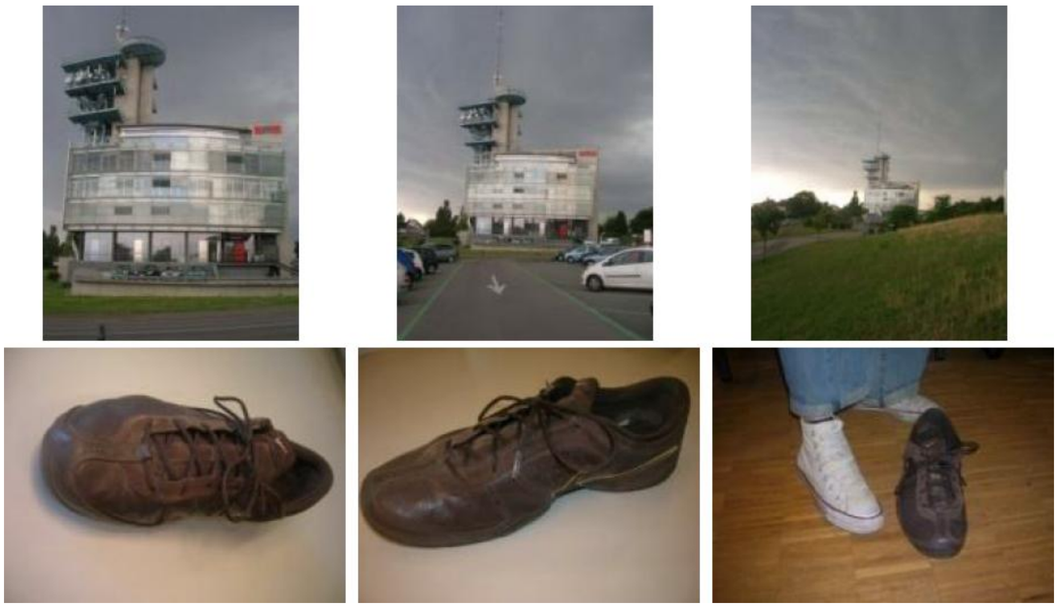

*Figure 8: Samples for two objects under diverse viewing conditions within the dataset used in evaluations.*

## Evaluation

The detection task (Fawcett, 2006) can be evaluated using correspondences between a set of predicted objects which are represented by their bounding boxes, and a set of ground truth objects. A pair-wise comparison of ground truth and predicted objects is performed in order to see if they are the same or not. The results are used to obtain the values of true positives ($TP$), true negatives ($TN$), false positives ($FP$) and false negatives ($FN$). This confusion matrix serves as a basis on which two curves can be derived.

First, the receiver operating characteristic (ROC) curve represents the true positive rate ($TPR$) versus the false positive rate ($FPR$), with:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Second, the precision recall (PR) curve plots the precision ($P$) versus the recall ($R$) with:

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

This curve does not consider $TN$ which is not uniquely defined for detection problems.

In order to determine the optimum thresholds for object detection, the F-measure is calculated as the harmonic mean of $P$ and $R$ values, given by:

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

which considers $P$ and $R$ equally weighted. The more general definition of the F-measure considers the weighted values of $P$ and $R$, and measures the effectiveness of the object duplicate detection algorithm with respect to a user who gives β times the importance to $P$ when compared to $R$:

$$F_\beta = \frac{(1 + \beta^2) \cdot P \cdot R}{\beta^2 \cdot P + R}$$

## Results and analysis

In this section we present the evaluation results of our graph-based object duplicate detection algorithm and analyze them in order to answer the following important questions:

1. What is the impact of the number of training images on the accuracy of our algorithm?
2. What are the optimal parameter settings?
3. How does the detection performance depend on the object class?
4. How does its performance compare to the state of the art?
5. How can object duplicate detection be performed reliably for any point of view?

### *Influence of the number of training images*

In a first experiment, we trained our system with one, two or three images, and tested it with the remaining images in order to analyze the performance of the object duplicate detection as a function of the number of training images. Negative images are all images which do not contain the ground truth object. In our experiments we define as negative images all different images from the same class of the object and several not related images.

Figure 9 plots the corresponding ROC curves. The results show that the performance of the algorithm varies according to the number of training images. One can notice that using more than one training image can significantly improve the performance because it makes the object model robust against different points of views. As an example, the results show that for $FPR = 0.10$ a $TPR = 0.85, 0.92, 0.97$ is achieved when the system is trained with one, two and three images, respectively.

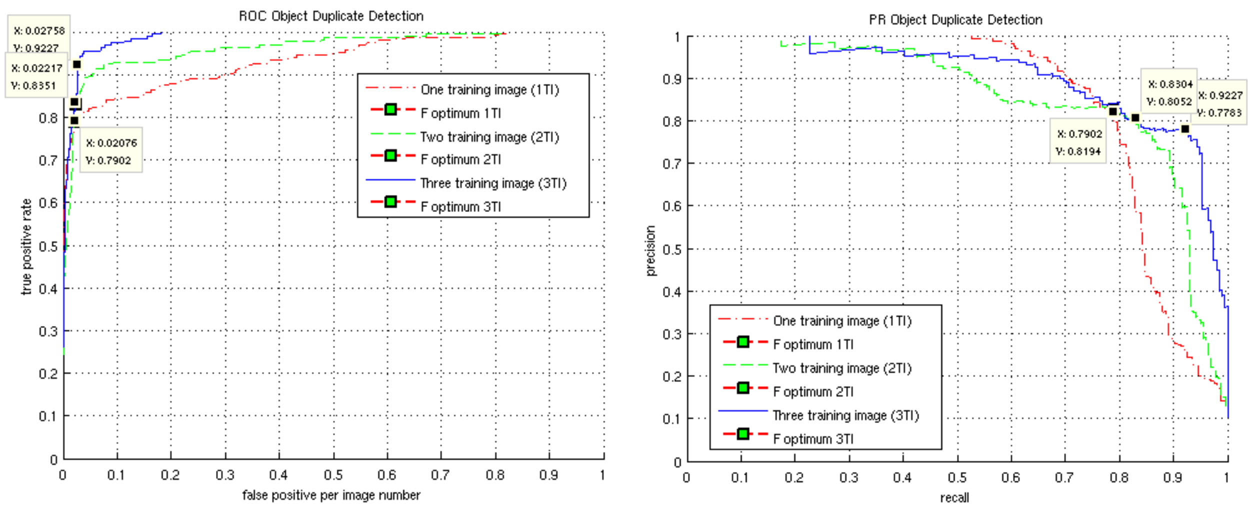

*Figure 9: Receiver operating characteristic (ROC) curves for different numbers of training images (1, 2 and 3) per object is shown on the left. A larger number of training images leads to an increased TPR for a fixed FP value. Recall versus precision curves for different number of training images (1, 2 and 3) per object is shown on the right.*

The object duplicate detection algorithm was also evaluated using PR curves as shown in Figure 9. These curves complement the previously discussed results and provide a better visualization of the opposing effects (high precision vs. high recall) which are inherent to any detection task. For instance, the results show that for $R = 0.90$ a then $P = 0.28, 0.67, 0.78$ is achieved when the system is trained with one, two and three images, respectively. For a high recall value greater than 0.8, higher precision values are obtained when more than one training image are used.

## *Optimal parameter selection*

In a second experiment, we estimated the optimal parameter settings of our system. The F-measure is calculated in order to determine the optimal threshold values for different applications.

First, the F-measures obtained for the different thresholds in the object duplicate detection approach are calculated to determine the optimal threshold value. The results are shown in Figure 10. For each of the cases where one, two and three training images are used, the maximum F-measure is found and shown with a marker in this figure. According to the results in this figure, F-measures of 0.80, 0.82 and 0.84 can be reached with thresholds equal to 51, 140 and 288, if our system is trained with one, two or three images respectively. Therefore, the optimal threshold is highly correlated with the number of training images. The optimal threshold increases significantly if more training images are used.

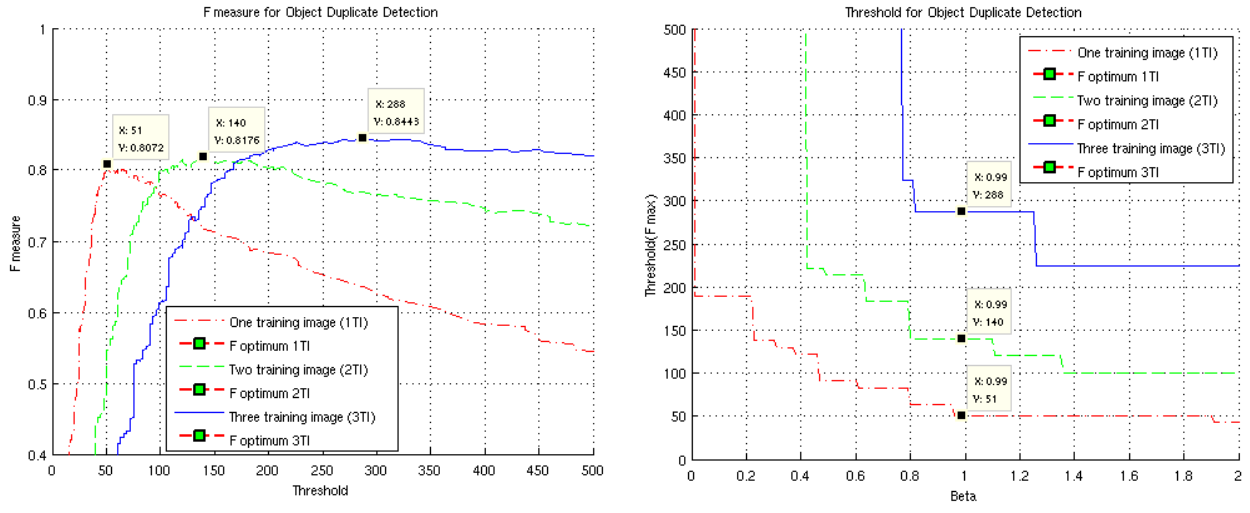

*Figure 10: F-measure versus detection threshold for different number of training images (1, 2 and 3) is shown on the left. Both the F-measure and the detection thresholds increase with a larger number of training images. Detection threshold versus $\beta$ parameter of the general F-measure for different number of training images is shown on the right.*

Figure 10 provides the parameter β of the general F-measure over different threshold values. Using the general F-measure, the importance of the precision and recall can be balanced according to the requirement of a given application using the object duplicate detection. If the β parameter is equal to one, the precision is as much important as the recall. Two other commonly used F-measures are the $F_{0.5}$ measure, which weights the precision twice as much as the recall, and the $F_2$ measure, which weights the recall twice as much as the precision. The optimal F-measure parameter settings, for which the precision and the recall are equally weighted, are also shown as markers in Figure 10.

## Performance across different classes

The goal of a third experiment was to measure the performance of the object duplicate detection algorithm as a function of different object classes. The F-measure is computed for each of the 2D and 3D object classes and different classes are compared with each other.

The results are shown in Figure 11. According to these results, there are big differences in detection performance between different classes, which are caused by various factors such as reflection properties, amount and presence of textures, or number of salient features. The object duplicate detection algorithm performs well with newspapers, thanks to the large number of pictures and textures in such objects. Duplicate detection of human bodies considers only the texture of the clothes and not their shape, which gives a surprisingly good result. Face identification is also a possible application, although its performance should be compared to those of state of the art face detection algorithms. Shiny objects, such as motor bicycles and cars, are hard to detect due to the changing reflections depending on lighting condition. Books are also among the classes showing the worst performance due to large illumination variations during the image acquisition of our dataset, which was not the case for newspapers.

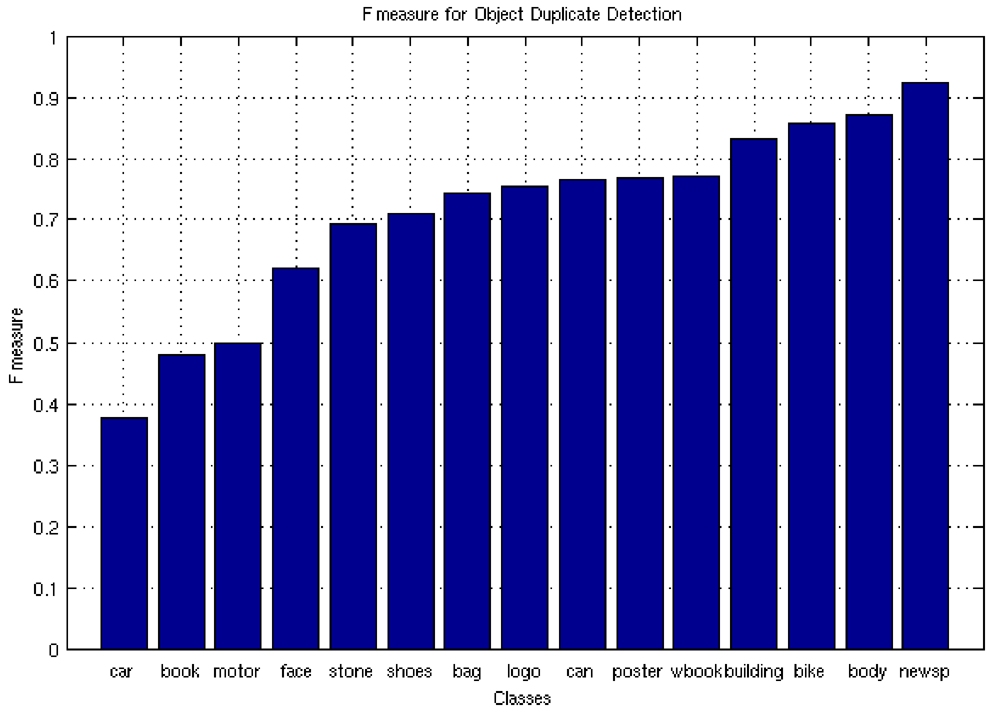

*Figure 11: Performance of the object duplicates detection in F-measure for each object class. The difference between classes is caused by various factors such as reflection properties, amount and presence of textures, or number of salient features.*

Finally, an interesting example is shown in Figure 12. The object duplicate detection algorithm is performed on the class of cars. Interestingly, even the opposite side of a car can be correctly recognized when only one training image is used, which is the case in this example. This is due to the fact that the license plate of a car is the most salient region on both the front and the back side of the car. Nevertheless, the location of the car is shifted upwards due to the different position of the license plate with respect to the overall object.

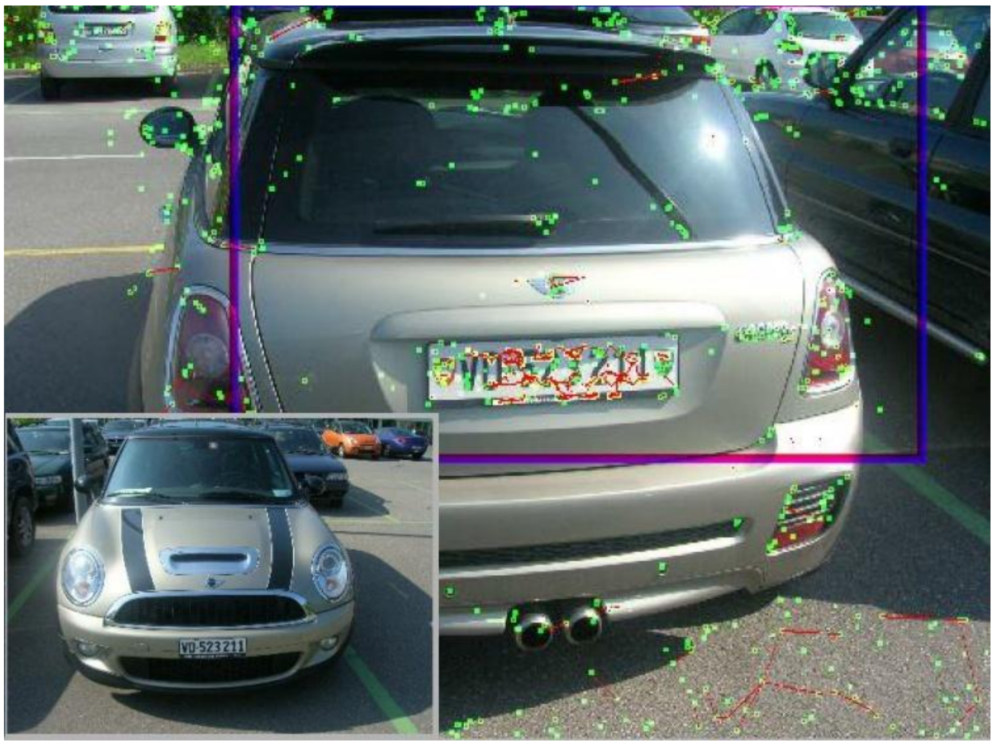

*Figure 12: An example where our object duplicate detection algorithm detects the back side of a car thanks to its license plate. The training image is shown in the bottom left corner.*

## Comparison with state of the art

The goal of the fourth experiment was to assess the quality of proposed method for object duplicate detection in relation to the state of the art. We have compared it to the BoW method (Fei-Fei & Perona, 2005) and to Lowe's object duplicate detection (Lowe, 2004).

The general drawback of the BoW method is that it does not consider any spatial information from the object; hence it creates a general histogram from local features. The resulted histograms are then compared by Euclidean, $L_1$ and histogram intersection distance (Swain & Ballard, 1991). Histogram intersection distance performed the best among them; therefore in this paper this result is presented

Lowe's object duplicate detection (Lowe, 2004) algorithm is considering the spatial information by using General Hough Transformation similar to our approach. However our method improves the quality of the matched features, due to the graph matching.

In Figure 13, ROC curves and PR curves are shown for the three methods. The results show significant differences between these algorithms. The BoW method performs worst, since it does not consider any spatial information. Lowe's method shows improved performance by considering this information. However, our method provides another considerable performance gain due to the graph matching.

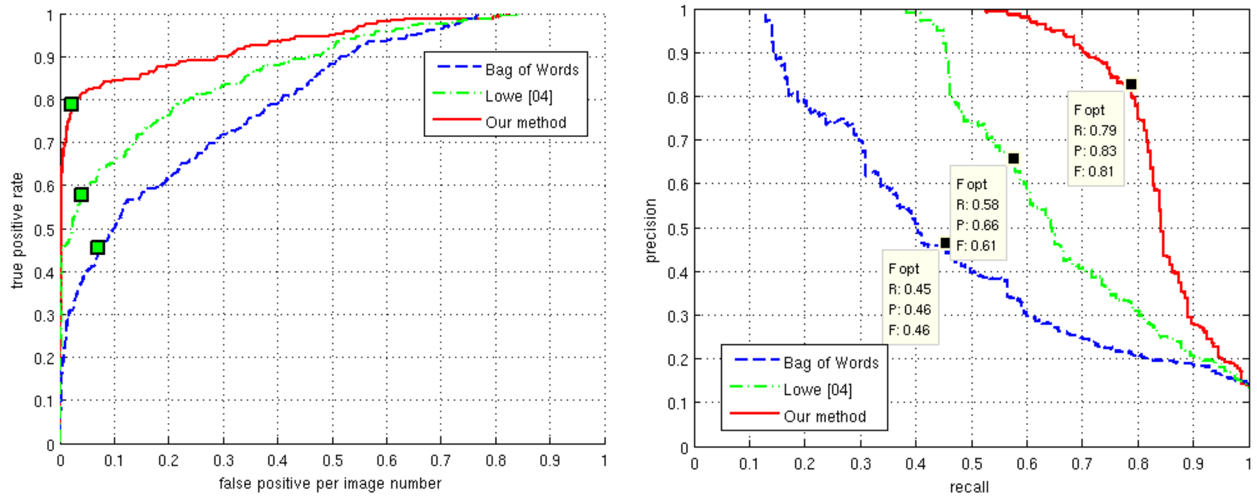

*Figure 13: Comparison of our method with BoW method and Lowe's object duplicate detection algorithm (Lowe, 2004). On the left side, ROC curve and on the right side PR curve are shown.*

For a more detailed analysis we compared the performance of the methods across the different classes shown in Figure 14. The advantage of our algorithm is observable for most of the classes. However, there are some cases where our method does not perform as good as the other methods. The BoW method works well on "cars" and "bags" compared to the other methods, since it is more robust to the spatial information changes caused by varying view point and distance. In these objects there are just a few features and the available spatial information is not very reliable. However, for objects where the spatial information is crucial such as "books", "shoes", "posters", "buildings", "bodies", "newspapers", the BoW method shows very bad results.

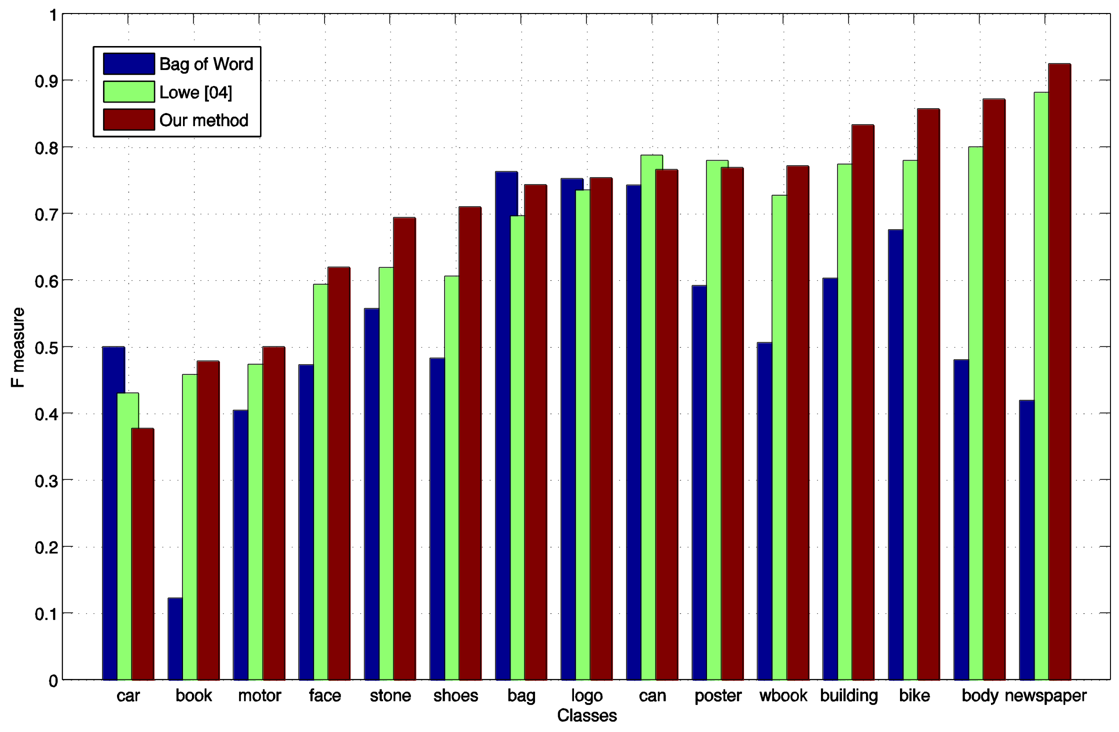

*Figure 14: Comparison is shown between our method, BOW and Lowe's object duplicate detection algorithm through different classes of objects.*

## ***Towards omnidirectional detection***

Imagine a scenario, where you make a picture from an object with your mobile phone and you would like to get feedback from an application about the object. To recognize the object you would need omnidirectional duplicate detection algorithms, where your algorithms works with 80% recall, even if the picture was taken from any direction.

In this experiment, we evaluate our object duplicate detection algorithm considering object duplicate detection from any point of view. The object duplicate detection is evaluated with test images from an object, taken from different viewing points and distances. Therefore we can analyze the recall of our object duplicate detection algorithm, depending on the viewing angle and the distance (measured as relative object size).

The results of this experiment are shown in Figure 15. It shows that the recall starts to decrease, if for the testing image the object size is less than the 75% or the viewing angle differs more than 40º from the training image. If the viewing angle differs by 90º, the recall drops down to 0.45. In contrast to previous works, this shows that real objects have to treat as convex 3D objects instead of planar 2D objects which may be detected up to 180º.

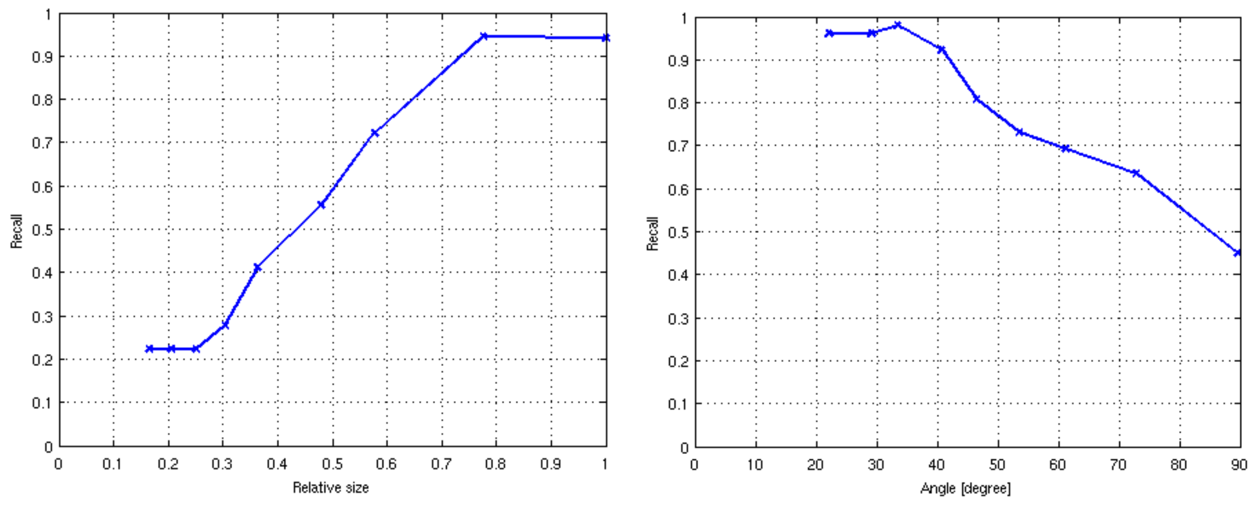

*Figure 15: Different images were taken from the object from viewing angle and distance. The left figure shows the recall measurement vs. relative size of the object in the test image. The right figure shows the recall measurement vs. viewing angle.*

Based on that, it is possible to estimate how many images from which viewing angles and distances have to be taken to achieve a certain overall recall. If we would like to detect at least 80% of the test objects using a training image, than the test images can differ up to ±47° in angle and 65% in size from the training image. If we would like to detect at least 80% of the test objects for all possible rotations around a single axis, four training images are enough. There is no need to take pictures from different distance, because a simple rescaling to 65% is enough.

The origin of problems of covering sphere with a given number of circles are come from Fejes Tóth, Hungarian mathematician, who gave exact solution for some of these problems depending on the number of circles (Fejes Tóth, 1972). However the full exact problem is still unsolved in mathematics. Considering different angles of detection, numbers of cameras is shown in Table 1, which is needed for omnidirectional object duplicate detection. The solutions for spherical coverings were gathered from (Hardin, Sloane, & Smith, 1997) webpage.

*Table 1: Covering a sphere with "#cameras" number of congruent, overlapping circles, if every point of the sphere belongs to at least one circle and if the common radius has is the minimum. Row of radius shows the radius of the circles in degree, where the centers of the circles are the cameras and the radius of the circles are the detection angle of a camera.*

| #cameras | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| radius | 70.53 | 63.43 | 54.74 | 51.03 | 48.14 | 45.88 | 42.31 | 41.43 | 37.38 | 37.07 | 34.94 | 34.04 | 32.90 |

Therefore, to cover sphere by 47° of circles, 9 training images are enough, if the ordering of the cameras are done as it is shown in Figure 16. The positions of the cameras in this case are shown in Table 2. If we would like to detect at least 90% of the test objects in any direction, than 12 images is enough to be taken as training image.

*Table 2: 9 3D coordinates of the centers of the circles, which are covering a sphere if the radiuses of the circles are 45°.*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| x | -0.039 | 0.870 | -0.013 | -0.673 | -0.577 | 0.057 | -0.858 | 0.664 | 0.568 |
| y | -0.078 | 0.266 | -0.637 | -0.726 | 0.344 | 0.992 | 0.372 | 0.268 | -0.802 |
| z | 0.996 | 0.415 | -0.771 | 0.145 | -0.741 | 0.110 | 0.356 | -0.698 | 0.187 |

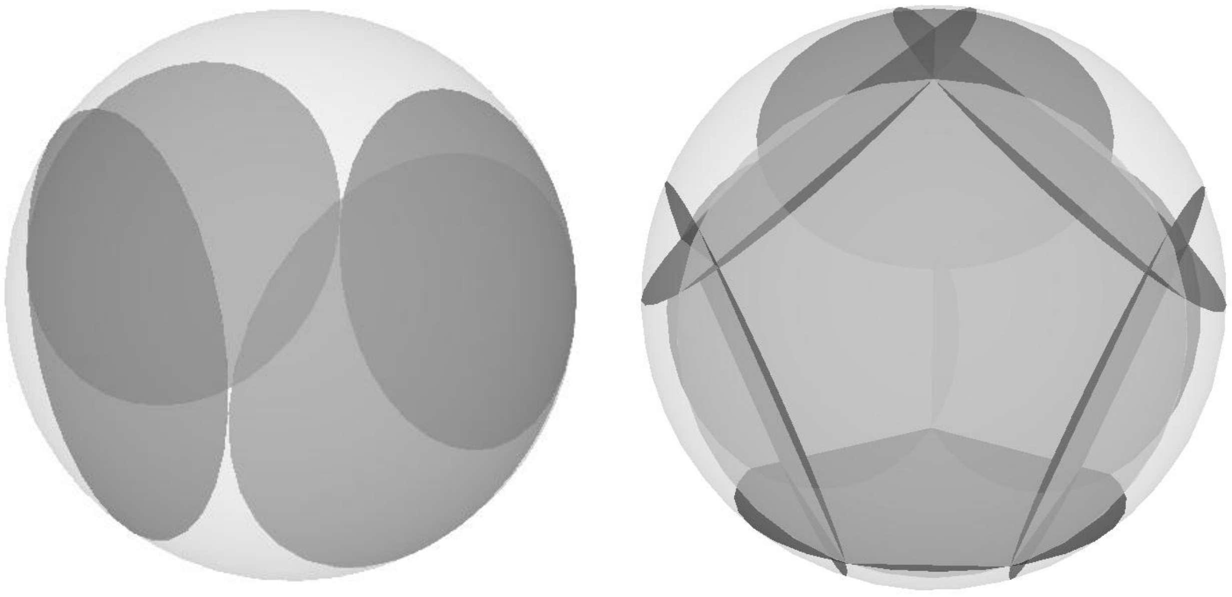

*Figure 16: The ordering of cameras is shown in the case if the detection is considered from a plane or from full sphere. The positions of the cameras are the centers of the circles and the detection area is shown as circles for each camera.*

## Conclusion

Image and video retrieval systems are becoming increasingly important in many applications. Automatic video and image tag propagation, video surveillance, and high level image or video search are among some of the applications which require accurate and efficient object duplicate detection methods. In this work, we have analyzed a robust graph-based object duplicate detection algorithm for 2D and 3D objects. The experiments are performed on various classes of objects. The main conclusions that can be drawn from our experiments are:

- The performance of our object duplicate detection algorithm shows that more than one training image can significantly improve the detection accuracy.
- Considering F-measure and $F_\beta$-measure values, the optimal threshold was determined and found to vary with the requirement of specific applications.
- The comparison of performance between various classes of objects shows that our algorithm performs well with textured objects, while shiny object are most difficult to detect.
- We showed that our method works significantly better than Bag of Words method and Lowe's object duplicate detection method (Lowe, 2004) on our dataset, due to way the spatial information is considered through graph matching.
- Four training images are enough for 3D object duplicate detection from planar point of view and 9 training images for full detection.

Furthermore, the our method has shown to be robust in detecting the same objects even if the images with objects are taken from very different viewpoints or distances.

As future work, we will explore automatic geotag propagation in social networks based on our object duplicate detection method. We will also consider extension of the method to deal with duplicate object detection in video.

## Acknowledgments

This work was supported by the Swiss National Science Foundation Grant "Multimedia Security" (number 200020-113709), the Swiss National Foundation for Scientific Research in the framework of NCCR Interactive Multimodal Information Management (IM2), and partially supported by the European Network of Excellence PetaMedia (FP7/2007-2011).

## References

Ballard, D. H. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition , 13* (2), 111–122.

Bay, H., Tuytelaars, T., & Gool, L. V. (2006, May). Surf: Speeded up robust features. *9th European Conference on Computer Vision* .

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters* , 861-874.

Fei-Fei, L., & Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. *Computer Vision and Pattern Recognition*, (pp. 524-531 vol 2.).

Fejes Tóth, L. (1972). *Lagerungen in der Ebene auf der Kugel und im Raum, 2nd edition.* Berlin: Springer-Verlang.

Felzenszwalb, P., Mcallester, D., & Ramanan, D. (2008, June). A discriminatively trained, multiscale, deformable part model. *IEEE International Conference on Computer Vision and Pattern Recognition* .

Gool, L. V., Breitenstein, M. D., Gammeter, S., Grabner, H., & Quack, T. (2009). Mining from Large Image Sets. *ACM Int. Conference on Image and Video Retrieval.*

Hardin, R. H., Sloane, N. J., & Smith, W. D. (1997, May 30). *Spherical Coverings* . Retrieved from http://www.sphopt.com/math/question/covering.html

Leibe, B., Leonardis, A., & Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision , 77*, pp. 259-289.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision , 60* (2), pp. 91–110.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, (pp. 281–297).

Mikolajczyk, K., & Schmid, D. (2002). An affine invariant interest point detector. *Proceedings of the 7th European Conference on Computer Vision* (pp. 128-142). Copenhagen, Denmark: Springer.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., et al. (2005). A comparison of affine region detectors. *International Journal of Computer Vision , 65*, pp. 43-72.

Neubeck, A., & Van Gool, L. (2006). Efficient non-maximum suppression. *International Conference on Pattern Recognition,* (pp. 850-855).

Neuhaus, M., & Bunke, H. (2006). Edit distance based kernel functions for structural pattern classification. *Pattern Recognition , 39*, 1852–1863.

Nister, D., & Stewenius, H. (2006). Robust scalable recognition with a vocabulary tree. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (pp. 2161-2168).

Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Rothganger, F., Lazebnik, S., Schmid, C., & Ponce, J. (2004). 3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. *International Journal of Computer Vision , 66* (3), 231-259.

Rothganger, F., Lazebnik, S., Schmid, C., & Ponce, J. (2004). Segmenting, modeling, and matching video clips containing multiple moving objects. *Conference on Computer Vision and Pattern Recognition*, (pp. 914-921).

Serre, T., Kouh, M., Cadieu, C., Knoblich, U., Kreiman, G., & Poggio, T. (2005). *A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex.* Massachusetts Institute of Technology. Cambridge, MA: CBCL Paper #259/AI Memo #2005-036.

Sivic, J., & Zisserman, A. (2006). Video Google: Efficient visual search of videos. *Toward Category-Level Object Recognition* (pp. 127-144). Lecture Notes in Computer Science, Springer.

Sivic, J., Schaffalitzky, F., & Zisserman, A. (2006). Object level grouping for video shots. *International Journal of Computer Vision , 67* (2), 189-210.

Swain, M. J., & Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision* .

Vajda, P., Dufaux, F., Minh, T. H., & Ebrahimi, T. (2009). Graph-based approach for 3d object duplicate detection. *International Workshop on Image Analysis for Multimedia Interactive Services.*

Warshall, S. (1962). A theorem on boolean matrices. *J. ACM , 9* (1), 11-12.

Zhang, D., & Chang, S.-F. (2006). A generative-discriminative hybrid method for multi-view object detection. *IEEE International Conference on Computer Vision and Pattern Recognition.*

Zhang, J., Marszalek, M., Lazebnik, S., & Schmid, C. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision , 73* (2), 213-238.