Towards Automatic Road Environment Mapping with Sparse Weakly Supervised Localization Models

Valentina Zadrija[a], Josip Krapac[a,*], Siniša Šegvić[a], Jakob Verbeek[b]

[a] *University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia*
[b] *INRIA Grenoble, Laboratoire Jean Kuntzmann, France*

Abstract

We propose a novel weakly supervised localization method based on Fisher-embedding of low-level features (CNN, SIFT), and model sparsity at the component level. Fisher-embedding provides an interesting alternative to raw low-level features, since it allows fast and accurate scoring of image subwindows with a model trained on entire images. Model sparsity reduces overfitting and enables fast evaluation. We also propose two new techniques for improving performance when our method is combined with nonlinear normalizations of the aggregated Fisher representation of the image. These techniques are i) intra-component metric normalization and ii) first-order approximation to the score of a normalized image representation. We evaluate our weakly supervised localization method on real traffic scenes acquired from driver's perspective. The method dramatically improves the localization AP over the dense non-normalized Fisher vector baseline (16 percentage points for zebra crossings, 21 percentage points for traffic signs) and leads to a huge gain in execution speed (91× for zebra crossings, 74× for traffic signs).

Keywords: Object localization, Weak supervision, Fisher Vectors, Sparse models, Convolutional features, Geographic information system (GIS), OpenStreetMap

*Corresponding author
Email address: josip.krapac@fer.hr (Josip Krapac)

1. Introduction

Detecting the presence of objects in images and recovering their locations are very important yet still open computer vision problems. These problems are often jointly addressed by applying a localization model at many image locations, and reporting objects where a positive response was obtained. Most successful representatives of this approach employ strong supervision at the training stage, which requires that each training image be annotated with accurate object locations. However, annotating object locations is expensive due to significant human labeling effort involved, even if a simple location model is used (e.g. bounding box). This is especially the case in realistic scenarios where thousands of annotations are required to achieve top performance. Annotation is particularly difficult when the objects of interest are small, since near to pixel-level annotation accuracy may be required for best results.

In order to alleviate the effort of full annotation, many recent approaches attempt to solve the localization problem in a weakly-supervised manner (cf. e.g. [1, 2, 3]). In this setting, training images are annotated only with class labels. The training procedure is supposed to train the localization model without knowing the object locations. At the test time, however, bounding boxes have to be predicted for each learned object class as in the strongly supervised case. This can be useful even if the recovered object classifier is not particularly fast, since the recovered object locations can be used to train a more efficient localization model in a strongly supervised fashion.

Weakly supervised training of object classifiers is a daunting task in most realistic scenarios. Even if we assume only one object in each positive image (which is not the case in our experiments), an exhaustive search would have to consider W^{T} hypotheses, where W is the number of image windows and T is the number of training images. This complexity may be decreased by sampling [4], clustering [5] or employing bottom-up location proposals based on trained segmentation [6, 3] or objectness cues [7, 8, 9, 10, 11]. However all these approaches risk to miss some objects at the selection stage, which may

invalidate all subsequent efforts.

A more conservative approach relies on classifiers able to detect the object presence in a larger image context. Such classifiers can be trained on positive images [12] or image regions [6] and then subsequently applied to recover or gradually improve the object localization. Some of the previous work along these lines has been based on BoW histograms [6, 12] and Fisher vectors coupled with bottom-up location proposals [3]. Recently, several researches have proposed approaches based on convolutional classifiers and end-to-end training [13, 14, 2].

In this paper we present a novel weakly-supervised object localization method based on Fisher vectors and model sparsity at the component level. Earlier accounts of this research appeared in [15, 16]. We extend that work in several ways:

- we present an end-to-end case study of road-environment mapping by training and evaluating our localization pipeline on images collected from crowd-sourced GPS labels (Sections 4, 5.2),
- we propose improved methods for generating location responses from top rated patches (Section 3.6),
- we improve support for our claims by presenting experiments on convolutional features (Section 5.2),
- we discuss comparative advantages of the sparse models against the dense ones for localization and classification tasks (Section 5.5).

In section 2 we argue that our method has important advantages with respect to other weakly-supervised localization approaches. These advantages arise since: i) Fisher vectors pool better than raw features due to ability to preserve and enhance unusual detail, ii) component-level sparsity enables fast evaluation and reduces overfitting, iii) we enable non-linear normalizations by intra-component normalization and approximated patch scoring and iv) the method does not require bottom-up location proposals. Section 3 presents details of the proposed method: patch-level Fisher vector embedding (Section 3.1), sparse localization

models (Section 3.3) and first-order approximation of the patch contribution (Section 3.4), efficient determination of the patch-level response (Section 3.5) and the recovery of bounding boxes (Section 3.6). Section 4 presents an effective semi-automatic procedure for collecting training images by exploiting crowd-sourced GPS labels. Section 5 presents an experimental validation of the proposed method on two datasets pertinent to the problem of automated road environment mapping. The datasets contain very small objects with much intra-class variation, in front of information-abundant background. We achieve good localization performance, comparable to strongly supervised approaches, while using a sparse model that accesses only a small fraction of the visual representation.

2. Related work

Most of existing weakly supervised localization approaches mitigate the computational complexity of weak supervision by relying on bottom-up location proposals. These approaches typically adopt the following iterative structure: i) train a discriminative model on the current guess of object locations in positive images, ii) use that model to select a better guess for the next iteration. This scheme optimizes a criterion that at least one (or exactly one [1, 7, 8, 12]) object is found in each positive image and that no objects are found in negative images. The optimization has been formulated as multiple-instance learning [6, 7, 4, 3, 1, 9] or end-to-end learning [10, 11]. All of these approaches may completely overlook small objects in training images, especially in traffic scenes with rich backgrounds. In our preliminary experiments, a popular objectness algorithm [1] consistently failed to produce accurate traffic sign locations in top 2000 proposals. Additionally, this kind of optimization is computationally very intensive, which complicates training on large datasets.

Another possible approach is to start optimization by discriminating entire (or almost entire) images and then gradually zoom onto object locations through iteration. One way to formulate this iteration is to present object locations as

latent variables in a deformable part model framework [1]. Another approach would be to construct an integral image of the patch scores and then find regions which maximize the classification score [12]. Both of these approaches do not require bottom-up location proposals, however they are prone to convergence issues, while not being able to handle training images with multiple objects. Classification of entire images has also been used to kick-off subsequent MIL optimization [3]. In each iteration, false positives of the current model are chosen as future negative samples, while future positive samples are set to top-scored bottom-up location proposals. However, as above, reliance on bottom-up location proposals may represent a liability in datasets with small objects.

Several recent approaches [13, 14, 2, 17] train weakly supervised localization models by exploiting deep convolutional architectures without bottom-up proposals and MIL-like iterations. These architectures are pre-trained on ImageNet and adapt the convolutional architecture for localization by converting fully connected layers to convolutional ones. The approach [14] completes the pipeline with global average pooling and fine-tunes all parameters with classification loss. The approach [13] avoids fine tuning by appending two convolutional adaptation layers and training them for classification of max-pooled scores on the target dataset. These approaches are attractive since they determine patch scores in a single forward pass while retaining the capability for end-to-end learning. However, they require joint training for all classes while our approach requires only the training of a distinct linear classifier for each class. Additionally, pooled convolutional features offer sub-optimal performance in popular architectures with a fully-connected back-end [18]. Hence [2] propose to perform localization by iteratively applying a fully-connected layer to down-sampled convolutional representations extracted within the active location hypotheses. The iteration is formulated as a beam-search under premise that better centered objects give rise to higher classification scores. Unfortunately, this is computationally much more expensive than the approaches based on pooling [13, 14], since it requires many forward passes through the network (additionally, beam search is not guaranteed to converge). Recently, Zhu et al [17] propose to generate object-

ness maps by iteratively accumulating confidence at the nodes that have high dissimilarity with their surroundings. The obtained objectness maps are then projected back, and further jointly optimized with network parameters.

Similarly to [13, 14], we also avoid bottom-up proposals by pooling local features. Our approach is perfectly suitable for end-to-end learning, although, in this work, we show experimental results exclusively on raw convolutional features provided by the public parameterization of the architecture VGG-19 [18]. However, different from [13, 14, 17], we avoid losing classification power with respect to the fully-connected case, by embedding low-level features (CNN, SIFT) into the Fisher space [19, 3, 20]. Aggregation of Fisher-embedded convolutional features has a very similar (if not larger) representative power as classical convolutional networks terminated with fully-connected layers [21]. Thus we obtain fair performance comparable to [2] and fast execution speed comparable to [13, 14, 17] in spite of the high dimensionality of Fisher representation. We succeed to keep computational complexity tractable by reinforcing a group-sparse classification model [22, 16, 23] and exploiting the first-order approximation of the patch contribution to the normalized Fisher vector of the image [16].

Recent work on zebra crossing localization [24, 25] is based on ad-hoc handcrafted features of appearance and shape. Line segments have been used in [24] to detect zebra crossings in corresponding Google satellite and street-view images acquired over a $1.6\,\mathrm{km}^2$ area in San Francisco[1]. A dataset of aerial images spanning across several countries has been proposed in [25], along with a detection approach based on HOG and LBPH features. Both approaches specialize for zebra crossings and require aerial imagery. On the other hand, our method is applicable to perspective views of various object classes (as shown in experiments on traffic signs and zebra crossings), while requiring only image-wide labels and no feature engineering whatsoever.

[1] In the process of acquiring the weakly supervised dataset for our experiments, we have contacted Google regarding the San Francisco dataset [26]. The response was negative so we acquired our dataset from other sources [27].

3. The proposed weakly supervised localization method

We present a method suitable for learning a localization model from crowd-sourced image-wide labels and geo-referenced video. The method is based on Fisher-embedding of low-level features [28] and group-sparse [29] classification models. We train a sparse model on Fisher vectors of entire images and use non-linear normalizations [30] to improve the classification performance. The obtained classifier is applied at all image locations to identify patches which contribute to the classification decision. A novel first-order approximation of the patch-contribution to the classification score ensures inter-operability with normalizations implied by improved Fisher vector [30]. We succeed to achieve near real-time performance due to fast evaluation of a group-sparse localization model. Localization responses (bounding boxes or convex hulls) are finally inferred by clustering positive patches.

3.1. Fisher vectors for weakly supervised localization

We regard images as orderless bags of patch descriptors (e.g. SIFT, convolutional features) to which we fit a generative Gaussian mixture model (GMM) $\theta = \{\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i\}_{i=1}^{K}$. Such model can be viewed as a visual vocabulary while its components can be referred to as visual words. The probability density function of a patch descriptor $\mathbf{x}$ can be stated as $p(\mathbf{x}|\theta)$. At this point, we may represent $\mathbf{x}$ as the gradient of the log-likelihood with respect to model parameters θ:

$$\mathrm{U}(\mathbf{x}|\theta) = \nabla_\theta \log p(\mathbf{x}|\theta)\,. \tag{1}$$

The score $\mathrm{U}(\mathbf{x}|\theta)$ succinctly describes the relation of the data point with respect to the parameters of the generative model [31]. We obtain the Fisher vector $\Phi(\mathbf{x}|\theta)$ by decorrelating the score [32]:

$$\Phi(\mathbf{x}|\theta) = \mathbf{F}(\theta)^{-0.5} \cdot \mathrm{U}(\mathbf{x}|\theta), \text{ where } \mathbf{F}(\theta) = \mathrm{E}_{\mathbf{x}}[\mathrm{U}(\mathbf{x}|\theta)\mathrm{U}^\top(\mathbf{x}|\theta)]\,. \tag{2}$$

The resulting representation captures first and second order statistics with respect to GMM components, and as such corresponds to a quadratic function

operating on low-level descriptors. Therefore, a linear classifier in the embedded space corresponds to a smooth piecewise quadratic decision boundary in the original space. However, besides being a quadratic kernel, the FV representation exhibits the following additional properties which make it especially suitable for weakly supervised localization:

1. additivity for $\mathbf{x}_i$ i.i.d.: $\Phi(\{\mathbf{x}_i\}|\theta) = \sum_i \Phi(\mathbf{x}_i|\theta)$
2. vanishing expectation: $\mathrm{E}_{\mathbf{x}}[\Phi(\mathbf{x}|\theta)] = 0$,
3. unit covariance: $\mathrm{E}_{\mathbf{x}}[\Phi(\mathbf{x}|\theta)\Phi^{\top}(\mathbf{x}|\theta)] = \mathbf{I}$.

We see that the representation of the whole image corresponds to the sum of patch representations. This allows to reverse the stages of pooling and scoring and to apply an image-wide linear classification model to locate patches responsible for the image label. Additionally, the FV representation attenuates the background information due to vanishing expectation. Equation (1) suggests that unusual datapoints "surprise" the generative model and therefore exert a strong influence to the aggregated representation. Thus, small distinctive objects stand a better chance to be noticeable in the image representation than in other aggregation approaches.

3.2. Non-linear normalizations of Fisher vectors

In this paper we use improved Fisher vectors [30] which involve power and metric normalizations of the image representation. The power normalization (signed square-rooting) is applied to each dimension X_d of the Fisher vector as $s(X_d) = \mathrm{sign}(X_d)|X_d|^{\rho}$, with $0 < \rho < 1$. The power normalization can be understood in terms of a positive semi-definite kernel function $K(\mathbf{X}, \mathbf{X}') = \langle \mathbf{s}(\mathbf{X}), \mathbf{s}(\mathbf{X}') \rangle$, where computing the power norm "un-sparsifies" the vector $\mathbf{X}$ and makes it more suitable for comparison with the dot product. The power normalization also accounts for the assumption that the low-level descriptors are i.i.d. [28]. The metric normalization projects the Fisher vector onto the unit hyper-sphere by dividing it with $\sqrt{n(\mathbf{X})}$ where $n(\mathbf{X}) = \sum_d s(X_d)^2$. This accounts for the fact that different images contain different amounts of background information [28].

In our work, we also experiment with the intra-component normalization [33, 23], where the ℓ_2 normalization is separately applied to the parts of the Fisher vector corresponding to different GMM components. In order to formally define the intra-normalized Fisher vector of the image, we use $\mathbf{X}^k$ to denote the part corresponding to the k-th visual word and write the corresponding ℓ_2 norm as $n(\mathbf{X}^k)$. Hence, given the GMM vocabulary with K components, the intra-component normalized FV corresponds to:

$$\frac{1}{\sqrt{K}} \cdot \left[\frac{\mathbf{s}(\mathbf{X}^1)}{n(\mathbf{X}^1)} \frac{\mathbf{s}(\mathbf{X}^2)}{n(\mathbf{X}^2)} \cdots \frac{\mathbf{s}(\mathbf{X}^k)}{n(\mathbf{X}^k)} \right] . \tag{3}$$

The normalization with $\sqrt{K}$ guaranties that $\|\mathbf{X}\| = 1$, when at least one patch is assigned to each GMM component.

3.3. Sparse models for classification and localization

Given the set of Fisher vectors $\mathbf{X_i}$ and the corresponding image-wide labels $y_i \in \{-1, 1\}$, we train a linear classifier $\mathbf{w}$ by minimizing the following regularized logistic loss function:

$$\ell(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{N} \log\left(1 + \exp(-y_i \cdot \mathbf{w}^\top \mathbf{X}_i)\right) + \lambda \cdot \mathcal{R}(\mathbf{w}) . \tag{4}$$

In the above equation, N denotes the number of images in the training subset, $\mathcal{R}(\mathbf{w})$ denotes the regularization function, while λ represents the trade-off between the loss and the regularization. The most widely adopted choices for $\mathcal{R}(\mathbf{w})$ include $\ell_2(\mathbf{w}) = \sum_j w_j^2$, and $\ell_1(\mathbf{w}) = \sum_j |w_j|$. The ℓ_1 regularization is of particular interest since it favours sparse models [34, 35] in which the majority of coefficients is zero. Such bias expresses a prior that the majority of image representation does not directly correspond to an instance of the considered object class. This kind of prior is very desirable in weakly supervised localization of small objects, since it discourages overfitting to soft contextual cues and allows efficient patch scoring.

However, ℓ_1 regularization ignores two important pieces of prior information specific to the classification of Fisher vectors. First, the contributions of particular object classes is typically concentrated in parts of the FV representation

corresponding to only a few words from the visual dictionary. This happens because feature extraction algorithms are designed or trained in a way that low-level features found at particular object classes group in the feature space. Second, the quickest way to process an image patch in our framework is to reject it immediately after the soft-assign stage, by noticing that the model coefficents are zeros at all components to which the patch contributes. Thus we see that instead of opting for unstructured *flat* sparsity induced by ℓ_1 regularization, it makes much more sense to prefer structured *component-level* sparsity where only coefficients corresponding to a few selected components are different than zero [22, 16, 23]. This kind of sparsity may conveniently be encouraged by supplying a regularization function defined as a sum of ℓ_2 regularization *within* components and ℓ_1 regularization *across* components:

$$\ell_{2,1}(\mathbf{w}) = \sum_k \ell_2(\mathbf{w}^k) \ . \tag{5}$$

3.4. From image classification to patch-level scores

Let $f(\mathbf{X})$ denote the classification score of the image representation $\mathbf{X}$. Then, the contribution of patch $\mathbf{x}_i$ to the overall image score can be expressed as:

$$pc_{\text{direct}}(\mathbf{x}_i) = f(\mathbf{X}) - f(\mathbf{X} - \mathbf{x}) \ . \tag{6}$$

In the case of a linear classification model and un-normalized FV representation, we have $f_{\text{linear}}(\mathbf{X}) = \mathbf{w}^\top \mathbf{X}$. Therefore, we can reverse the scoring and sum-pooling operations and express the patch contribution as a simple dot product:

$$pc_{\text{linear}}(\mathbf{x}_i) = f_{\text{linear}}(\mathbf{X}) - f_{\text{linear}}(\mathbf{X} - \mathbf{x}) = \mathbf{w}^\top \mathbf{x}_i \tag{7}$$

In the case of improved FV [30], the linear image score can be expressed as:

$$f(\mathbf{X}) = \mathbf{w}^\top \mathbf{s}(\mathbf{X}) / \sqrt{n(\mathbf{X})} \tag{8}$$

We see that non-linear normalizations described in Section 3.1 invalidate additivity of the Fisher representation and make the linear decomposition impossible: $\mathbf{w}^\top \mathbf{s}(\mathbf{X})/\sqrt{n(\mathbf{X})} \neq \sum_i \mathbf{w}^\top \mathbf{s}(\mathbf{x_i})/\sqrt{n(\mathbf{x_i})}$. Instead the patch contribution

may be computed directly, as in (6). This requires the following operations to be applied at each image patch: (i) subtracting the patch representation $\mathbf{x}$ from the image representation $\mathbf{X}$, (ii) applying the power and ℓ_2 normalizations onto the result, (iii) scoring with the model $\mathbf{w}$ and (iv) subtracting the obtained score from the image score. Unfortunately, this procedure is computationally very complex which makes its application to all image patches impractical. We therefore propose a first-order approximation which corresponds to taking the dot-product between the un-normalized patch representation and the gradient of the normalized image score.

We now derive the gradient of the classification score of the normalized image representation w.r.t. non-normalized patch representation $\mathbf{x}$. The partial derivative of the score w.r.t. an element of the non-normalized patch x_d is given by $\partial f(\mathbf{X})/\partial x_d = \partial f(\mathbf{X})/\partial \mathbf{X} \cdot \partial \mathbf{X}/\partial x_d$. The derivative of the non-normalized image representation w.r.t. the d-th element of the patch FV corresponds to the vector with all zero elements except the d-th which is equal to one. Hence, the gradient w.r.t. the patch element x_d is equal to the gradient w.r.t. image element X_d:

$$\frac{\partial f(\mathbf{X})}{\partial x_d} = \frac{\partial f(\mathbf{X})}{\partial X_d} = \frac{\rho |X_d|^{\rho-1}}{\sqrt{n(\mathbf{X})}} \left(w_d - \frac{s(X_d) f(\mathbf{X})}{\sqrt{n(\mathbf{X})}} \right) . \tag{9}$$

For more details regarding the gradient derivation, the reader can refer to Appendix A. The above expression is undefined in cases where $X_d = 0$ (a rare case since the full image FV are dense). In such cases, we set the derivative to zero to ignore the impact of such dimensions.

In the case of intra-component normalization, the classification score is a sum of per-component classification scores: $f(\mathbf{X}) = \sum_k f_k(\mathbf{X}^k)$. Since the $f_k(\mathbf{X}^k)$ have precisely the same form as $f(\mathbf{X})$ above, we can compute the gradients in the same manner, per each component.

3.5. Efficient patch scoring with a sparse model

The complexity of patch scoring can be subdivided into the following three stages with similar computational complexity: i) computing the soft-assign

$p(k|\boldsymbol{x})$, ii) computing the FV, and iii) determining the patch contribution by (6), (7), or (9). In this paper, we consider efficient implementation of the latter two stages for CPU architectures (efficient soft-assign is addressed in [36]). In the naive implementation, both of these two stages are $O(NKD)$ where N is number of patches, K is the number of components, while D is raw feature dimensionality. We reduce that complexity by exploiting two kinds of sparsity. The soft-assign sparsity refers to the fact that the GMM posterior is very sparse: a majority of patches are dominantly assigned to only one GMM component. The model sparsity indicates that our models typically reference only a tiny part of the representation as described in Section 3.3.

First, we find GMM components with significant soft-assign $p(k|\boldsymbol{x}) > 1/K$ and denote their number with K_s. Due to soft-assign sparsity we typically have $K_s \ll K$. Second, we identify K_w non-zero blocks of the model $\boldsymbol{w}$, where blocks correspond to the GMM components. In the case of a group-sparse regularizer, K_w is directly influenced by the amount of regularization and we typically have $K_w \ll K$. At this point we can reject the patches that are not assigned to any of the K_w selected components. This reduces the number of patches from N to N' and, depending on the abundance of the object class, may result in very large speedups ($N' \ll N$), similarly to the effects of the first few stages of a cascaded classifier [37]. For the remaining N' patches we need to compute only the parts of the Fisher vector which correspond to the intersection of K_s assigned components and K_w components incident to the model. Therefore, we need to compute Fisher vectors and patch contributions only for at most $K' = \min(K_s, K_w)$ components. Total complexity of these two stages for all image patches is $O(N'K'D)$, corresponding to a tiny fraction of the original complexity.

This efficient procedure is equally applicable for linear scoring with the model $\mathbf{w}$ and the gradient (9). The proposed procedure can also be applied for direct scoring (6) of intra-normalized patch representations. In this case the sparsity of patch representation and the locality of intra-normalization ensure that only the patches generated by the K_w selected components may result in non-zero

scores.

Please note that the efficient procedure is not exact in the case of global normalization (8) and direct scoring (6), since the contribution outside the selected K_w components alters the representation norm. Hence, this combination requires exhaustive evaluation of the patch score which results in extraordinary long execution times as we shall see in the experiments.

3.6. From patches to objects

We extract low-level patches on several scales, embed them into high-dimensional FV space and compute their contributions to the overall image score as described in previous sections. We explore two different approaches for generating localization responses from top-rated patches: (i) independent processing of each particular scale, and (ii) combining per-scale responses in a unified heat map.

3.6.1. Per scale approach

We build a spatial graph by connecting top T rated patches on each particular scale which overlap more than $P\%$. Localization responses are generated as a union of patches assigned to the particular connected component. Connected components with less than N patches are removed from consideration. The main motivation behind this approach is to prevent co-occurring background patches of different sizes to be recognized as objects. This approach has been used to generate localization responses in experiments presented in Section 5.3, which contains referent values for parameters T, P and N.

3.6.2. Multi-scale heat map approach

We compute per-scale patch contributions at the resolution of the low-level features, and up-sample them to the resolution of the full original image by nearest neighbour interpolation. Let $f(x_i|c,s)$ denote the patch score at the pixel x_i for the class c obtained by up-sampling the scores of patches at the scale s. The cumulative score $f(x_i|c)$ at each pixel x_i is then computed as:

$$f(x_i|c) = \sum_s f(x_i|c,s) \ . \tag{10}$$

In order to generate localization responses from the obtained unified heat map, we consider pixels with a score greater than some threshold T. In practice, we set T to be an average of all pixels with a positive score, i.e. $T = 1/P \cdot \sum_i f(x_i|c)$, such that $f(x_i|c) > 0$, where P denotes the number of such pixels. We group the selected pixels into connected components and generate localization polygons as convex hulls of pixels in a particular component. This approach has been used in conjunction with convolutional low-level features in Section 5.2.

4. Weak labels for road mapping

This section describes how to generate image-wide labels by matching crowd-sourced location database such as OpenStreetMap [38] to geo-referenced video.

4.1. OpenStreetMap data representation

The OpenStreetMap (OSM) database comprises three basic entities: nodes, ways and relations [38]. An OSM node represents a point object defined by its GPS coordinates, i.e. longitude and latitude. A way is defined as an ordered list of nodes and its purpose is to describe linear or area-like features (e.g. roads, rivers, buildings). A relation is defined as an ordered list of nodes, ways or relations and its goal is to define logical or geographical relationships (e.g. bus routes, turn restrictions or multi-polygons). Each of these entities can be assigned a collection of tags which provide semantic meaning. Tags are defined as key-value pairs, where keys are used to organize entities into different categories. In general, road-environment objects are designated with a key "highway". Some examples of tags include "highway"="traffic_signals" (used to denote the traffic lights), "highway"="give_way" (used to denote the "yield" traffic sign) or "highway"="crossing" (used to denote the zebra crossing). OSM entities [39, 40] can be accessed through the Overpass website [41] which provides an interface to query the OSM database for specific features (e.g. zebra crossings, rest area facilities or traffic signs).

4.2. Geo-referenced video

Geo-referenced video consists of a file with video data in one of the standard video formats, and a corresponding text file containing spatio-temporal information such as time, GPS location and camera direction. There are numerous ways on how to acquire a geo-referenced video [42] including GPS enabled cameras and camcorders, or smartphone applications like Mapillary [43] or OpenStreetView [44]. In this paper, we used geo-referenced video sequences delivered by the E-roads projects for Croatian cities of Karlovac and Sisak [27]. The obtained material consists of 1536 videos captured by different cameras, ranging from high resolution 1080p to 720p devices. Each video sequence is assigned a corresponding text file in a JSON format, where each JSON object contains an array of GPS coordinates and a time offset from the beginning of the video, e.g. {"coordinates": [15.603174, 45.478279], "time": 53.5}. The frequency of the GPS sampling depends on the underlying video and ranges from 5 Hz to 0.5 Hz. The frame rate of video sequences is 25 fps.

4.3. Matching OSM objects to geo-referenced videos

We use the OSM nodes tagged as "highway"="crossing" and match them to geo-referenced video sequences to obtain positive training images with weak labels. In order to improve performance, we preprocess the list of spatio-temporal coordinates (GPS location, time offset) assigned to the video, by partitioning it into segments $\mathbf{g}$ of at least 5 m in length. These segments are used to quickly locate the pertinent part of the video, as illustrated in Figure 1.

For each OSM node $\mathbf{n}$, the matching algorithm considers all segments $\mathbf{g}_i$ which are i) in a 10 m radius around $\mathbf{n}$, and ii) closer to $\mathbf{n}$ than the neighbouring nodes $\mathbf{g}_{i-1}$ and $\mathbf{g}_{i+1}$. On each such segment we find the GPS reference $\mathbf{p}_n$ which is closest to $\mathbf{n}$. This can be seen as an approximate projection of $\mathbf{n}$ onto $\mathbf{g}_i$. Subsequently we take T snapshots from the video every Δd meters backward, in order to retrieve images for which the desired object is visible and close to the viewpoint. The whole procedure is summarized in Algorithm 1.

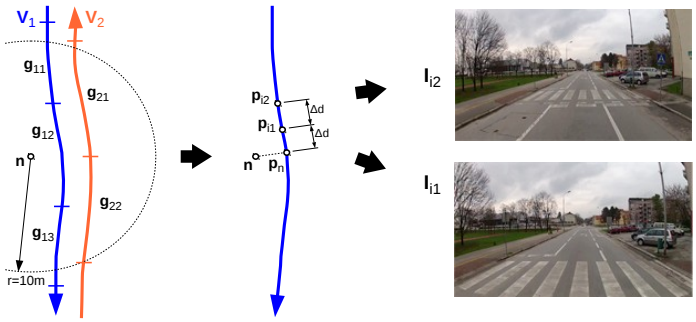


Figure 1: Matching the OSM node with osm_id = 2043645281, located at $\mathbf{n}$ = 45.487347 N, 15.556345 E to geo-referenced videos $\mathbf{V}_1$ and $\mathbf{V}_2$. The first step of matching selects segments $\mathbf{g_{12}}$ and $\mathbf{g_{21}}$. We detail the second step for the segment $\mathbf{g_{12}}$: i) a GPS reference closest to $\mathbf{n}$ is located and designated as $\mathbf{p}_n$, ii) each of T=2 images is extracted by following the path backwards for Δd=3m.

Algorithm 1 Matching OSM objects to geo-referenced videos

Parameters:

T: number of images per each occurrence of the desired OSM node,

Δd: desired spatial distance between images;

Input:

OSM node $\mathbf{n}$=(lat, lon),

Videos $\mathbf{V} = (\mathbf{F}, \mathbf{G})$ where

i) $\mathbf{F} = \{\mathbf{I}_i, \mathbf{p}_i\}$: video frames and the corresponding GPS references

ii) $\mathbf{G} = \{\mathbf{g}_j\}$: piecewise linear approximation of $\{\mathbf{p}_i\}$

(each $\mathbf{g}_j$ represents a range of GPS references $\{\mathbf{p}_k\}, k \in \mathbf{g}_j$)

For all videos $\mathbf{V} = (\mathbf{F}, \mathbf{G})$

  For all $\mathbf{g}_j \in \mathbf{G}$: $\|\mathbf{g}_j - \mathbf{n}\| < \min(\|\mathbf{g}_{j-1} - \mathbf{n}\|, \|\mathbf{g}_{j+1} - \mathbf{n}\|)$

    find $n = \arg\min_{k \in \mathbf{g}_j} \|\mathbf{p}_k - \mathbf{n}\|$,

    if $\|\mathbf{p}_n - \mathbf{n}\| > 10\,\mathrm{m}$

      continue

    For t $\in 1, 2, \ldots T$

      find $i_t = \arg\min_i (t \cdot \Delta d - \sum_{k=i+1}^{n} \|\mathbf{p}_k - \mathbf{p}_{k-1}\|)^2$

      if $i_t = 0$

        break

      extract image $\mathbf{I}_{i_t}$

Return: extracted images with the label "object"

In order to account for discrepancies between the OSM geometry and GPS data, we set T=3 and Δd=3 m. Due to various kinds of uncertainty involved (errors in GPS references, imprecise OSM tags, etc.) the proposed algorithm may also extract some images without an object of interest. The prevalence of such images in the case of zebra crossings was around 15%. In the actual experiments, we filter such images out by manual inspection. An interesting direction for future work would be to fully automate the process by omitting the manual filtering and learning the desired visual concept from noisy labels.

5. Experimental evaluation

We evaluate the proposed workflow on two kinds of traffic infrastructure: zebra crossings and triangular warning signs. We compare efficiency of two kinds of local features and quantify the localization accuracy with average precision (AP) [45] for various IoU thresholds. Experiments show that our method succeeds to localize objects of varying size, both very small and large, in rich traffic scenes. Besides the recognition accuracy, we also measure the execution time spent in scoring of feature embeddings as the critical part of the localization pipeline.

5.1. Implementation details

We perform experiments on two kinds of local features: VGG conv5_4 [18] and dense SIFT [46]. We extract approximately $80{\cdot}10^3$ 128-D SIFT descriptors per image at 4 scales for traffic signs and 5 scales for zebra crossings. The smallest patch size was set to 16 (traffic signs) and 32 (zebra crossings) pixels, while the stride was always set to 1/8 patch width. Subsequently, the descriptors are ℓ_2 normalized and projected onto an 80-D principal component subspace.

VGG conv5_4 features correspond to 512-dimensional activations from the conv5_4 layer of the deep convolutional model VGG-19 [18]. We use the provided parameters trained on ImageNet and abstain from fine-tuning [14, 10, 17] in order to avoid catastrophic forgetting [47] and preserve applicability to a

diverse set of object classes. In the case of zebra crossings, we extract VGG conv5_4 features at 3 scales obtained by rescaling the image by factors 2^s where $s \in \{0, -0.5, -1\}$ [18]. In the case of traffic signs we use single scale $s = 0$.

We embed the extracted local features into Fisher vectors with respect to GMMs with diagonal covariances, trained with the EM implementation from Yael [48]. We used $K_{\mathrm{SIFT}} = 1024$ and $K_{\mathrm{VGG}} = 128$ components. Doubling these figures did not produce noticeable effects on a held-out dataset. The dimensionalities of the Fisher vectors are 164 864 (SIFT) and 131 200 (VGG). We aggregate local Fisher vectors into image descriptors and apply different non-linear normalizations [30]. Both local and image-wide Fisher vectors are scored with linear models trained by SPAMS [49], as described in Section 3.3. The regularization hyper-parameter λ was determined by 10-fold cross-validation.

5.2. Zebra crossings

Experiments on zebra crossings provide a proof of concept for the road mapping scenario presented in Section 4. We detail the semi-automatic dataset acquisition procedure, present classification and localization experiments, and discuss the experimental results. We first present experiments on VGG conv5_4 features extracted from the dataset with noiseless labels. Experiments on SIFT descriptors and noisy labels are presented towards the end of this section.

5.2.1. Dataset

We collected the images from geo-referenced video obtained in the scope of a public road mapping project [27]. We recovered the locations of zebra crossings by querying the OpenStreetMap (OSM) database with a "highway=crossing" tag using the Overpass API interface [41]. We matched the object locations to geo-references of the video frames as described in Section 4. For each OSM entry, we extracted on average 6 images taken from different videos and different distances to the object, and rescaled them to the lowest common resolution 1280 $\times$ 720. In this way we recovered 1259 noisy positive images and 1122 negative images by random selection far from OSM entries. The extracted images depict

Figure 2: Top row: images extracted for the query osm_id 981409265 located at 45.483031 N, 15.546749 E. From left to right: an image in which the zebra crossing was freshly painted, an image acquired from a bicycle, a close-up image in which the crossing was worn-out, and another image from that video taken 20 metres from the crossing. Bottom row: negative images contain many objects with patterns which are similar to zebra crossings.

objects under different perspectives and various weather conditions as shown in Figure 2.

We produced a clean version of the dataset by manual filtering of the 1259 noisy positives into 1067 positives and 192 negatives, and by manual verification that the 1122 negatives do not contain an unmapped object. The cleaning procedure took around 20 minutes. We split both versions of the dataset into roughly equal train and test subsets by taking care that all physical objects are assigned to the same subset.

In order to evaluate the localization performance, we annotated the ground-truth object locations in the test set with polygonal approximations. Some of the test images contain several objects at different distances from the camera. For the sake of completeness, we annotated all of them, even the smallest ones. Details concerning the statistics of annotated objects with respect to the image area are shown in Table 1. Hence, from 484 positive test images we obtained 674 object annotations, where 337 images contain a single object, 120 of them contain 2 objects, 14 contain 3 objects and another 13 contain more than 3 objects. The complete dataset shall be made public upon publication of this manuscript.

Table 1: Annotated object size statistics for the zebra crossing dataset. The first column contains the object size intervals expressed in percentages of an image area. The second column indicates the overall share of objects in the corresponding size interval.

Relative object size	Frequency
<1%	30%
2% − 7%	48%
>7%	22%

5.2.2. Classification with VGG conv5_4 and noiseless labels

These experiments evaluate effects of cross-validated regularization and non-linear normalization to the classification AP. The first section of Table 2 (rows #1-#3) assumes that images are represented with raw Fisher vectors. We notice that the $\ell_{2,1}$ regularization succeeds to induce a five-fold increase of component-level sparsity with respect to ℓ_1 regularization, which shall be especially useful in localization experiments.

Table 2: Classification of zebra crossings with different FV normalizations (p: power, ℓ_2 global, ℓ_2 intra) and regularizations (ℓ_1, ℓ_2, $\ell_{2,1}$: ℓ_2 inside component, ℓ_1 between components). Average overall density (AOD) and average component density (ACD) denote percentages of non-zero model coefficients and non-zero model coefficient groups, respectively.

Nr.	FV norm.	Penalty	AOD	ACD	AP train	AP test
1	-	ℓ_2	77.4	100.0	97	95
2	-	ℓ_1	2.1	78.9	94	95
3	-	$\ell_{2,1}$	8.5	14.8	95	95
4	p, ℓ_2 global	ℓ_2	77.4	100.0	100	97
5	p, ℓ_2 global	ℓ_1	0.2	30.5	100	98
6	p, ℓ_2 global	$\ell_{2,1}$	3.8	**7.0**	100	98
7	p, ℓ_2 intra	ℓ_2	77.4	100.0	100	97
8	p, ℓ_2 intra	ℓ_1	0.1	21.1	97	98
9	p, ℓ_2 intra	$\ell_{2,1}$	2.6	**4.7**	100	98

The next section (rows #4-#6) considers models trained on Fisher vectors with power and global ℓ_2 normalization. We notice that non-linear normalizations increase the classification performance (all models) and the component-level sparsity (ℓ_1 and $\ell_{2,1}$). We believe that non-linear normalizations promote the model sparsity by decreasing the range of Fisher vector elements and therefore encouraging the model to bring decisions by testing presence instead of testing the aggregated value. Finally, we consider intra-component ℓ_2 normalization (rows #7-#9). This technique works especially well in conjunction with the group-sparse $\ell_{2,1}$ regularization, since the combined model retained only 5% of GMM components while reaching the same level of classification performance as in the case of global ℓ_2 normalization (rows #4-#6).

To summarize, the sparse models achieve comparable or better results with respect to their dense counterparts, despite the fact that they utilize only a tiny fraction of the visual dictionary. Non-linear normalizations boost the performance on the test dataset up to 3 percentage points (pp).

5.2.3. Localization with VGG conv5_4 and noiseless labels

In these experiments, we train the model on entire images, apply it to all image patches at multiple scales, and recover objects as described in Section 3. In order to compensate for large variability in scale and semi-automatic dataset generation, we first use a permissive IoU threshold [45] of 0.10 while other IoU thresholds shall be considered in one of the following paragraphs. The obtained results are summarized in Table 3 and Figure 3.

The first section of Table 3 (rows #1-#3) shows that sparse models outperform the dense baseline (ℓ_2) for up to 16 pp in AP and 15 pp in p_{miss}. We also note that the $\ell_{2,1}$ model (row #3 in Table 3) outperformed the ℓ_1 model (row #2), despite employing much less GMM components. The first row in Figure 3 reflects that result: the $\ell_{2,1}$ model (right) selects less background pixels than the ℓ_1 and ℓ_2 models. This advantage was consistent in all our experiments and supports the hypothesis outlined in Section 3.3 that the $\ell_{2,1}$ model may perform better due to agreement with the group structure of the Fisher vector.

Table 3: Localization of zebra crossings with different configurations (M: direct patch contribution, G: gradient) and FV normalizations. p_{miss} denotes the miss frequency at the rightmost data point of the PR curve. t_{op} denotes the average time required to compute the scores for the selected patches. For gradient configurations (rows#6-#7), we show the speed-up w.r.t. corresponding row with direct patch contribution.

Nr.	Conf.	FV norm.	Penalty	ACD	AP test	p_{miss}	t_{op}/s	speed-up
1	M	-	ℓ_2	100.0	76	0.46	26.6	
2	M	-	ℓ_1	100.0	91	0.33	19.1	
3	M	-	$\ell_{2,1}$	14.8	92	0.31	2.8	
4	M	p, ℓ_2 global	$\ell_{2,1}$	7.0	92	0.27	29.8	
5	M	p, ℓ_2 intra	$\ell_{2,1}$	4.7	**93**	**0.25**	0.8	
6	G	p, ℓ_2 global	$\ell_{2,1}$	7.0	90	0.28	1.0	28.8×
7	G	p, ℓ_2 intra	$\ell_{2,1}$	4.7	**92**	**0.25**	**0.3**	2.7×

Hence, we include only the results of $\ell_{2,1}$ models in the rest of the table. The middle section of Table 3 (rows #4-#5) employs normalized Fisher vectors and computes the patch contribution directly as $f(\mathbf{X}) - f(\mathbf{X} - \mathbf{x})$. This further decreases the p_{miss} frequency and selects fewer pixels in the background as confirmed by rows 2 and 3 in Figure 3. We notice that the best localization result (93% AP and 0.25 p_{miss}) is obtained by intra-component ℓ_2 normalization which constrains the influence of unusual patches to the components with significant soft-assign probability. We also note that intra-component ℓ_2 normalization has a negative effect when ℓ_1 regularization is used. We believe this happens because ℓ_1 regularization disrupts the component structure of the Fisher vector and thus makes intra-component normalization counter-effective for localization purposes. The final section of Table 3 (rows #6-#7) evaluates the effects of the gradient approximation (9) to the configurations from the previous section (rows #4-#5). The results show that the performance hit is minimal: up to 2 pp drop in AP, and up to 1 pp increase in p_{miss}. We arrive to the same conclusion by performing a qualitative comparison between rows 2-3, and rows 4-5 in Figure 3.

Failure cases. Figure 4 shows representative examples of our best configuration (row #5 in Table 3). The correct localization polygons are shown in yellow, incorrect ones are shown in red while ground truth polygons for the missing localizations are shown in magenta. Our method is able to localize objects taken from different viewpoints (frontal, lateral) as well as worn-out objects (cf. row 1, far right). Most false negatives (80%) correspond to extremely small

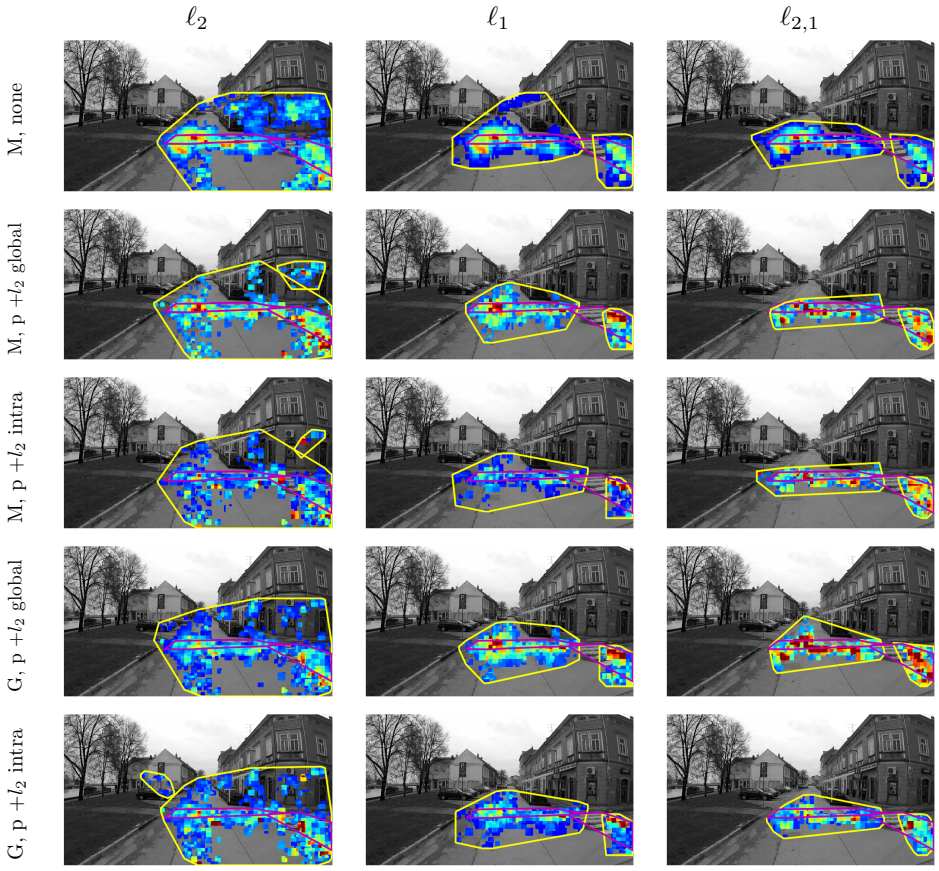


Figure 3: A comparison of multi-scale heat maps for different non-linear normalizations (rows) and different regularizations (columns). The input image contains 2 zebra crossings: one in the lateral position very close to the viewpoint, and another one in the frontal position 10 meters from the camera. Magenta polygons denote the ground truth instances; they are used for evaluation purpose only. Yellow polygons denote the resulting localization responses.

objects far away from the viewpoint. The remaining false negatives arise in images with adjacent objects (ground-truth polygons are touching each other) where our method reports one instead of two or more objects (cf. rows 2 and 3, far right). The average size of the missed objects corresponding to the first

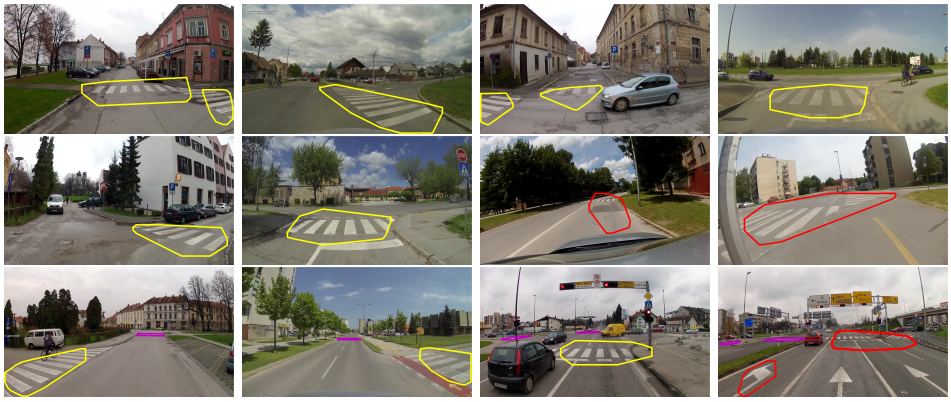


Figure 4: Localization results on test images: correct localization polygons (yellow), false postive responses (red), and ground-truth polygons for false negatives (magenta).

failure type is 4573 px, which is only 0.5% of the total image. As we already pointed out in Section 3, such failures are not critical for accurate road mapping where we care most about the objects which are close to the viewpoint.

IoU threshold. Figure 5 explores the influence of the IoU threshold to the localization performance for our best configuration (row #5 in Table 3). The left subfigure shows the counts of objects and the corresponding average IoU depending on the ground-truth size interval. The graph reveals that most of objects which are missed for IoU < 0.1 are smaller than 10^4 pixels which corresponds to 1% of the image size. The right subfigure shows the effect of the IoU threshold to average precision and the miss frequency for objects which are greater than 10^4 pixels. The graph shows that the increase of IoU threshold results in gradual decrease of AP and increase in $\mathrm{p_{miss}}$. Setting IoU to 0.3 results in 80% AP and 0.3 p_{miss}, which is still a fairly good localization accuracy in a weakly supervised setting. Further increase of the IoU threshold severely degrades the AP due to false negatives and failures to distinguish nearby objects.

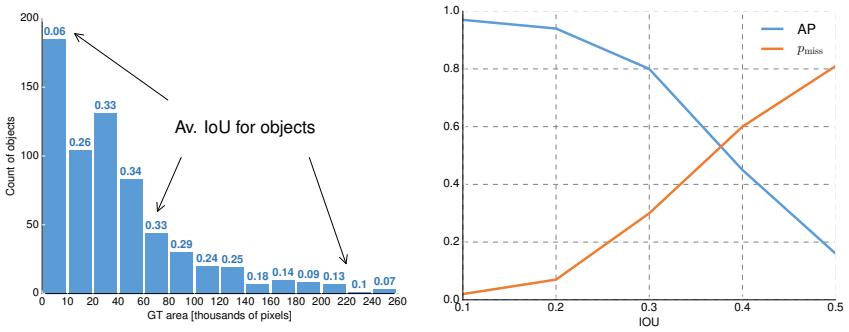


Figure 5: Influence of the IoU threshold to the localization AP performance on the zebra crossings dataset for the configuration #5 from Table 3. Left: distribution of objects over a range of ground-truth area intervals. For each ground-truth area range, we report the average IoU. Right: the effect of the IoU threshold on AP and p_{miss} for objects greater than 1% of the image area.

Execution time. Finally, we provide a brief analysis of the execution profile. We extract on average 6344 CNN features per image, which takes around $t_{\text{lf}} = 1$ s on an NVidia GTX 980 GPU. All subsequent processing is performed on a single CPU core of a 2.00 GHz Intel Xeon E5-2620. Computing the soft assign probability for the extracted features takes $t_{\text{sa}} = 0.11$ s. The times t_{lf} and t_{sa} are constant for all configurations. Our Python implementation takes on average 1.9 s per image ($t_{\text{lf}} + t_{\text{sa}} + t_{\text{op}}$) for our best configuration in terms of AP (row #5 in Table 3). The gradient approximation further reduces that time to 1.4 s with a small penalty on the localization AP (row #7 in Table 3).

We conclude that $\ell_{2,1}$ models are faster than their ℓ_1 counterparts due to a better chance of having an empty intersection with the Fisher vector of image patches (improvement over ℓ_2 models is even larger). Intra-component normalization further decreases the processing time since it allows to pre-compute most components of $f(\mathbf{X} - \mathbf{x})$ when patch contributions are computed directly. Finally, the gradient approximation allows to score a patch with a simple dot product. Gradient approximation works very well with intra-component normalization since the latter preserves the sparsity of $\partial f(\mathbf{X})/\partial x_d$ and therefore leads to empty intersections for most patches. When compared with the base-

line (row #1 in Table 3), our fastest configuration (row #7 in Table 3) leads to 89-fold improvement in t_{op} or about 20-fold improvement in overall execution time.

5.2.4. Comparison with SIFT and noisy labels

Table 4 further explores our best configuration in terms of localization AP by presenting results with different features and noisy labels. For convenience of comparison, the row #1 repeats the results from the row #5 in Table 3. The row #2 shows effects of replacing VGG conv5_4 features with dense SIFT features. The results show that our weakly supervised pipeline is able to successfully accommodate different kinds of local features on this dataset. The row #3 shows effects of training the model on raw image dataset obtained by sampling the geo-referenced video at GPS locations provided by crowdsourced OSM tags. The raw dataset contained significant noise in positive labels since around each 9th positive image in the training set was in fact a negative (we used a clean test dataset in all experiments). Nevertheless the obtained results show that this leads to only a slight decrease (1 pp) in localization AP. We hypothesize that this robustness is due to regularization imposed by the $\ell_{2,1}$ penalty and the low capacity of the chosen classification model. This suggests that fully automated road environment mapping is feasible.

Table 4: Comparison of our best configuration in terms of localization AP (row #5 in Table 3) with the results obtained with different features and noisy labels. All experiments use $\ell_{2,1}$ regularization, and direct computation of the patch contribution. Experiments on VGG conv5_4 use intra-component normalization, while experiments on traffic signs use raw Fisher vectors.

Nr.	Features	Training set	ACD	AP test	p_{miss}	t_{op}/s
1	VGG conv5_4	noiseless	4.7	93	**0.25**	0.8
2	SIFT	noiseless	4.1	**95**	0.26	0.9
3	VGG conv5_4	noisy	3.9	92	0.25	0.7

5.3. Traffic signs

Experiments on traffic signs explore the capability of our approach to deal with extremely small objects. We focus on 33 different kinds of European triangular warning signs [50] which we treat as a single object class in weakly supervised localization experiments. We first present experiments on SIFT features, while the comparison with VGG conv5_4 is presented in the subsequent subsection.

5.3.1. Dataset

We use a public traffic sign dataset [51] and adapt it for weakly supervised localization[2]. The adapted dataset contains 1705 train and 1591 test images. Most images contain only one traffic sign while around 16% images contain two traffic signs. The train and test subsets are disjoint: all physical object are present in only one subset. The main challenge in this dataset is object size, since many instances are less than 30×30 pixels large (0.2% of image size). All images are correctly labeled and they have the common resolution 720×576.

5.3.2. Classification with SIFT features

Classification experiments evaluate effects of cross-validated regularization and non-linear normalization. The results are summarized in Table 5. Group sparse models ($\ell_{2,1}$) outperform alternatives (ℓ_1 and ℓ_2) in all configurations, both with respect to generalization performance (AP test) and average component density (ACD). This can be observed in rows #1-#3 which address classification with unnormalized Fisher vectors. Similar effects occur with normalized Fisher vectors as well, however in that case we only show the group sparse models in order to save space (rows #4-#5). We achieve the best classification performance with global normalization (row #4, 81% AP), however, intra-component normalization is only marginally worse (row #5, 80% AP).

[2]The adapted traffic sign dataset can be downloaded from
`http://multiclod.zemris.fer.hr/ts2010a.shtml`

Table 5: Image classification performance for traffic signs with different FV normalizations (p: power, ℓ_2 global, ℓ_2 intra) and regularizations (ℓ_1, ℓ_2, $\ell_{2,1}$: ℓ_2 inside component, ℓ_1 between components). AOD (average overall density) denotes the percentage of non zero model coefficients (out of 164 865 total). ACD (average component density) denotes the percentage of non-zero model components (out of 1024 total).

Nr.	FV normalization	Penalty	AOD	ACD	AP train	AP test
1	-	ℓ_2	92.8	100.0	100	66
2	-	ℓ_1	0.1	6.1	87	71
3	-	$\ell_{2,1}$	1.0	1.1	83	75
4	p, ℓ_2 global	$\ell_{2,1}$	1.1	1.1	87	**81**
5	p, ℓ_2 intra	$\ell_{2,1}$	0.8	0.8	85	80

5.3.3. Localization with SIFT features

We follow the same experimental setup as in the case of zebra crossings. Table 6 shows the obtained results. Due to the fact that the traffic sign dataset [51] contains ground truth bounding boxes for both train and test splits, we were able to train a strongly supervised baseline [52] (row #1). We employ the standard OpenCV implementation and invoke it through Python interface `cv2.HOGDescriptor.detectMultiScale`. We configure the baseline approach for high accuracy, by adjusting parameters of the sliding window according to the range of scales at which the traffic signs appear in our dataset. Thus we extract the HOG features at 64 different scales ranging from 24×24 to 160×160 (the inter-scale factor was set to 1.03), while the stride was set to two pixels. In comparison to our best weakly supervised result (row #7: 92% AP, 0.15 p_{miss}), the strongly supervised baseline achieves worse AP (88%) and better miss frequency (only 5% of objects were not found). However, please note that we extract local features on only 4 scales, which is 8 times less than the baseline (64 scales).

Experiments with weakly supervised models support our hypotheses (i) that sparse models are a good match for weakly supervised localization, and (ii)

Table 6: Localization performance for traffic signs. Configuration B denotes strongly supervised baseline (HOG [52]); the respective execution time includes both t_{op} and t_{lf}. The remaining notation is the same as in previous tables.

Nr.	Conf.	FV Norm.	Penalty	ACD	AP test	p_{miss}	t_{op}/s	speed-up
1	B	-	l_2	-	88	**0.05**	10.0	
2	M	-	l_2	100.0	66	0.27	8.9	
3	M	-	l_1	6.1	81	0.21	1.9	
4	M	-	$\ell_{2,1}$	1.1	84	0.23	0.1	
5	M	p, ℓ_2 global	$\ell_{2,1}$	1.1	**92**	0.15	27.1	
6	M	p, ℓ_2 intra	$\ell_{2,1}$	0.8	88	0.13	0.3	
7	G	p, ℓ_2 global	$\ell_{2,1}$	1.1	87	0.15	**0.1**	226.0×
8	G	p, ℓ_2 intra	$\ell_{2,1}$	0.8	86	**0.12**	**0.1**	4.5×

that gradient approximation retains localization accuracy and gains execution speed. The second section (rows #2-#4) shows that the sparse models (ℓ_1, $\ell_{2,1}$) substantially outperform the dense baseline (ℓ_2). Recognition performance is further improved by non-linear normalizations. Similarly to zebra crossings, intra-component ℓ_2 normalization yields a better p_{miss}. Contrary to zebra crossings and similarly to classification results, intra-component ℓ_2 normalization yields a lower AP. Experiments with the gradient approximation (rows #7-#8) achieve slightly worse AP than direct patch contribution (rows #5-#6). Nevertheless, gradient approximation of intra-component normalization (row #8) yields the best p_{miss} performance in all weakly supervised experiments.

Failure cases. Figure 6 shows the results of our best configuration (row #5 in Table 6). The first row shows that our method is able to successfully localize very small objects in both urban and rural traffic scenes. The second row shows two types of failure cases: (i) false positives appearing as parts of house roofs or other traffic signs seen from behind (the first two images in the second row), (ii) cases where top rated patches are located on objects, but either the patch

Figure 6: Examples of localization results for traffic signs: the top row shows correct localizations, the bottom one shows examples of failure cases. Localization responses are shown as yellow rectangles, the ground truth annotations (used for evaluation purpose only) are shown as red rectangles, centers of top rated patches are illustrated as yellow dots. The images are shown using grayscale colormap to emphasize the locations of top rated patches.

connectivity is insufficient, or there are too few patches to form a bounding box. These problems could be mitigated by exploiting the spatial relationship between the visual words to rule out patches in the background [16].

Execution time. As in the case of zebra crossings, we perform all experiments on a single core of a 2.00 GHz Intel Xeon E5-2620 CPU. We extract on average $87 \cdot 10^3$ patches from the input image, which takes $t_{\mathrm{lf}} = 0.7$ s. The soft-assign step takes approximately $t_{\mathrm{sa}} = 3.7$ s, which is a 33-fold increase with respect to the setup used for zebra crossings. This increase is a combined effect of (i) a 14-fold increase in the number of features, (ii) an 8-fold increase in the number of components (1024 vs 128) and (iii) a 6-fold decrease in feature dimensionality (80 vs 512). As in the case of zebra crossings, group sparsity and gradient approximation substantially decrease the execution time. Processing an image with the configuration #8 (cf. Table 6) takes on average 4.4 s which is more than two times faster than the strongly supervised HOG approach.

5.4. Comparison with related work

We compare our best configurations from Tables 3, 4 and 6 with the related approach [13] which is similar to our work since it also refrains from fine-tuning pre-trained convolutional features. However, instead of Fisher embedding and linear classification, they postprocess convolutional features with a mini-net comprised of two adaptation layers which are subsequently pooled and trained with the maximum likelihood classification loss. The adaptation layers are implemented as convolutions 3×3×D×2048 and 3×3×2048×C, where D is dimensionality of input features, while C is the number of classes. We have obtained best results by performing the following changes to the original setup [13]: i) we replaced the global max pool [13] with global average pool [14], ii) we trained with dropout regualrization in both adaptation layers, iii) for fairness, we use the same convolutional front-end as in our experiments (VGG conv5_4). The results are summarized in Table 7.

Table 7: Comparison of our best localization results with the related approach [13] on both datasets. We repeat the best results from Table 3, Table 4, and Table 6, and provide additional experimental results on traffic signs with VGG conv5_4 features (row #4).

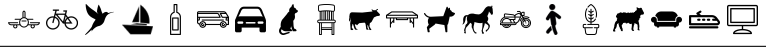
Nr.	Dataset	IoU	Features	Aggreg. & scoring	AP test	p_{miss}
1	zebras	0.1	VGG conv5_4	FV + linear	93	**0.25**
2	zebras	0.1	SIFT	FV + linear	**95**	0.26
3	zebras	0.1	VGG conv5_4	mini-net [13]	85	0.38
4	signs	0.5	VGG conv5_4	FV + linear	39	0.72
5	signs	0.5	SIFT	FV + linear	**92**	**0.15**
6	signs	0.5	VGG conv5_4	mini-net [13]	17	0.89

The first section (rows #1-#3) contains experiments on zebra crossings (IoU=0.1) while the second section (rows #4-#6) presents experiments on traffic signs (IoU=0.5). Our approach produced better localization results on both datasets. The results on traffic signs show that convolutional features pre-trained on ImageNet are unable to locate very small objects such as traffic

signs. These results would improve after fine-tuning, however that would adversely affect the scalability of the approach and simplicity of training. The results on both datasets show that SIFT features are able to outperform pre-trained convolutional features in the cases of very small objects and objects with simple structure.

Finally, we present weakly supervised localization experiments on Pascal VOC 2012. As we shall see, this dataset is not very well-suited for our method due to strong classification cues provided by context. Nevertheless, we present these results in order to enable comparison with other methods on this widely used dataset. As in Table 7, we compare our method with a mininet comprised of two 3×3 adaptation layers initialized from scratch [13]. The mininet is applied to VGG conv5_4 features, and trained with dropout and average pooling. As in Table 7, the max pooling variant resulted in very poor generalization. Log-mean-exp pooling was better than max-pooling, but still worse than average pooling which we used in the end. We believe these difficulties are due to different features [13] and much less training data than in [14]. Our method embeds VGG conv5_4 features into a Fisher space according to a GMM with 64 components (doubling this led to same results). We employ the best configuration from Table 3: power and intra-component normalization and cross-validated $\ell_{2,1}$ regularization. Our experiments measure the localization AP for the strongest per-class response [13]. The results are summarized in Table 8.

Table 8: Comparison of our method (FV) with the mininet approach [13] (MN) on Pascal 2012 val. We report classification (C) and localization performance (L). The localization performance considers only the strongest per-class response [13]. Both approaches operate on VGG conv5_4 features. In order to account for context, we postprocess our per-class responses by 6 convolutions with the 3×3 mean filter.

		mAP
FV C	98 88 95 90 72 93 86 98 77 85 71 97 92 92 97 65 90 72 96 89	**87.1**
MN C	99 88 96 90 72 93 85 98 76 87 67 97 92 92 97 61 89 67 97 84	86.2
FV L	90 83 89 45 55 84 68 93 45 60 46 94 78 80 88 39 77 **52** 79 75	71.0
MN L	94 84 91 **75** **62** 86 **76** 95 44 **66** 42 95 81 82 93 **50** 76 43 **87** 77	**75.0**

Clasification results are presented in the first two rows of the table where we see that Fisher vectors outperform mininet for 0.9pp. This is remarkable, since the employed FV pipeline has much less parameters than the mini-net approach (128 000 vs more than one million). Unfortunately, most of our models were dense (ACD $> 50\%$) which makes them unsuitable for localization. This is reflected in the localization results where mininet outperforms our approach for 4pp with the exception of the class sofa where a semi-dense model was selected (ACD $= 50\%$). These results show that our method is suitable for datasets such as those from our main experiments, where the object class can not be inferred from context.

5.5. Discussion

We have evaluated the performance of our method on two different datasets. In both cases, the experiments point out four facts: (i) sparse models achieve better localization accuracy with faster run-time, (ii) non-linear normalizations increase the localization performance of sparse models, (iii) the gradient approximation achieves comparable localization accuracy while substantially improving the run-time, and (iv) intra-component normalization increases the sparsity and improves p_{miss} when used in conjunction with group-sparse regularization while otherwise achieving negative effects.

We also note certain differences between the two datasets. For the classification task, the sparse models outperform the dense models on traffic signs (up to 9 pp), but achieve comparable performance on zebra crossings. For the localization task, the sparse models achieve better performance on both datasets. In the case of zebra crossings, there are other objects which commonly appear in positive images (e.g. traffic signs, traffic lights, or road surface markings). Hence, dense models are able to compensate the loss of focus by relying on context. Classification performance of zebra crossings is much better than for traffic signs (97% vs 81%). This performance gap can be explained as follows: (i) zebra crossings are in general larger than traffic signs and as such produce a larger contribution to image representation, (ii) other objects co-occurring with zebra

crossings in positive images provide additional classification evidence. Clearly, these co-occurring distractors make weakly supervised localization harder. Fortunately, their impact can be alleviated with sparse models. Results from Table 3 and Figure 3 show that sparse models are able to identify patches on zebra crossings as the ones responsible for the image label, while dense models are not able to ignore other content characteristic for positive images. Finally, nonlinear normalizations improve both the localization AP and p_{miss} on traffic signs, while on zebra crossings they yield comparable AP and only decrease the miss frequency.

The overall execution time for zebra crossings is 3 times faster than for traffic signs. This discrepancy is due to soft-assign stage being 33 times slower in the case of traffic signs. Our detectors may be accellerated further by using the fast approximation of the GMM soft-assignment, as proposed in [36]. We leave this, however, for future work.

6. Conclusion

We have presented a novel weakly supervised localization method based on (i) Fisher embedding of local features (CNN, SIFT) and (ii) component-level sparsity induced by $\ell_{2,1}$ regularization. The Fisher embedding allows weakly supervised training of localization models by employing image-wide labels. The induced model sparsity reduces overfitting and enables fast evaluation by effective reduction of the representation dimensionality. This dramatically improves localization accuracy over the dense Fisher vector baseline (16 pp for zebra crossings, 21 pp for traffic signs) and leads to a huge gain in the execution speed since our models use only a fraction of image representation (5% for zebra crossings, 1% for traffic signs).

In order to make this approach compatible with power and metric normalizations employed in the improved Fisher vector, we have proposed two methodological novelties. First, we have introduced a first-order approximation of the normalized FV score, which allows to determine the patch-level contribution

by a simple dot product. This approximation achieved comparable localization accuracy with respect to the direct approach, while gaining a huge overall execution speedup (around 200% on the traffic sign dataset). Second, we have proposed to use intra-component metric normalization in conjunction with the component-level sparsity. This setting further increased the model sparsity and reduced the localization miss frequency.

We have evaluated our method on two challenging road mapping datasets. First, we have introduced a novel dataset containing 2381 images of zebra crossings obtained by semi-automated matching of OpenStreetMap data (longitude, latitude) to GPS references of video frames. Second, we also show experimental results on an adapted version of a public traffic sign dataset [51]. Our study has shown that reliable general purpose object localization models can be trained from geo-referenced video and crowd-sourced image-wide labels provided by services such as OpenStreetMap, despite weak supervision signal provided by these resources. These localization models can be applied in automated road safety inspection and other kinds of road-environment mapping.

Acknowledgments

This work has been partially supported by Croatian Science Foundation under the project I-2433-2014 and the Centre of Research Excellence for Data Science and Advanced Cooperative Systems.

This work was partially supported by the grants ANR-16-CE23-0006 and ANR-11-LABX-0025-01.

The authors would like to thank Marko Ševrović and Mario Miler for their kind assistance in accessing the employed geo-referenced videos.

Appendix A. Gradient of the normalized image score

In this appendix, we derive the expression for gradient approximation given by the equation (9). We consider $\mathbf{X}$ to be an un-normalized full image Fisher vector, and X_d to be a d-th dimension of the corresponding vector. The Fisher

vector $\mathbf{X}$ is first normalized over each dimension d with the power normalization $\mathbf{s}(\mathbf{X}) = \mathbf{sign}(\mathbf{X})|\mathbf{X}|^{\rho}$. The partial derivative of the vector $\mathbf{s}(\mathbf{X})$ with respect to X_d is given by:

$$\frac{\partial \mathbf{s}(\mathbf{X})}{\partial X_d} = [0 \dots \rho |X_d|^{\rho-1} \dots 0] \; . \tag{A.1}$$

After the power normalization, we apply the metric normalization by dividing with the square root of $n(\mathbf{X}) = \mathbf{s}(\mathbf{X})^{\top}\mathbf{s}(\mathbf{X})$. Throughout this appendix, we assume the global ℓ_2 normalization for the sake of simplicity. The proposed reasoning also holds for intra-component normalization, where we substitute the $n(\mathbf{X})$ with $n(\mathbf{X}^k)$, where $\mathbf{X}^k$ corresponds to the k-th Gaussian statistics from the descriptor $\mathbf{X}$. The partial derivative of the scalar $n(\mathbf{X})$ with respect to X_d is given by:

$$\begin{aligned} \frac{\partial n(\mathbf{X})}{\partial X_d} &= 2 s(X_d) \frac{\partial s(X_d)}{\partial X_d} && \text{(A.2)} \\ &= 2 s(X_d) \, \rho |X_d|^{\rho-1} \; . && \text{(A.3)} \end{aligned}$$

The classification score of the normalized image FV is computed as $f(\mathbf{X}) = \mathbf{w}^{\top}\mathbf{s}(\mathbf{X})/\sqrt{n(\mathbf{X})}$. Thus the gradient $\nabla_{X_d} f(\mathbf{X})$ can be expressed as:

$$\begin{aligned} \frac{\partial f(\mathbf{X})}{\partial X_d} &= \frac{\partial \mathbf{w}^{\top}\mathbf{s}(\mathbf{X})/\sqrt{n(\mathbf{X})}}{\partial X_d} \\ &= \frac{1}{\sqrt{n(\mathbf{X})}} \frac{\partial \sum_d w_d \cdot s(X_d)}{\partial X_d} + \mathbf{w}^{\top}\mathbf{s}(\mathbf{X}) \frac{\partial [n(\mathbf{X})]^{-0.5}}{\partial X_d} \\ &= \frac{w_d}{\sqrt{n(\mathbf{X})}} \frac{\partial s(X_d)}{\partial X_d} - 0.5 \cdot \frac{\mathbf{w}^{\top}\mathbf{s}(\mathbf{X})}{[n(\mathbf{X})]^{1.5}} \frac{\partial n(\mathbf{X})}{\partial X_d} \end{aligned} \tag{A.4}$$

When we substitute expressions for power and metric derivatives (A.1) and (A.3), we arrive to the equation (9):

$$\begin{aligned} \frac{\partial f(\mathbf{X})}{\partial X_d} &= \frac{w_d}{\sqrt{n(\mathbf{X})}} \rho |X_d|^{\rho-1} - 0.5 \cdot \frac{\mathbf{w}^{\top}\mathbf{s}(\mathbf{X})}{[n(\mathbf{X})]^{1.5}} \cdot 2 \cdot s(X_d) \cdot \rho |X_d|^{\rho-1} && \text{(A.5)} \\ &= \frac{\rho |X_d|^{\rho-1}}{\sqrt{n(\mathbf{X})}} \left(w_d - \frac{\mathbf{w}^{\top}\mathbf{s}(\mathbf{X})}{n(\mathbf{X})} \cdot s(X_d) \right) && \text{(A.6)} \\ &= \frac{\rho |X_d|^{\rho-1}}{\sqrt{n(\mathbf{X})}} \left(w_d - \frac{f(\mathbf{X}) \, s(X_d)}{\sqrt{n(\mathbf{X})}} \right) && \text{(A.7)} \end{aligned}$$

References

[1] M. Pandey, S. Lazebnik, Scene recognition and weakly supervised object localization with deformable part-based models, in: ICCV, 2011, pp. 1307–1314.

[2] A. J. Bency, H. Kwon, H. Lee, S. Karthikeyan, B. S. Manjunath, Weakly supervised localization using deep feature maps, in: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I, 2016, pp. 714–731.

[3] R. G. Cinbis, J. J. Verbeek, C. Schmid, Weakly supervised object localization with multi-fold multiple instance learning, IEEE Trans. Pattern Anal. Mach. Intell. 39 (1) (2017) 189–203.

[4] E. J. Crowley, A. Zisserman, Of gods and goats: Weakly supervised learning of figurative art, in: BMVC, 2013.

[5] O. Chum, A. Zisserman, An exemplar model for learning object classes, in: CVPR, 2007.

[6] C. Galleguillos, B. Babenko, A. Rabinovich, S. J. Belongie, "Weakly Supervised Object Localization with Stable Segmentations", in: ECCV, 2008, pp. 193–207.

[7] P. Siva, T. Xiang, "Weakly supervised object detector learning with model drift detection", in: ICCV, 2011, pp. 343–350.

[8] T. Deselaers, B. Alexe, V. Ferrari, "Weakly Supervised Localization and Learning with Generic Knowledge", IJCV 100 (3) (2012) 275–293.

[9] H. Bilen, M. Pedersoli, T. Tuytelaars, Weakly supervised object detection with convex clustering, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, 2015, pp. 1081–1089.

[10] H. Bilen, A. Vedaldi, Weakly supervised deep detection networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2016, pp. 2846–2854.

[11] A. Diba, V. Sharma, A. M. Pazandeh, H. Pirsiavash, L. V. Gool, Weakly supervised cascaded convolutional networks, CoRR abs/1611.08258.

[12] M. H. Nguyen, L. Torresani, F. D. la Torre, C. Rother, "Learning discriminative localization from weakly labeled data", Pattern Recognition 47 (3) (2014) 1523–1534.

[13] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Is object localization for free? - weakly-supervised learning with convolutional neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, 2015, pp. 685–694.

[14] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2016, pp. 2921–2929.

[15] J. Krapac, S. Šegvić, Weakly supervised object localization with large Fisher vectors, in: VISAPP, 2015.

[16] V. Zadrija, J. Krapac, J. J. Verbeek, S. Šegvić, Patch-level spatial layout for classification and weakly supervised localization, in: Pattern Recognition - 37th German Conference, GCPR 2015, Aachen, Germany, October 7-10, 2015, Proceedings, 2015, pp. 492–503.

[17] Y. Zhu, Y. Zhou, Q. Ye, Q. Qiu, J. Jiao, Soft proposal networks for weakly supervised object localization, in: IEEE International Conference on Computer Vision, ICCV, Venice, Italy, 2017, pp. 1859–1868.

[18] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", in: ICLR, 2015.

[19] J. Sánchez, F. Perronnin, T. Mensink, J. J. Verbeek, "Image Classification with the Fisher Vector: Theory and Practice", IJCV 105 (3) (2013) 222–245.

[20] Z. Qiu, T. Yao, T. Mei, Deep quantization: Encoding convolutional activations with deep generative model, CoRR abs/1611.09502.
URL http://arxiv.org/abs/1611.09502

[21] M. Cimpoi, S. Maji, I. Kokkinos, A. Vedaldi, Deep filter banks for texture recognition, description, and segmentation, International Journal of Computer Vision 118 (1) (2016) 65–94.

[22] R. Jenatton, J. Mairal, G. Obozinski, F. Bach, Proximal methods for hierarchical sparse coding, J. Mach. Learn. Res. 12 (2011) 2297–2334.

[23] D. Novotný, D. Larlus, F. Perronnin, A. Vedaldi, Understanding the fisher vector: a multimodal part model, CoRR abs/1504.04763.
URL http://arxiv.org/abs/1504.04763

[24] D. Ahmetovic, R. Manduchi, J. M. Coughlan, S. Mascetti, Zebra crossing spotter: Automatic population of spatial databases for increased safety of blind travelers, in: Y. Yesilada, J. P. Bigham (Eds.), Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility, ASSETS 2015, Lisbon, Portugal, October 26-28, 2015, ACM, 2015, pp. 251–258. doi:10.1145/2700648.2809847.
URL http://doi.acm.org/10.1145/2700648.2809847

[25] D. Koester, B. Lunt, R. Stiefelhagen, Zebra crossing detection from aerial imagery across countries, in: K. Miesenberger, C. Bühler, P. Penáz (Eds.), Computers Helping People with Special Needs - 15th International Conference, ICCHP 2016, Linz, Austria, July 13-15, 2016, Proceedings, Part II, Vol. 9759 of Lecture Notes in Computer Science, Springer, 2016, pp. 27–34. doi:10.1007/978-3-319-41267-2_5.
URL http://dx.doi.org/10.1007/978-3-319-41267-2_5

[26] Google Inc, ICMLA 2011 StreetView Recognition Challenge, Online; accessed 2016-11-22 (2016).
URL http://www.icmla-conference.org/icmla11/challenge.htm

[27] Promet i Prostor, E-roads web platform, Online; accessed: 2016-09-17.
URL https://prometiprostor.hr/en/solutions/

[28] J. Sánchez, F. Perronnin, T. Mensink, J. Verbeek, Image Classification with the Fisher Vector: Theory and Practice, International Journal of Computer Vision 105 (3) (2013) 222–245.

[29] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, J. R. Stat. Soc. Ser. B Stat. Methodol. 68 (1) (2006) 49–67.

[30] F. Perronnin, J. Sánchez, T. Mensink, Improving the Fisher kernel for large-scale image classification, in: ECCV, 2010, pp. 143–156.

[31] T. Jaakkola, D. Haussler, Exploiting generative models in discriminative classifiers, in: NIPS, 1998, pp. 487–493.

[32] F. Perronnin, C. R. Dance, Fisher kernels on visual vocabularies for image categorization, in: 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA, 2007.

[33] R. Arandjelović, A. Zisserman, All about VLAD, in: IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[34] C. Bishop, Pattern recognition and machine learning, Springer, New York, 2006.

[35] Z. Zhang, Y. Xu, J. Yang, X. Li, D. Zhang, A survey of sparse representation: Algorithms and applications, IEEE Access 3 (2015) 490–530.

[36] J. Krapac, S. Šegvić, Fast Approximate GMM Soft-Assign for Fine-Grained Image Classification with Large Fisher Vectors, in: GCPR, 2015.

[37] P. A. Viola, M. J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.

[38] OpenStreetMap Wiki, Map Features — OpenStreetMap Wiki,, online; accessed 2016-11-14 (2016).
URL http://wiki.openstreetmap.org/w/index.php?title=Map_Features&oldid=1356182

[39] Geofabrik, OpenStreetMap Data Extracts, Online; accessed: 2016-11-14.
URL http://http://download.geofabrik.de/

[40] OpenStreetMap Foundation, OpenStreetMap, Online; accessed 2016-11-14 (2016).
URL https://www.openstreetmap.org

[41] OpenStreetMap Wiki, Overpass API — OpenStreetMap Wiki,, Online; accessed 2016-09-17.
URL http://wiki.openstreetmap.org/wiki/Overpass_API

[42] OpenStreetMap Wiki, Video mapping — OpenStreetMap Wiki,, Online; accessed 2016-11-14 (2016).
URL http://wiki.openstreetmap.org/w/index.php?title=Video_mapping&oldid=1316747

[43] Mapillary, Mapillary mobile application, online; accessed: 2016-11-16.
URL https://play.google.com/store/apps/details?id=app.mapillary&hl=en

[44] Telenav GmbH, OpenStreetView mobile application, Online; accessed: 2016-11-16.
URL https://play.google.com/store/apps/details?id=com.telenav.streetview&hl=en

[45] M. Everingham, L. Gool, C. K. Williams, J. Winn, A. Zisserman, The Pascal visual object classes (VOC) challenge, Int. J. Comput. Vision 88 (2) (2010) 303–338.

[46] K. Chatfield, V. S. Lempitsky, A. Vedaldi, A. Zisserman, The devil is in the details: an evaluation of recent feature encoding methods, in: British Machine Vision Conference, BMVC 2011, Dundee, UK, August 29 - September 2, 2011. Proceedings, 2011, pp. 1–12.

[47] R. M. French, Catastrophic forgetting in connectionist networks: Causes, consequences and solutions, Trends in Cognitive Sciences 3 (4) (1999) 128–135.

[48] M. Douze, H. Jégou, The Yael library, in: Proceedings of the ACM International Conference on Multimedia, 2014.

[49] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding, J. Mach. Learn. Res. 11 (2010) 19–60.

[50] Inland transport comitee, Convention on road signs and signals (1968).

[51] K. Brkić, A. Pinz, S. Šegvić, Z. Kalafatić, Histogram-based description of local space-time appearance, in: SCIA, 2011, pp. 206–217.

[52] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, 2005.