Learning an efficient and robust graph matching procedure for specific object recognition

Jerome Revaud*, Guillaume Lavoué*, Yasuo Ariki† and Atilla Baskurt*
*Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
Email: firstname.lastname@liris.cnrs.fr
†*CS-17, Kobe University, Japan*
Email: ariki@kobe-u.ac.jp

***Abstract*—We present a fast and robust graph matching approach for 2D specific object recognition in images. From a small number of training images, a model graph of the object to learn is automatically built. It contains its local keypoints as well as their spatial proximity relationships. Training is based on a selection of the most efficient subgraphs using the mutual information. The detection uses dynamic programming with a lattice and thus is very fast. Experiments demonstrate that the proposed method outperforms the specific object detectors of the state-of-the-art in realistic noise conditions.**

***Keywords*-specific object recognition; cascade; graph matching;**

I. Introduction

Object recognition has been a very active topic in the past 30 years. Whereas the perfect *class* object recognition system is yet to be invented, *specific* object recognition has almost been thought to be solved with the apparition of *keypoints,* whose most famous avatar is probably SIFT [1]. Indeed, methods using keypoints present numerous advantages: they are invariant to translation, scale, rotation and occlusion without significant increase in complexity; thanks to the high descriptive power of the keypoints, any training is quite unnecessary; they are close to real-time; and finally they are simple to carry out.

Numerous works have put their interest in the *way* of detecting an object from its keypoints. A very popular one among them is RANSAC [2] as it can detect an object lost in a cluttered scene thanks to its robustness to outliers. But there are two main problems with RANSAC: firstly, it needs quite a large number of matches to ensure a detection, which may be difficult in noisy conditions; and secondly under complex transformations, the number of required iterations becomes very important. To solve those issues, Chum et al. have proposed a local optimization in the RANSAC algorithm [3]: in the main loop, a simpler transform is computed while in the local optimization the full transform parameters are inferred from the previously found inliers set. In comparison, our method also uses a simple transform but makes no assumption on the full transform by relaxing the position constraints between distant keypoints, like in [4]. Also, the assumption of RANSAC or voting methods like [1] that each model keypoint carries the same amount of information about the model often does not hold in practice: some features are very specific while some others are quite casual. On the contrary our method can estimate the importance of each model feature and choose a detection threshold fitted for each of them. We show later that when it comes to realistic noise conditions, the performances of those existing methods can drop dramatically.

On the other hand, graph matching seems to be a straightforward way to object detection. Indeed, after having extracted some salient keypoints, both model object and scene can be represented as graphs. While numerous researchers have put their efforts in finding some efficient heuristics for graph matching, we only retain here the idea of Messmer and Bunke [5] which involves growing subgraphs progressively. In order to enable a detection as fast as possible, we combined this idea with the work of Viola and Jones [6], in which the authors have cascaded several classifiers to minimize the number of examined features. In the end, our specific object recognition system can deal with noise and occlusion while still being minimalist in terms of computations.

The proposed algorithm takes as input a small collection of model and background images. It automatically gathers every model keypoints into a prototype graph (section II-B), which is then used for the recognition (section II-C). More precisely, a detection lattice is derived from it: the lattice contains several different ways of building the prototype graph by adding keypoints one by one. The case of occlusion is dealt with by pruning the detection lattice as soon as the corresponding subgraphs become specific enough to the model (section II-D). Finally, we present some results in section III and conclude in section IV.

II. Algorithm description

The algorithm is summarized in figure 1: (a) (b) first, a prototype graph is extracted from sparse local model features; (c) then, the detection lattice is derived from it so as to enable robustness to occlusion and low run-time complexity. In the context of this paper a lattice is a structure comparable to a tree, at the difference that two different

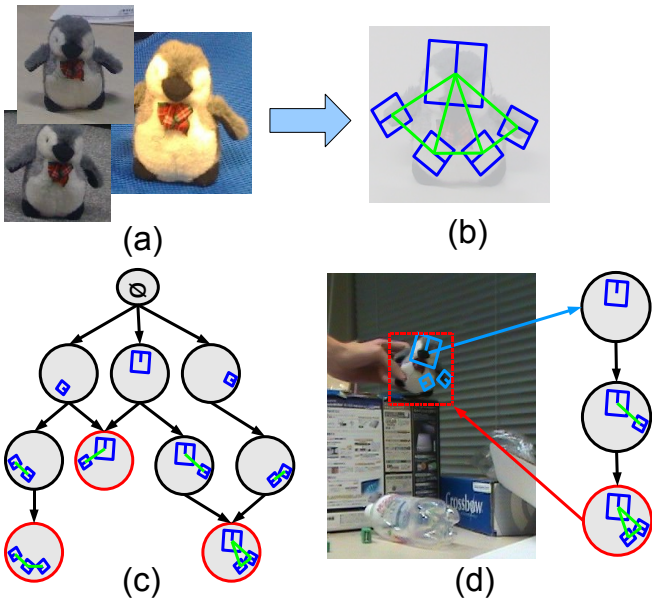

Figure 1. Summary of the proposed method (see text for details).

paths starting from the root can meet up later (but still excluding cycles since edges are oriented). (e) Finally, the lattice is used to detect the model object in test images.

A. Preliminary keypoints indexation

We detect and extract beforehand SIFT keypoints in each training image $I \in \mathbb{I}$. Each keypoint $k \in K_I$ is defined by a center $\mathbf{c}_k$, a radial vector $\mathbf{h}_k$ (i.e. scale and orientation) and a descriptor $\mathbf{z}_k$.

Search distance: Since the system will need in the following to quickly check for the presence/absence of a given keypoint at a given position in the image, we index the keypoints in a k-d tree so as to enable their fast retrieval. The *search distance* returned for a request keypoint r in image I is defined as:

$$sd(r,I) = min\{\|\mathbf{c}_r - \mathbf{c}_k\| + \alpha\|\mathbf{h}_r - \mathbf{h}_k\| \mid \|\mathbf{z}_r - \mathbf{z}_k\| < \sigma\}_{k \in K_I} \quad (1)$$

B. The prototype graph

After having aligned each model image in a reference frame, we conduct a co-occurrence analysis of every feature. Specifically, each keypoint is searched in other model images at the same relative position and the resulting *search distances* are summed. The most redundant keypoints thus own the minimum sums and are preserved while their twins (same keypoints in other images) are removed. Then, the remaining keypoints are connected together in function of their proximity. Two keypoints $k_1$and k_2 are linked if $\|c_1 - c_2\| < \|\mathbf{h}_1\|\,\|\mathbf{h}_2\|$ (we assume here the position noise to be dependent of the keypoint size, so that each graph edge ideally stands for a stable neighborhood relationship). Figure 1.(b) presents an example of a simple prototype graph.

C. The detection lattice

The aim of the detection lattice is to store various ways of building the prototype graph by adding model keypoints one by one (see figure 1.(c)). In other words, each lattice node is a connected subgraph and each lattice oriented edge represents a feature addition to the start subgraph. For this reason, the lattice can be seen as a *cascade of classifier*s, where each edge stands for a micro-classifier that bases its decision on the *search distance* of its associated keypoint: if the added keypoint can be found in the test image at the specified position (relatively to the feature position in the model image[1]), the edge is crossed, otherwise the progression is stopped.

Globally, the recognition procedure is initiated by a RANSAC-like random picking ; then the algorithm "feeds" the lattice with the chosen keypoint and tries to make it go as deep as possible. When a terminal node is reached, a vote is cast for the global model position in the test image (red square in fig. 1.(d)). Finally, votes are clustered in the scale-space and the cluster size is used to take the final decision.

D. Building the lattice

Initialization: The first lattice level (i.e. subgraphs of cardinality $l = 1$) is constituted by the N_{seed} best model keypoints in terms of redundancy[2]. This restriction comes from the fact that the complexity is roughly linear with N_{seed}, but it still gives good results when N_{seed} is low (section III-C).

Iterative pruning of the lattice: For obvious reasons, it is not possible to build the full lattice. Instead, we adopt an iterative procedure where, for each lattice level $l \geq 2$, we create every possible candidate subgraphs of cardinality l (with respect to lattice level $l-1$), train the new edges classifiers and finally retain only a limited number of the best candidates.

Classifier training: We used the mutual information as in [7] to train the micro-classifier embedded on each candidate edge e_i. Practically, the mutual information (MI) measures the correlation between two random variables. In our case, the decision function has the form $sd(r,I) < constant$ (see eq. (1)) so we only take the positive correlation (pMI) into account:

$$pMI(A;O) = \sum_{(a,o)\in\{(0,0),(1,1)\}} p(a,o).log\frac{p(a,o)}{p(a)p(o)}$$

where A and O are binary random variables which respectively denote the detection of a subgraph and the correctness of the detection. Training is done as traditionally with

[1]We use here a simple 2D similarity transform.

[2]Scale is used instead if only one training image is available.

cascaded classifiers [6]: the detection is launched both on positive and negative images with the current incomplete lattice, then for each new edge e_i, the optimal threshold sd_i^{max} is chosen as the one that maximizes the pMI. We used a Parzen window technique with a Gaussian kernel to deal with the situations where only a few training samples are available.

Global ranking: Afterward, a second optimization step is run to elect the best subset of candidates from a global point of view: a single parameter r^{max} controls the trade-off between a good spatial coverage of the model (i.e. to be able to recognize it whatever the occluded area) and a maximum detection speed. From this perspective, the candidates are ranked according to the amount of information that they can provide to the current lattice:

$$gain(A|\mathbf{M};\mathbf{O}) = MI(\mathbf{M},\mathbf{M}^A;\mathbf{O}) - MI(\mathbf{M};\mathbf{O})$$

Here, $\mathbf{M} = [r_1,\ldots,r_T]$ is a vector that contains the probabilities of detecting the different model areas with the current lattice. Practically, $\mathbf{M}$ is simply the concatenation of several discretized scale-space pyramids, one for each model image (4 scales per pyramid: 1x1, 2x2, 3x3 and 4x4 = 30 values/image) and an additional value to store the probability of false positives. Vectors $\mathbf{M}^A$ and $\mathbf{O}$ relate to the candidate subgraph A and to the binary ground truth expressed under the same form, respectively. In order for $\mathbf{M}$ to play its role of memory of the already learned areas, it is updated each time that a subgraph becomes terminal: $\mathbf{M}^A$ is added to $\mathbf{M}$ and the result is thresholded to the maximum value r^{max}.

For example, a value of $r^{max} = 0$ amounts to deactivate this memory (i.e. $\mathbf{M}$=null vector), giving priority to the most efficient subgraphs, whereas a value of 1 will lead to the opposite (i.e. never 2 subgraphs covering the same area). For the experiments, we set r^{max} to 0.5, its optimal value in our case. Finally, a limited number of the best candidates is retained for the current level, and the process repeats until there are no more candidates or the maximal level L^{max} is reached. In our experiments, we set L^{max} to 6 and limit the lattice width to at most $4N_{seed}$ as well as the number of children per node to 4 at most.

Subgraphs termination: The validation posterior probability of each candidate subgraphs A_k is estimated as the minimum probability over the classifiers that points at it:

$$p_{A_k} = \min_{\forall i:e_i \to A_k} p(O=1|sd_i \leq sd_i^{max})$$

If p_{A_k} exceeds the user-defined threshold p_{min}, then A_k becomes a *terminal* node and will not be grown thereafter (red squares in fig. 1.(c)). Thus, lattice subgraphs may become terminal at different sizes depending on the specificity of their keypoints regarding the object. This can be connected to the human cognitive way of recognizing occluded objects:

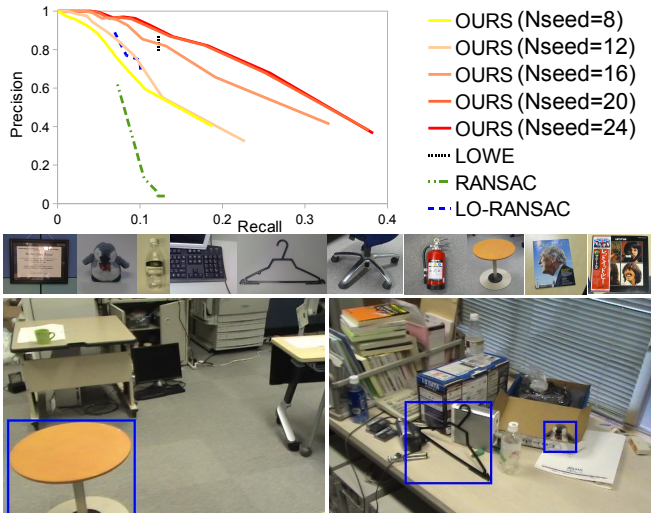

Figure 2. Top: global R-P curves for the 10 model objects (middle row). Bottom: examples of detections.

an object can be identified even if a very small but very characteristic part is visible and *vice versa*.

III. Experimental settings

A. Keypoint cluster size

The threshold σ in eq. (1) was set to $\sigma_1 = 0.5$ for the first level classifiers (i.e. corresponding to about 1000 visual words since $p(\|\mathbf{z}_r - \mathbf{z}_0\| \leq \sigma_1) \approx 1/1000$) and to $\sigma_{2+} = 0.7$ for the subsequent levels (about 100 words). This choice enables more filtering at start and hence a faster detection speed without noticeable loss of efficiency. Also, we empirically set $\alpha = 8$.

B. Test dataset

Often, recognition systems have been tested on HQ photos as in [1], [8]; however we wanted to truly simulate the realistic conditions of robotic vision so we manually have shot a dozen of indoor videos with a SONY handycam (resolution: 720x480). The variety of represented noises includes a poor/garish luminosity due to the indoor lighting, various camera noises (captor, movement blur, interlace) and also the objects themselves which are not always heavily textured. The videos were sampled at 10 fps, resulting in 2838 frames where the ground truth was manually labeled.

Ten objects at our disposal were used to test our system (see fig. 2). They were chosen so as to cover a large range of possible indoor objects: some are textured, some are not; some have complex 3D shapes; some are more prone to specular reflections; etc. We have used between 1 and 4 training images for each model though this number does not seem to have a great importance. Each object is visible in a few hundreds of frames. All objects were searched in the whole set of frames (multiple instances per frame are

allowed). In order to conform ourselves to a realistic case of use, we only took 19 photos as negative training set.

C. Comparison with existing systems

We compared our system against widely used specific object detection methods:

- The baseline RANSAC with a k-d tree matching scheme followed by an homography.
- The locally optimized RANSAC (LO-RANSAC) by Chum et al. [3] adapted by Philbin et al. [9].
- The object recognition system by Lowe [1].
- The proposed method with $N_{seed} \in [8, 24]$ and $p_{min} = 0.95$.

Results are presented in fig. 2 in terms of recall-precision curves (N_{seed} between parenthesis for our method). Recall and precision are defined as N_c/N_g and N_c/N_d respectively, where N_c is the number of correct detections, N_g the number of ground truth boxes and N_d the total number of detections (the higher is the curve, the better). Some detection examples are shown in fig. 2. Globally, our system outperforms the others from only $N_{seed} = 20$. Although our method starts the detection from such a small number of seed keypoints (to relate to the ~300 keypoints/model image for the other methods), superior values of recall are achieved (i.e. more instances are detected). Those are the main reasons:

- Lowe's method and LO-RANSAC requires respectively at least 3 and 4 correctly matched keypoints to assert a single detection, which is a rather difficult pre-requisite in noisy conditions.
- Our implicit proximity constraint helps to filter out more false positives, whereas the other methods are global and hence more sensitive to noise.
- The movement blur makes the keypoint descriptors become unspecific, decreasing the probability that the first-to-second distance ratio used in the other methods is accepted. On the contrary, we used an absolute distance between keypoint descriptors.

This indicates that an efficient detection scheme can be built in a two-steps manner: one first part for fast and rough hypothesis generation using a small fraction of the model information, and a second part for verification using the rest of the information. Moreover, there is no need for a complex transform, a simple 2D similarity suffices as long as the spatial constraints between distant features are relaxed.

Detection speed: Our method is also faster than the other methods: 55 ms/image on average for 10 objects, while Lowe's method and LO-RANSAC requires ~70 ms and RANSAC about 224 ms. This straightly follows from the cascaded lattice structure.

IV. Conclusion

We presented a novel approach which enables fast 2D object recognition robust to various realistic noises and occlusion. Thanks to the use of a cascade of classifiers as well as an absolute distance between keypoints, our method outperforms RANSAC-based or voting-based specific object detectors in noisy conditions. Moreover, we have demonstrated that reaching big values of recall does not necessarily require much of the model information ; interestingly only a tiny fraction of the model keypoints are sufficient to detect most instances in test images. Finally, the graph matching framework presented here is very generalist and could be extended to different image cues (e.g. contours or textures).

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[2] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[3] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," *Pattern Recognition*, pp. 236–243, 2003.

[4] V. Ferrari, T. Tuytelaars, and L. Gool, "Simultaneous object recognition and segmentation from single or multiple model views," *Int. J. Comput. Vision*, vol. 67, no. 2, pp. 159–188, 2006.

[5] B. T. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 493–504, 1998.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, pp. 511–518, 2001.

[7] B. Epshtein and S. Ullman, "Feature hierarchies for object classification," in *ICCV*, pp. 220–227, 2005.

[8] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *IJCV*, vol. 66, no. 3, pp. 231–259, 2006.

[9] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *CVPR*, pp. 1–8, 2007.