Video-based Deep Matching for Object Occlusion Detection

Javier Martínez Samblas
Signal Theory and Communications Department
Universidad Carlos III de Madrid
jmsamblas@tsc.uc3m.es

Iván González Díaz
Signal Theory and Communications Department
Universidad Carlos III de Madrid
igonzalez@tsc.uc3m.es

***Abstract*— This paper proposes a system capable of automatically detecting occlusions of roadway traffic milestones, with the goal of improving the detection performance of these signals and decreasing the missing rates. To this purpose, we propose a matching system based on the well-known CNNs for image retrieval and extend it with a novel weak learning algorithm that fits better our problem and application scenario. Our results demonstrate that our proposal improves the results of the baseline CNN and outperforms traditional approaches as SIFT.**

***Keywords* — Computer Vision, Image Matching, Deep Learning, Weak Learning, Contrastive Loss.**

I. INTRODUCTION

Efficient transport infrastructure greatly contributes to economic development and provides indisputable social benefits. Since 2010, over 20.000 lives have been saved on the roads exclusively due to the ETSC's PIN programme in the European Union [1]. However, an extra of 8470 deaths could have been prevented if the original budget planning had been rigorously followed. Road maintenance is a really expensive task. Even though roughly more than half of the infrastructure investment of a country is employed in the maintenance of the roadways, it is still insufficient [2]. Moreover, the future advent of self-driving cars will require smarter roads and cheaper ways to constantly maintain them.

A critical factor in road maintenance is the assurance of an adequate traffic signaling. If there are inconsistencies or missing signals, the risk of accident drastically increases. In the GPM's research group of UC3M we are currently working on developing a system that automatically detects traffic signals such as delineator posts or edge milestones. The goal of this system is to spot any kind of disrepair in the signals, *e.g.* they are absent or broken, in order to refurbish them and increase the driving safety. However, this paper is not focused on the detection of these signals but on the identification of cases in which they are occluded by front objects such as cars or trucks, with the objective of decreasing the missing rate of the detector and thus improving the performance. Trying to directly detect some signs in cities can be challenging as the number of cars is usually large and so is the probability of occlusion. Thus, developing a system capable of finding hidden signs is essential for the task at hand. Figure 1 shows a common scenario where this situation occurs: an overtaking on the left. Hopefully, our research will help to automate arduous common chores in the maintaining of the roadways, improving their sustainability and reducing the labor costs to a high degree.

We propose to address this problem as follows: given some sets of video sequences that belong to the same roadway but were recorded in different days, the goal is to compare them frame-to-frame in order to determine correspondences between images that could contain the same traffic signal. Hence, for some certain location (in a latitude-longitude GPS pair format fetched from the video metadata) where a signal should have been detected but was not, nearby frames of different days are compared by means of an image matching system so that the unseen signal can be finally labeled as missing or present. Thus, we tackle the problem similarly to a CBIR (Content-based Image Retrieval) scheme where most akin images from a determined database need to be retrieved for some input queries. The main difference lies in the fact that in our problem we just need to retrieve the most similar image, instead of having to accurately spell out the whole ranking of them. Note that this matching system is required and essential for the motif, since we cannot fully trust GPS coordinates to detect occlusions due to the location error inherent to the GPS sensor embedded in the camera system. Concretely, the problem is accentuated in those cases where the traffic signals are pretty close to each other so that there is no way to discern between them just by inspecting the GPS data.

Next, we list our main contributions of the project that will be presented throughout the paper:

- We have built an image matching pipeline to address the occlusion detection problem.
- We have designed a weak learning approach that adapts better to our real problem in which only the most similar image has to be detected and data labels are noisy (due to GPS errors).
- We have compared our approach with the baseline and other traditional reference methods based on local descriptors.

The rest of the paper is organized as exposed along the next lines. Section II reviews two of the most influential algorithms - SIFT and CNNs - in the history of computer vision. In Section III we present the database of the project. Section IV exposes the proposed system, explaining which is the CNN baseline and what is the weak learning approach we used to improve it. Section V presents the experimental setup of the project as well as the evaluation metric and results. Conclusions and future lines of research are drawn in Section VI. Finally, Appendix I looks over the time management of

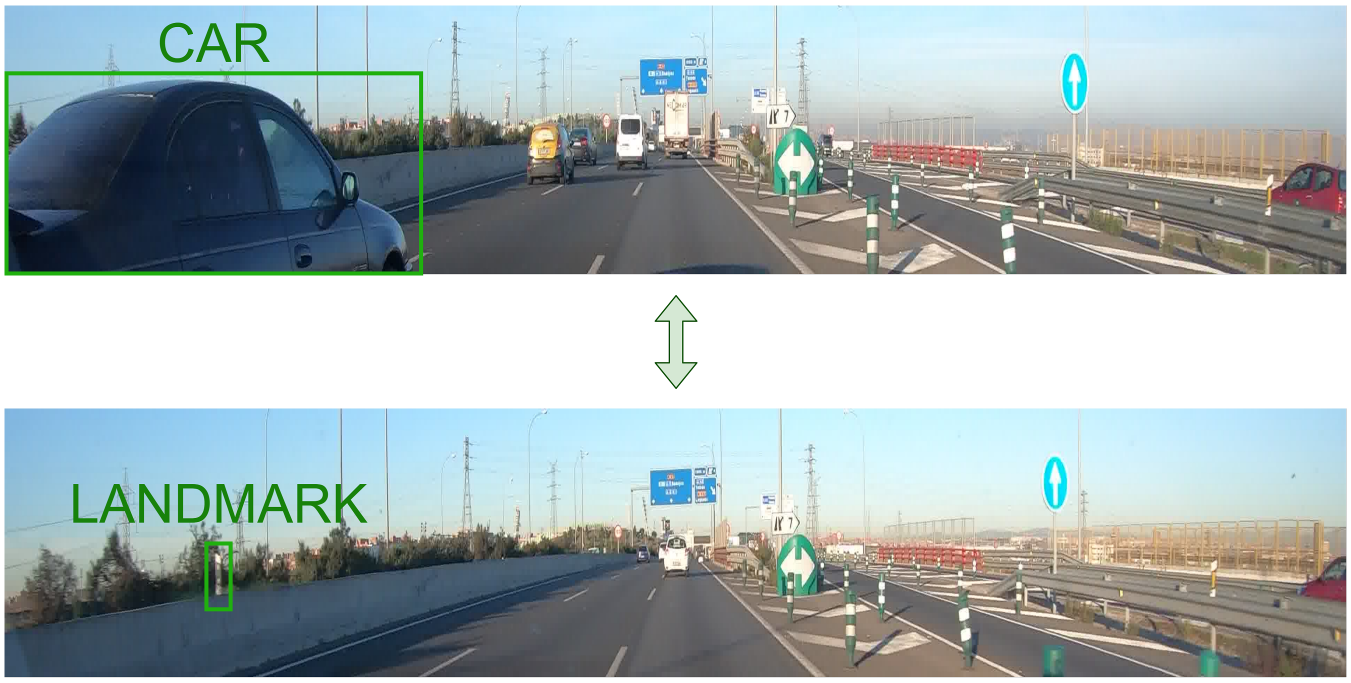

Fig. 1: Top: Image where a car is occluding a landmark on the left side of the road. Bottom: Image from a different day of the same spot of the roadway that does not present any landmark occlusions.

the project.

II. RELATED WORK

Human vision is a tremendously complex and powerful system. By visualizing an image for just a time instant our visual system is able to perceive and deduce a large amount of information about it; we are not only able to classify different objects that appear in the image, but to precisely detect their boundaries or even to answer context-related questions apparently effortlessly [3]. Thus, the action of comparing different images and analyzing the common matching features remains trivial for our brain. However, this seems a really hard task for a computer that only interprets images as matrices of real numbers.

The simplest approach one could think of is just frame differencing in a pixel-wise way. Following this, different images should return high score values (large differences) whereas similar images should return low ones. Predictably, this method is very far from reaching human performance since it doesn't take into account the variability (*i.e.* view-point, scale, rotation and illumination variations, occlusions, *etc.*) of the items that appear in the image nor their physical nature.

A huge variety of algorithms have appeared since the 60s [4] in order to imitate our brain's behavior. Nevertheless and broadly speaking, algorithms in computer vision can be classified into two general groups: local feature based methods and CNN based methods.

Local feature based methods. Local feature methods are based on the detection and description of distinct patterns within the images like edges, corners or blobs. These structures are commonly denominated as local features and are the basis of many computer vision algorithms [5]. Famous algorithms *s.a.* Canny edge detector [6], Harris corner detector [7]) or SIFT (Scale-Invariant Feature Transform) [8] belong to this group. In the following paragraphs, we will focus on the explanation of SIFT due to its historical importance [9] [10].

SIFT was published by David G. Lowe in 2004 and it is an algorithm capable of detecting and describing the most relevant local features of an input image. The main advantage of SIFT that supposed a breakthrough versus older feature detectors is its scale-invariant nature. Traditional methods were robust to limitations such as illumination, translation (*e.g.* Canny edge detector) and even rotation changes (*e.g.* Harris corner detector). However, all of them resoundingly failed when trying to detect and describe setups of different dimensions or scales. SIFT easily solves this problem by construction, as it finds local features or *keypoints* in a space of different scales (*a.k.a.* scale-space [11]). In its original formulation, SIFT finds the keypoints by identifying blobs in the image. Blobs are not concrete points but local regions where visual properties such as color or illumination remain constant. The first blob detector ever designed was the LoG (Laplacian of Gaussian) [12]. However, to speed up the detection process, SIFT uses the DoG (Difference of Gaussians) [13] instead, which is essentially an approximation of the LoG operator. It greatly eases the implementation of the scale-space as it is based on the consecutively building of gaussian filters that can be re-used to obtain the DoG spaces by just differencing them. Keypoints, alongside the scales

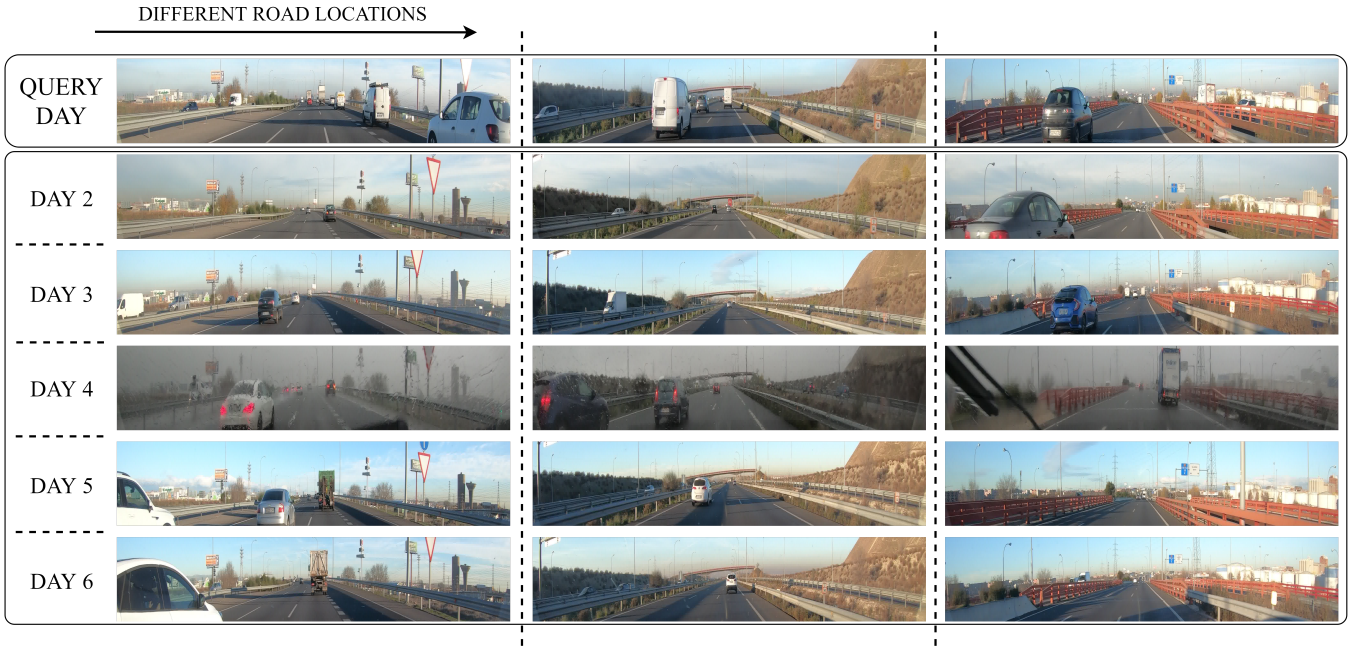

Fig. 2: Database scheme. Columns show different road locations whereas rows indicate different recording days. The first row is the query day and it always presents car occlusions.

where they appear, are then collected by obtaining the DoG extrema and refined with a peak threshold parameter that filters out the smallest ones.

Even though DoG scheme has been widely used in the computer vision community, it has been proven that subsequent formulations that use different blob detectors, specially affine-covariant ones, generally achieve better results [14] [15]. Affine-covariant refers to the detection of regions whose shapes proportionately change with affine transformations [16]. Thus, they offer a reliable keypoint detection even under large viewpoint shifts. Hessian-Laplace [17] [18] is an example of these kind of methods. It not only singles out keypoints in the scale-space but also their elliptical regions. The detection of the interest points is carried out by finding the extrema of the hessian matrix, as second derivatives provide intense responses to ridges and blobs. Scales are chosen by applying the laplacian operator to the detected keypoints over different scales and selecting the scale that provides the maximum value (commonly known as characteristic scale [19]). Lastly, the shape of the elliptical region is procured by obtaining the eigenvalues (normally with an iterative estimation algorithm [20]) of the second moment matrix (*a.k.a.* autocorrelation matrix). This matrix depict the gradient distribution of a keypoint in its neighborhood and it is calculated for every keypoint. Both detectors, DoG and Hessian-Laplace are designated to be a baseline for CNN methods and are compared in section V in terms of accuracy and computational efficiency.

Regarding the description of the keypoints, the original SIFT descriptor has been shown to be very robust, attaining competitive results in several benchmarks [21]. It associates a 128-dimensional vector to every keypoint by analyzing its surroundings with a HOG (Histogram of Oriented Gradients) procedure. Once the keypoints have been described, a matching scheme can be set by just calculating the euclidean distances between feature vectors.

The revolution of deep learning. Deep learning methods started to become popular in the 2010s, specially after 2012 when the architecture AlexNet [22] won the ILSVRC competition by a great margin, achieving unprecedented results in the field of image classification. This milestone carried out a paradigm shift in computer vision towards deep learning. Since then, most of the research has been focused on developing convolutional neural networks, which have been proven to outperform (in both, accuracy and computational efficiency) local feature based methods in many scenarios such as object detection or image classification. In image retrieval and feature matching problems, CNN architectures, specially finetuned ones, have achieved competitive results in a lot of benchmarks (*e.g.* Holidays or Ukbench datasets) [23] [24] [25] even though there are some cases where algorithms like SIFT still obtain slightly better accuracy scores at the expense of higher memory costs [26].

Besides efficiency benefits, one of the main advantages of deep learning approaches is the ability of automatically learn features, as opposed to hand-crafted local feature engineering. Convolutional layers are able to learn the filters by themselves via the training process which usually leads to a better generalization as long as the training datasets are big enough. Also, transfer learning - or the process by which the weights learned by a determined network when solving a different problem are re-used for solving a new problem into consideration - has helped to popularize CNNs in the recent years since it allows to build and train accurate models much

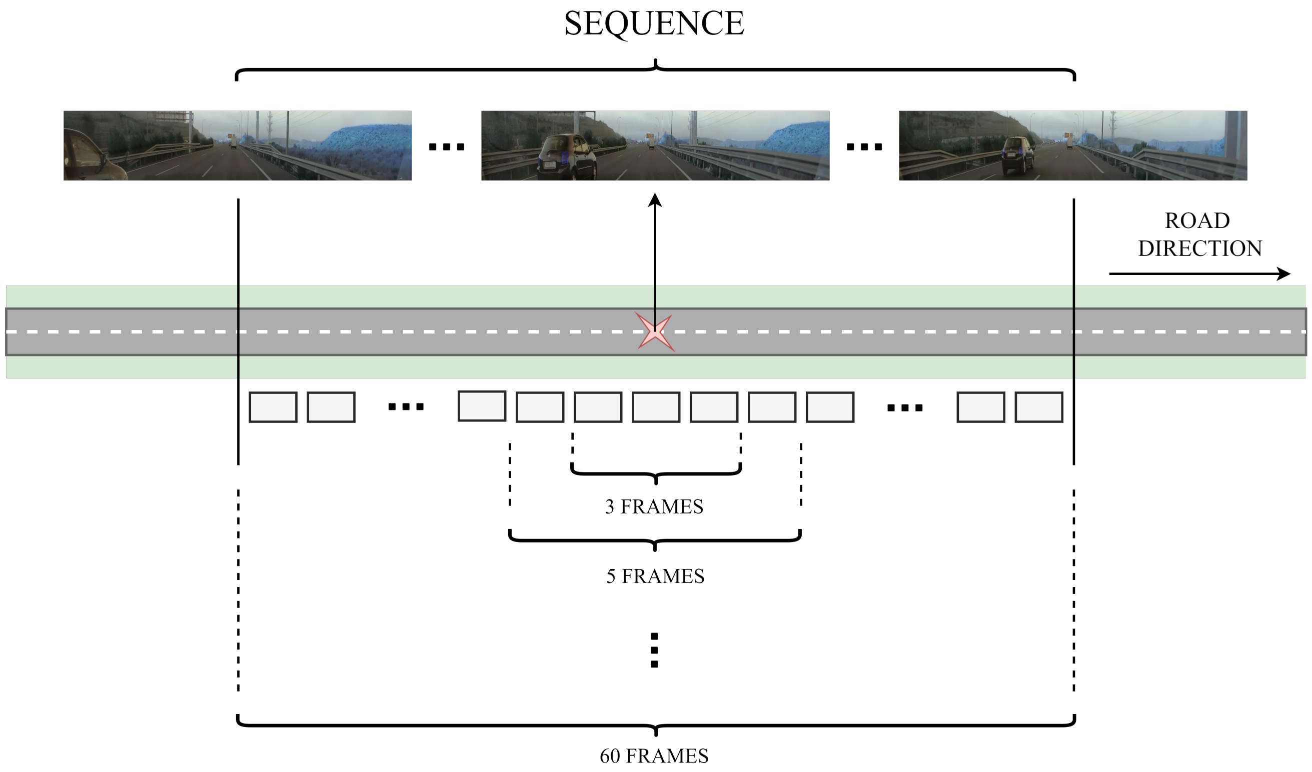

Fig. 3: Sequence outline.

faster and with a considerably smaller amount of data.

III. DATABASE DESCRIPTION

In this section, we explain the database of the project as well as the corresponding processing that allowed it to be generated.

Video recording and pre-processing. The database of the project is composed of six 4K (*i.e.* 3840×2160 *px*) videos recorded with a GoPro camera from the same roadway route (of roughly 25 *km*) but in different days. Thus, each video was recorded with different illumination conditions. Regarding meteorological conditions, only one of them was recorded in a rainy environment. Due to the enormous frame resolution and for practical reasons, videos were additionally trimmed and cropped in their vertical dimension, resulting in horizontal videos of size 3840×400 *px* that point out the center of the carriageway.

Car detection. One of the recordings was selected as the query video or the video where occlusions of traffic milestones are supposed to happen. Then, we automatically detected all the cars present in the video that could be occluders by virtue of a Faster R-CNN network trained on PASCAL VOC [27] and whose backbone was ResNet-50 [28]. Faster R-CNN is a multi-stage object detector algorithm (in contrast to one-shot detectors *s.a.* YOLO [29]) that first generates interest regions via a RPN (Region Proposal Network) and then classifies and detects the discernible items of each of these regions, giving out as output their corresponding bounding boxes and object scores. We picked up those frames that had at least one car detection with a probability score greater than $th_{score} = 0.999$. Thereby, we ensured that the probability of false alarm remains minimal so that the number of bad car annotations for training the models is insignificant.

Database alignment. Next, we made assignments between the frames of the query video and the frames of all the other videos by calculating their corresponding haversine GPS distances [30] and selecting those frames that obtained the minimum distance. Previously to this step, a linear interpolation between the GPS data and the frames was carried out, since their corresponding rates were recorded differently (18 and 30 *Hz*, respectively). All of these processes resulted in a database with six videos of 6000 frames each, aligned with each other, deriving in a total number of 36000 images. An example of this database is shown in Figure 2. Each column is a different GPS spot of the roadway whereas rows indicate the day images were recorded. Note that images of the first row are query images and always show at least one car occlusion, which is usually an overtaking on the right or on the left, the most typical scenario of occlusion.

Sequence grouping. Ultimately, frames were grouped into different non-overlapping video sequences. Each of these represent the occurrence of one traffic milestone or landmark, from the point where it begin to be visually distinguishable to the point where it fully disappears. In total, 200 consecutive sequences of frames were created. These sequences were automatically selected by analyzing the speed metadata from the GoPro accelerometer, resulting in groups of 20-60 frames

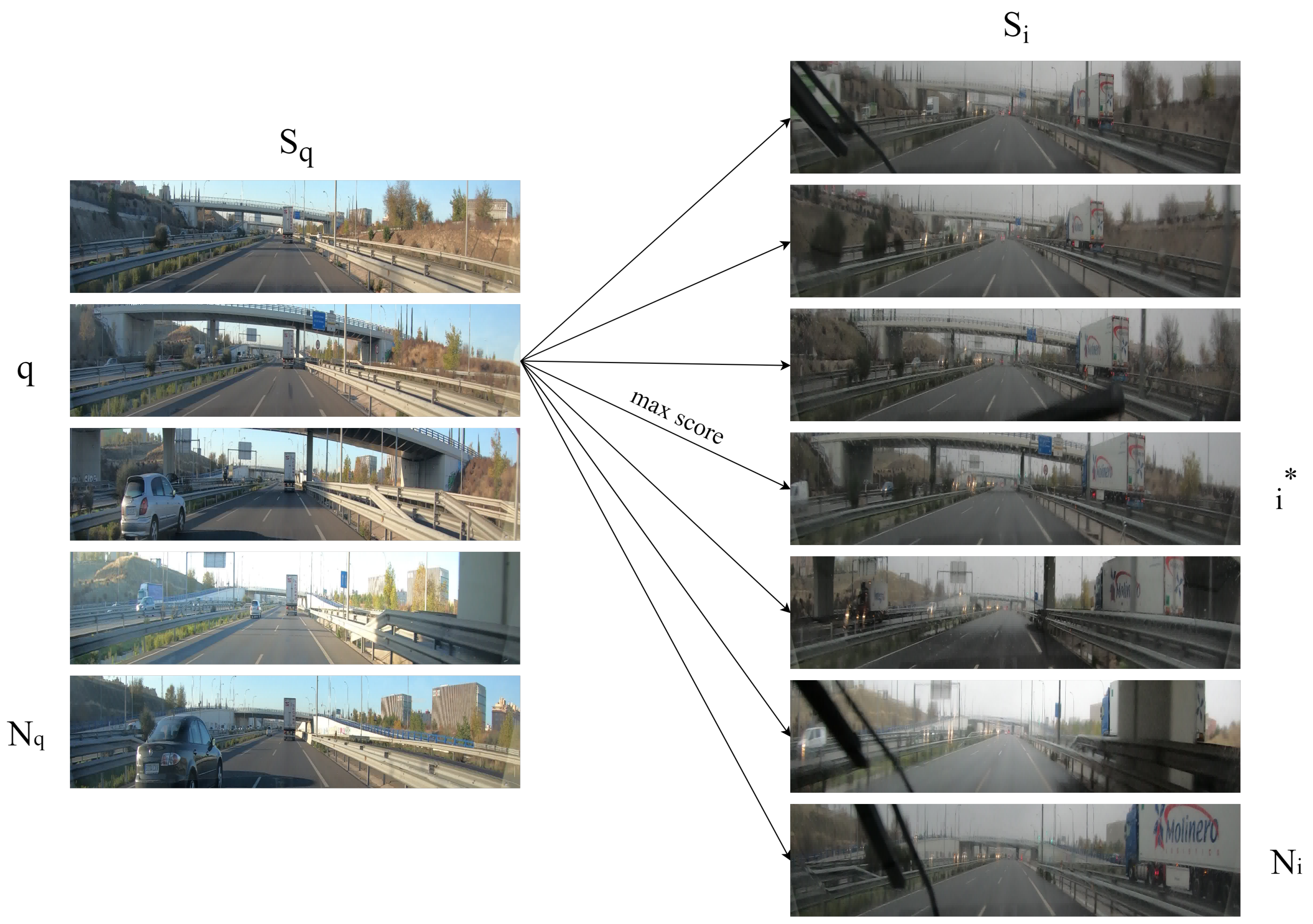

Fig. 4: Matching scheme between subsets S_q and S_i. Images q and i^* are supposed to obtain the highest similarity score as they represent the same road location.

(depending on the speed). This organization of the dataset into groups of frames is essential for the evaluation of the system as well as for the training process. Figure 3 displays this scheme, where a sequence of video is shown with a size up to 60 frames.

IV. PROPOSED SYSTEM

In this section, we explain the methods we used to assess if the images taken from the recordings of different videos represent the same traffic milestone. This problem can be outlined similarly to a general image retrieval problem that consists of two main steps: feature extraction and feature matching. We used a deep learning approach to solve it.

A. *Problem statement*

The problem can be defined as follows. Let S_q be a subset of N_q frames that have been picked from the query video and let S_i be a subset of N_i frames that belong to a set of corresponding aligned groups in the dataset. Then, given a specific query image $q \in S_q$, the goal of the project lies in finding its most *visually similar* image $i^* \in S_i$ (see Figure 4). Hence, if there is a signal in i^* in the area where a car was detected, we can establish the correspondence and assume that the car is occluding it. Conversely, if there is no signal in that area of i^*, we can assume that there is no signal present in that location, so it may be a possible missing landmark.

B. *A baseline CNN system for image matching*

To find similarities between images we first need to extract features or information from them that we can actually compare. This can be achieved by means of any of the algorithms introduced in Section II. In our case, we chose ResNet-101 [28] as the CNN algorithm to extract features from. Our choice is based on the fact that residual networks have become one of the most prosperous deep learning networks in the recent years [31] [32], as they solve the degradation problem, a common problem that makes very deep networks work worse as the number of layers increases. This is solved by adding skip connections to some layers of the network, which end up learning not only their inputs but also the residual of their outputs. Thus, in the extreme case where the parameters of a certain residual layer are all zeroes, the network can simply learn the identity (the input) and hence the information of that layer is not lost through the resting deeper layers of the network, which additionally

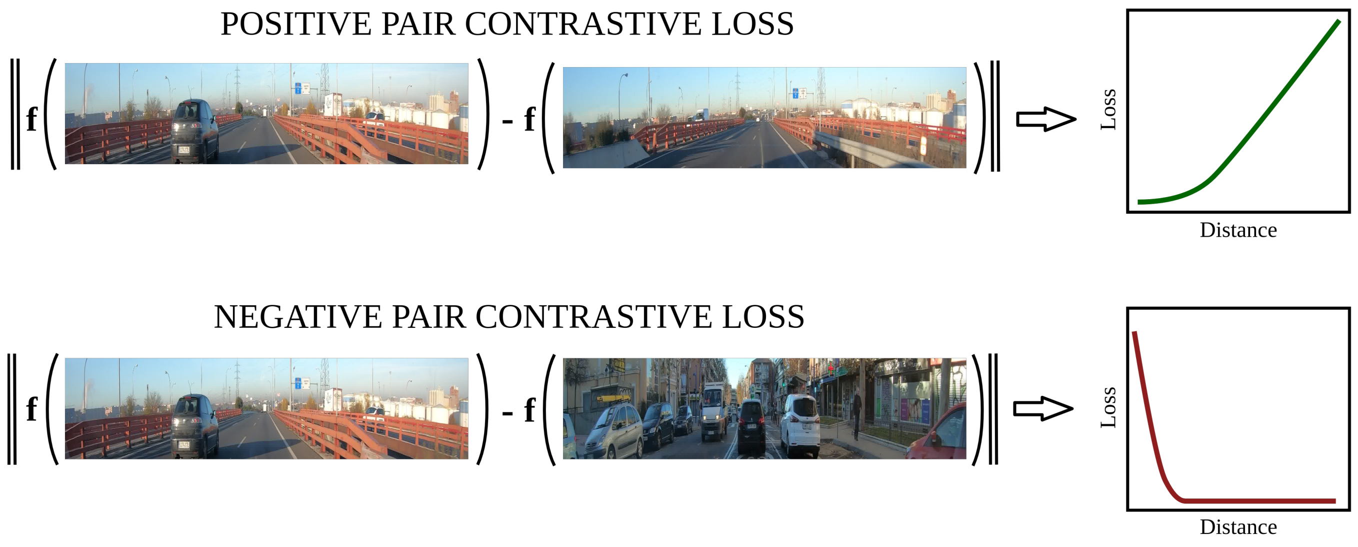
POSITIVE PAIR CONTRASTIVE LOSS
f
- f
Loss
Distance
NEGATIVE PAIR CONTRASTIVE LOSS
f
- f
Loss
Distance

Fig. 5: Siamese learning scheme.

alleviates the problem of vanishing gradients [33] [34] when backpropagating in the training process.

Baseline CNN. ResNet-101 is a fully convolutional network [35] (*i.e.* there are no fully connected layers) made up of 101 layers, including standard convolutional layers, as well as skip connections, activation functions and pooling layers in between. Activation functions are located after each of the convolutional layers and are used in order to add non-linearities to the network, which naturally breaks the linearity of the model, making it capable of solving non-linearly separable problems. The activation function it uses is the ReLU [36] which makes zero all negative values within the output volume, molding it into a non-negative tensor. Pooling layers allow to control the spatial dimensionality of the model, diminishing overfitting, reducing the computational cost and easing the management of the data structures within the layers. They also provide translation invariance to a certain degree [37]. ResNet-101 uses several max-pooling layers throughout the network which help extracting the sharpest (lower-level) features of the images.

Architecture for image retrieval and training procedure. Apart from max-pooling, an special pooling layer is the located at the end of the architecture, after the last convolutional layer. It is usually the mechanism that transforms the last activation volume into a manageable feature vector that can be used in a matching system. In our case, we scale down the activation volume by utilizing the GeM [38] [39] as the subsampling method. Let $\mathcal{X}_o$ be the output tensor of the last convolutional layer, after passing through its corresponding ReLU so that every value of $\mathcal{X}$ is not negative. From there on, the 3-dimensional tensor $\mathcal{X}_o$ can also be discerned as a set, $\mathcal{X}_{o_K}$, composed of K 2-dimensional tensors of shape $W_o \times H_o$ (*a.k.a.* feature maps), where K is the number of learning filters of the last layer. Then, the output of the GeM pooling layer is a feature vector, $\mathbf{f} = [\mathrm{f}_1 \dots \mathrm{f}_k \dots \mathrm{f}_K]^\top$, that can be written as:

$$\mathrm{f}_k(x) = \left(\frac{1}{|\mathcal{X}_k|} \sum_{x \in \mathcal{X}_k} x^{p_k} \right)^{\frac{1}{p_k}} \tag{1}$$

Note that when $p_k \to \infty$ it performs the same operation as the max-pooling layer, whilst if $p_k \to 0$ it acts as an average-pooling one by taking the average of each of the 2-dimensional tensors instead of the maximum value. Additionally, the parameter p_k can be learned via the training process, which have been proven to return very good results [40].

Concerning the training of the model, we followed a siamese two-branch architecture approach. Both branches use the same ResNet-101 model and hence share the same parameters. Each of them receive a different image as input so that output vectors are also different and can be compared in order to determine their visual similarity.

We use the contrastive loss [41] to assure that semantically akin images are embedded close together. Given an input pair (a, b) with labels $Y(a, b) \in \{0, 1\}$ where 0 denotes a negative pair and 1 a positive one, the contrastive loss between images a and b can be defined as:

$$\mathcal{L}(a,b) = \begin{cases} \frac{1}{2} D(a,b)^2, & \text{if } Y(a,b) = 1 \\ \frac{1}{2}(\max\{0, \tau - D(a,b)\})^2, & \text{if } Y(a,b) = 0 \end{cases} \tag{2}$$

where $D(a, b) = \|\overline{\mathbf{f}}(a) - \overline{\mathbf{f}}(b)\|$ refers to the euclidean distance between the output vectors of the contrasted images, $\overline{\mathbf{f}}(a)$ and $\overline{\mathbf{f}}(b)$, after passing through the GeM pooling layer and being normalized. The constant τ is a parameter that establishes the margin that the distance of a non-matching pair has to surpass in order to be ignored by the loss. Thus, if the pair is a negative pair and the distance between the images is large enough, the loss is zero. In the case that the pair is positive, the loss is zero if and only if the distance between both images is also zero. This is illustrated in the scheme of Figure 5. The margin parameter is a key parameter in siamese learning as it avoids that the network learns trivial solutions (*e.g.* predicting zero for every output vector so that distances are always zero).

Accordingly, pairs of images were picked up from the training set and labels were assigned to each of these pairs to distinguish between matching and non-matching pairs. For this purpose, we created a dataset of tuples of images composed of positives and negatives from different days. More concretely, the tuples consist of one query, five positives and six negatives.

The query corresponds to a frame of the sequence from the query day into consideration. Positives are those images that belong to the same sequence as the query, *i.e.* to the approximately same road location. Contrariwise, negatives are images that do not belong to the same sequence as the positives or the query. Specifically, negatives are hard-negatives [42], which are non-matching images that have the smallest feature vector distance among all the dataset except for the sequence in consideration. This implies that negatives are usually images that reside in contiguous sequences, as these are the hardest to distinguish *w.r.t.* the images of the sequence at issue. However, since this is not always true due to illumination changes, hard-negatives cannot be a priori hand-crafted and need to be re-calculated in every epoch of the training procedure. For each of the positives and for the query, a negative is brought back from the database, giving out a total of six negatives per tuple.

Image representation and matching. In test, our system works as follows: given an input image, the output of the system characterizes it in the form of a feature vector or a descriptor, $\mathbf{f}$, of 2048 elements, where each of these elements is the generalized mean of their respective feature mapping of the last convolutional layer (see *eq.* (1)). Now, the need arises for determining a similarity metric that can be used to compare output global descriptors. By defining a similarity metric, we can compare images all-versus-all and calculate their corresponding similarity scores so that the highest one corresponds with the most alike image to a certain query. The matching procedure is rather simple when using neural networks since the similarity metric can be defined as just the euclidean distance between the global descriptors. So, small distances will imply high scores whereas large distances mean that images are very different.

C. *Weak loss learning*

In this subsection, we describe the approach we followed in order to beat the baseline CNN. The goal of our novel weak learning proposal is twofold:

- To fit our particular task in which the query is required to be matched only to the most similar image.
- To adapt to an scenario in which labels are approximate and noisy (due to the GPS error).

We base our approach on the temporal structure of the video sequences. Assuming a perfect frame alignment (*i.e.* no GPS errors), let us first divide positive images into three subgroups according to their location within the sequences: nearest-positives, near-positives and border-positives. Nearest-positives are positive frames that are not further than two frames from the central query. Near-positives are positives that are close to the central query in the range of 2-5 frames of distance, depending on the sequence size. Lastly, border-positives are those positives that are located in the boundaries of the sequence. The visual similarity between the central query and them is the minimal one within the entire sequence due to the lower degree of overlapping. From the five positives of the tuples, one of them is a nearest-positive, two of them are near-positives and the another two stand for border-positives.

The final loss that is passed to the network during back-propagation can be thought as an additive compendium of the individual contrastive losses between the query and the positive and negative examples that coexist within the same tuple. Thus, we can describe the total loss vector per tuple with the following expression:

$$\overline{\mathbf{L}}_{tuple} = [\mathcal{L}_{p_1} \dots \mathcal{L}_{p_P}\ \mathcal{L}_{n_1} \dots \mathcal{L}_{n_N}] = \left[\overline{\mathbf{L}}_p\ \overline{\mathbf{L}}_n\right] \tag{3}$$

where $\mathcal{L}_{p_z}$ with $z \in \{1, \dots, P\}$ is the contrastive loss between the query and positive image p_z. Analogously, $\mathcal{L}_{n_z}$ with $z \in \{1, \dots, N\}$ is the contrastive loss between the query and negative n_z. P and N are the total number of positives and negatives of the tuple, respectively. Note that this loss vector can be interpreted as an ensemble of two other vectors, $\overline{\mathbf{L}}_p$ and $\overline{\mathbf{L}}_n$, which are the loss vectors of the positive and negative pairs.

Now, let us define the statement $\min\{\overline{\mathbf{v}}, k\}$ as the k-th smallest element of the vector $\overline{\mathbf{v}}$ in such a way that $\min\{\overline{\mathbf{v}}, 1\}$ refers to the standard minimum operator. Then, the weak positive loss per tuple can be described by the following expression:

$$\begin{aligned}\mathcal{L}_{wp} = w_o \min\{\overline{\mathbf{L}}_p, 1\} + w_1 \sum_{k=2}^{P_1+1} \min\{\overline{\mathbf{L}}_p, k\} \; + \\ + \, w_2 \sum_{k=P_1+2}^{P_1+P_2+1} \min\{\overline{\mathbf{L}}_p, k\}\end{aligned} \tag{4}$$

with P_1 and P_2 correspondingly being the number of near and border-positives within the tuple. Variables w_0, w_1 and w_2 are constant parameters that weight each of the terms of the equation. Empirically, we set $w_0 = 1$, $w_1 = 0.8$ and $w_2 = 0.05$. In this way, we expect that the model hopefully ends up learning that the first element of the equation (*i.e.* the minimum value of $\overline{\mathbf{L}}_p$) corresponds to the nearest-positive, whereas the other two elements correspond to the near and border-positives, respectively. Note that by following this procedure, only the gradient of the first element is entirely passed backwards through the network during the training process, whilst the gradients of the other elements are considerably weakened. Concretely, if training is going good, the network will only pass a 5% of the gradients of

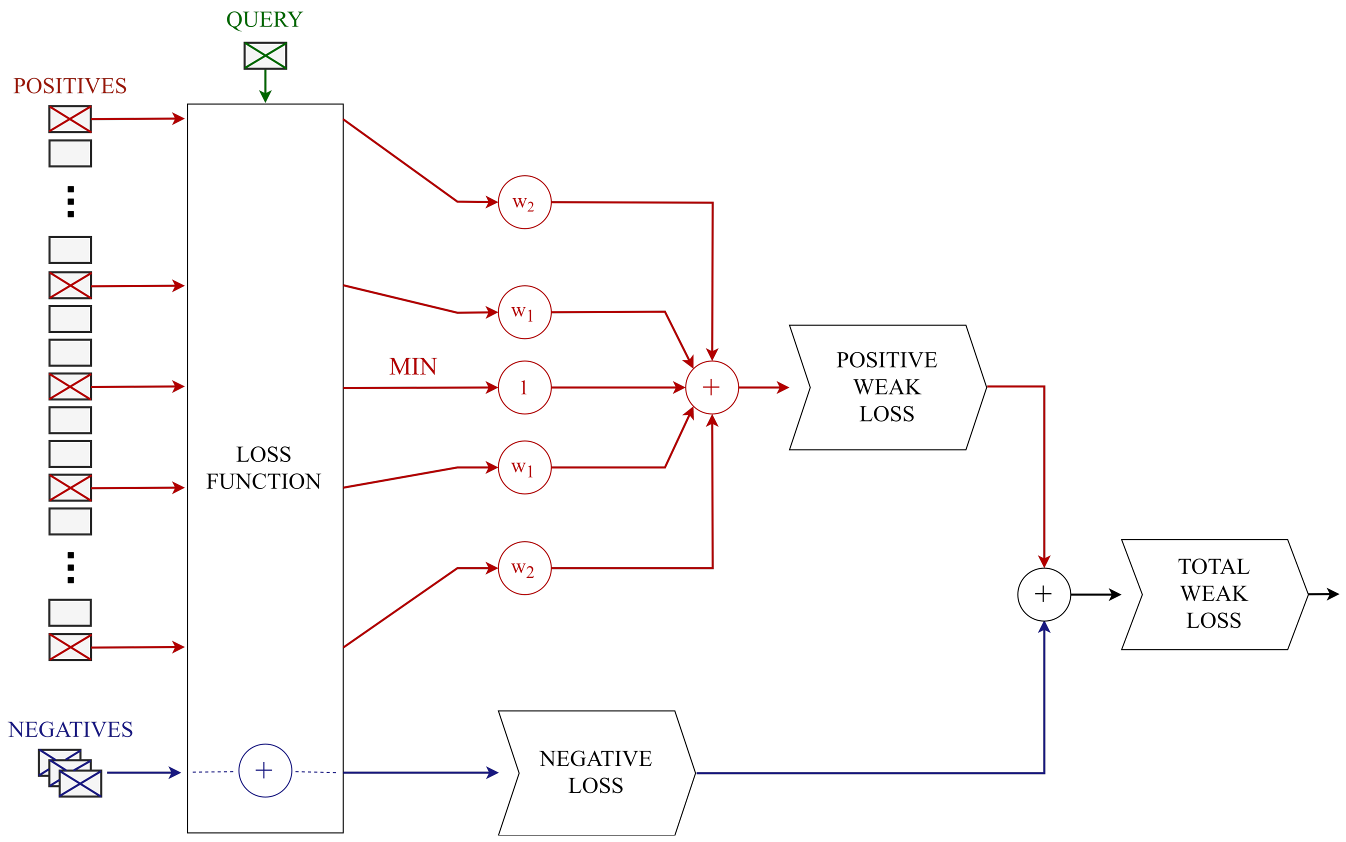

Fig. 6: Weak loss scheme.

border-positives, which are the positives that differ the most with respect to the central query. Therefore, the network is able to gain a better understanding of the different types of positives and thus a better ability to distinguish between consecutive frames. Notice that for this to work, weights have to be initialized in such a way that the network can make relatively good predictions from the beginning of the training. This makes our approach ideal for the finetune of deep neural networks that have been already trained in larger datasets.

Lastly, the final total loss is defined by the sum of the positive weak loss and the individual negative losses, as stated below in *eq.* (5). An scheme of the whole procedure is shown in Figure 6 with a graphical breakdown of each of the losses.

$$\mathcal{L}_{weak} = \mathcal{L}_{wp} + \mathcal{L}_{n_1} + \cdots + \mathcal{L}_{n_N} \tag{5}$$

V. EXPERIMENTS AND RESULTS

In this section, we explain the experimental setup and the metric we employed to evaluate the project. We also present the results in the last subsection.

A. *Experimental setup and compared algorithms*

The experiments carried out during the course of the project were performed over a validation set composed of five random seeded video pieces of 1.5 *km* each, which approximately represent the 25% of the whole dataset (refer to Section III). Both, SIFT and CNN methods were tested over this dataset. The remaining 75% was mainly used to train and finetune the CNN models.

We have chosen the following reference methods:

- **SIFT**. We used SIFT as the baseline for the CNN methods. We evaluated the two feature detectors we described in Section II - DoG and Hessian-Laplace with peak threshold values of 10 and 100, respectively. We stuck with the original SIFT descriptor in both cases as the principal way to describe features.
- **Trained ResNet-101 models**. We tested two different trained ResNet-101 models, one based on ImageNet [43] and the other based on SfM [40], a monument database of popular cities from Flickr. Their goal is to serve as a reference for the finetuning of the network. It's important to remark that the second model was already trained in a matching setup by following a siamese approach whereas the first one was trained in a classification scheme.
- **Finetuned ResNet-101 model**. We finetuned the SfM ResNet-101 model following an standard learning approach. It is the model that will be compared with our weak approach.

Learning setup. For training, we employed Adam [44] instead of SGD [45] as the optimization algorithm since

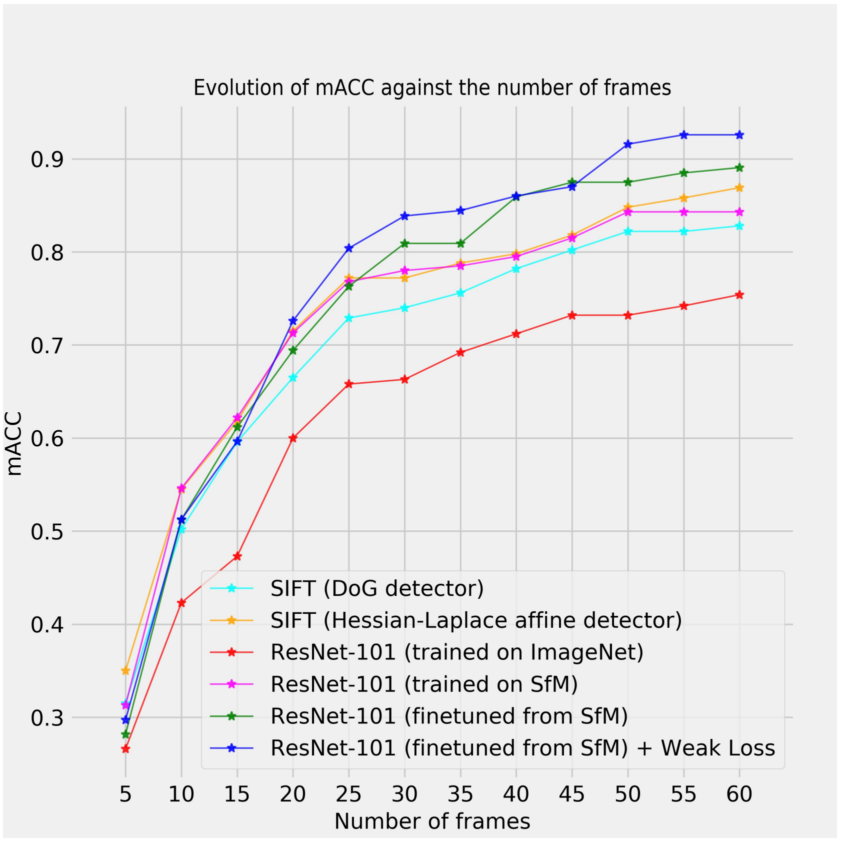

Fig. 7: mACC@F evolution plot for different values of F.

Algorithm	**mACC@60**
SIFT (DoG detector)	82.80%
SIFT (Hessian-Laplace affine detector)	86.91%
ResNet-101 (trained on ImageNet)	75.42%
ResNet-101 (trained on SfM)	84.30%
ResNet-101 (finetuned from SfM)	89.01%
ResNet-101 (finetuned from SfM) + Weak Loss	**92.58%**

TABLE I: mACC@60 results table.

it seems a little more stable with residual networks like ResNet-101. We used an initial learning rate equal to $lr_0 = 5 \cdot 10^{-8}$ with exponential decay $lr = lr_0 \exp(-0.01ep)$ over each epoch ep, momentum 0.9 and weight decay of 10^{-4}. In addition, we used a constant margin of 0.85 for the contrastive loss.

Due to the massive resolution of the road images and in order to not exceed out GPU's memory (in our case, 11 GB Nvidia 1080 Ti), images had to be resized to 1536×160 px during training (*i.e.* images were downsampled in a 2.5 factor). No letterbox or any additional fill resizing was needed due to the fully convolutional nature of the network that manages non-square images without having to alter inner filter dimensions. Additionally, we set the batch size to 1 as this is the maximum size we could elect that handles a moderate number of positive and negative pairs as well as an acceptable resolution. Choosing a larger number of batches would have returned in getting a less noisy estimation of the gradients. Nonetheless, we prioritized choosing a high resolution rather than a large number of batches since resolution limits much more the performance due to the horizontal nature of our video sequences. Note that this is only done during training in order to avoid memory problems; during inference the resolution of the images remains intact. Lastly, training is performed for 30 epochs and 1500 queries are analyzed in each of these epochs (we randomly shuffled the queries). As each query is associated to a tuple of positives and negatives (5 positive and 6 negatives), a total of 12 images are processed per batch.

B. *Procedure*

Evaluation was carried out by adopting the accuracy as the main performance metric of the system. There are a lot of metrics we could have used *s.a.* the mAP or the F1-Score. The mAP is a very common metric in many image retrieval and object detection problems and scenarios. It is a really meticulous metric that allows to globally evaluate how good

Algorithm	Inference time
SIFT (DoG detector)	0.0175 *ms*
SIFT (Hessian-Laplace affine detector)	0.0821 *ms*
ResNet-101	0.0082 *ms*

TABLE II: Unit inference times for SIFT and ResNet-101.

the system does at retrieving a set of queries by taking into account the whole ranking of comparisons. However, for our specific problem, it is not so important to retrieve the whole ranking of scores but just the first entry of it, as that is the frame that we will use to draw conclusions about the occlusions. Thereby, given a video in which an occlusion was detected, instead of comparing that sequence with the whole database (with all the videos from different days), it results more appropriate to implement the inverse comparison in an iterative way; pick up the interest frame from a video of a different day, treat it as if it were the query, compare it with the frames of the video where an occlusion was detected and finally take the most similar image (the first of the ranking). If this image is the occluded one, matching can be settled and we can corroborate if the landmark was missing or present. Otherwise, a video from another day may be taken in order to repeat the process and re-assess the possible occlusion. Since we only take into consideration the first position of the ranking, our evaluation metric resembles more to the accuracy score than to the mAP. Concretely, we make use of the mACC (Mean Accuracy), which can be written according to the following expression:

$$\mathrm{mACC} = \frac{1}{N_{\mathrm{vid}}} \sum_{\mathrm{vid}=1}^{N_{\mathrm{vid}}} \frac{1}{N_{\mathrm{day}}} \sum_{\mathrm{day}=1}^{N_{\mathrm{day}}} \mathrm{ACC}_{\mathrm{vid}}(\mathrm{day}) \quad (6)$$

where $\mathrm{ACC}_{\mathrm{vid}}(\mathrm{day})$ refers to the accuracy score obtained for video vid between a certain day and the day where the occlusions occurred, N_{day} is the number of evaluated days ($N_{\mathrm{day}} = 5$) and N_{vid} is the total number of validation video pieces ($N_{\mathrm{vid}} = 5$). It's important to remark that we do not evaluate on the whole dataset but just on the validation videos in an individual way, *i.e.* a different accuracy value is obtained for each video and then all the accuracies are merged into a final average score. We split the evaluation into different videos because we consider that evaluating them jointly is a really unrealistic scenario as a GPS location error of more than 1.5 *km* is rather improbable.

The accuracy score itself can be defined as the ratio between the number of successes and the total number of query comparisons within the video. We consider as successes or right guesses those cases where the first image of the ranking belongs to the same sequence (see Figure 3) as the analyzed query. Additionally, we studied how much the size of the sequences affects the performance of the algorithms so that sequences were re-defined by a different number of frames, from 5 to 60, taking as reference the central frame. Hence, we define the mACC@F as the mACC evaluated on sequences of a size equal to F frames. For instance, if we are evaluating the mACC@5 score, we only take as ground-truth the central frame of the sequence plus the two contiguous frames at the right and left sides. For those cases where F is larger than the original size of the sequence, we just take the entire sequence as the ground-truth at issue.

C. Results

In this subsection we show and analyze the results of the experiments discussed previously. Figure 7 shows the mACC@F results attained with each of the six proposed methods. As expected, all the algorithms obtain higher accuracy scores as the number of frames per sequence increases. All curves present low scores at $F = 5$ frames, as this is the margin where the inherent GPS error lies. Note that the accuracy rapidly grows until reaching configurations of about $F = 30$ frames, where the performance keeps increasing but in a steadier way. This is the point where most algorithms start to correctly match the semantic information of the videos. Table I shows the maximum mACC values of each of the algorithms, which are attained for $F = 60$ frames. This table is essential, as it demonstrates how good the algorithms perform when the original sequences are used as ground-truth.

From the plot, we can observe that ResNet-101 trained on ImageNet (red curve) is the model that works the worst by a large margin, due to the fact that its weights come from a too generic semantic problem. The SIFT method that uses the DoG detector (cyan curve) is in the second to last position. ResNet-101 trained on SfM (magenta curve) is placed next in the ranking. Its performance is very similar to SIFT with Hessian-Laplace detector (yellow curve), although this one wins by a little when F is large. As expected, SfM dataset relates better than ImageNet to our image matching problem. Lastly, we can see that best results are attained when finetuning the network with the SfM initialization. Finetuning allows the network to learn specific features of our problem (*e.g.* roadway boundaries or different tree patches) and hence improve the performance. Concretely, at mAP@60, our weak learning approach (blue curve) works slightly better, obtaining a mACC of 92.58%. This implies an improvement of $+3.57\%$ with respect to the finetuning SfM baseline (green curve).

From inspection, we realized that most of the errors of the system come from the algorithms failing at detecting occlusions when comparing the images with the rainy day of the database. Thus, training the models with a larger and more diverse dataset should help to alleviate these problems so that magnitudes closer to 100% can be reached.

Regarding the computational efficiency of the algorithms, CNNs outperform both SIFT methods. Unit inference times can be found in Table II. These represent how long each algorithm took to extract the features of two frames and accordingly match them, within a comparison of a video piece of 100 images. SIFT made use of a GPU parallelization speed-up in order to compete with CNN models. Still, Hessian-Laplace SIFT is an order of magnitude below the neural network approach in terms of computational efficiency.

VI. CONCLUSIONS

In this paper, we presented a system that is capable of automatically detecting occlusions of traffic milestones by analyzing and comparing road images from videos that were recorded in different days. We tackled the problem as a feature matching problem where the most similar image for a given query has to be retrieved so that correspondences between images with and without occlusions can be realized. Traditional computer vision methods such as SIFT were compared with deep learning CNN models in order to establish a performance reference baseline. Evaluation was addressed by using the mACC@F metric which depends on the number of chosen frames F of the sequences. Results have shown that the SfM finetuned ResNet-101 architecture performs the best in both, accuracy and computational efficiency. We managed to outperform every proposed baseline with a novel weak learning approach that fits better our sequential problem. It obtained a mACC@60 score of 92.58%, which we consider that is a sufficiently high score as to apply the system in a real scenario.

Regarding the future lines of research, it would be very useful to train the system with a bigger dataset, specially one with a wider variety of meteorological conditions. Learning an illumination transformation that could relate to different meteorological scenarios would be incredibly convenient too. In addition, new training configurations and modifications of well-known loss functions *s.a.* the triplet loss [46] could be tested and compared with our weak approach in order to further assess the performance of our system.

APPENDIX I
MASTER'S THESIS DEVELOPMENT TIME

In this Appendix we specify the time spent on each of the steps taken in the development of the project, according to the normative of the UC3M Master studies.

1) The project started in November 2018. We dedicated the first months of the project - from November 2018 to February 2019 - to record the roadways in order to create an initial database of videos from different days.
2) From February to June, we built the object detector system capable of finding traffic milestones.
3) From June to July, we processed the videos in order to create an image matching experimental dataset. During this lapse of time, we carried out the detection of the cars, the GPS alignment and the creation of the sequences that compose the dataset.
4) August was dedicated mainly to build the global matching system.
5) Finally, in September we were able to improve the results already obtained by finetuning the CNNs with a novel weak learning approach.

REFERENCES

[1] "10th Road Safety Performance Index Report," ETSC, Tech. Rep., Jun. 2016.

[2] *Infrastructure Maintenance Data*, https://data.oecd.org/transport/infrastructure-maintenance.htm, 2017.

[3] D. Regan, *Human perception of objects*. Sunderland, MA: Sinauer, 2000.

[4] S. Papert, "The summer vision project," *MIT AI Memos*, 1966.

[5] G. Guy and G. Medioni, "Inferring global perceptual contours from local features," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 1993, pp. 786–787.

[6] J. Canny, "A computational approach to edge detection," in *Readings in computer vision*, Elsevier, 1987, pp. 184–203.

[7] C. G. Harris, M. Stephens, *et al.*, "A combined corner and edge detector.," in *Alvey vision conference*, Citeseer, vol. 15, 1988, pp. 10–5244.

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[9] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–8.

[10] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Ieee, vol. 2, 2006, pp. 2161–2168.

[11] A. P. Witkin, "Scale-space filtering," in *Readings in Computer Vision*, Elsevier, 1987, pp. 329–332.

[12] T. Smith Jr, W. Marks, G. Lange, W. Sheriff Jr, and E. Neale, "Edge detection in images using marr-hildreth filtering techniques," *Journal of neuroscience methods*, vol. 26, no. 1, pp. 75–81, 1988.

[13] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on communications*, vol. 31, no. 4, pp. 532–540, 1983.

[14] L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Packing and padding: Coupled multi-index for accurate image retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1939–1946.

[15] H. Jegou, M. Douze, and C. Schmid, “Hamming embedding and weak geometric consistency for large scale image search,” in *European conference on computer vision*, Springer, 2008, pp. 304–317.

[16] S. Lee, G.-G. Lee, E. S. Jang, and W.-Y. Kim, “Fast affine transform for real-time machine vision applications,” in *International Conference on Intelligent Computing*, Springer, 2006, pp. 1180–1190.

[17] K. Mikolajczyk and C. Schmid, “An affine invariant interest point detector,” in *European conference on computer vision*, Springer, 2002, pp. 128–142.

[18] K. Mikolajczyk *et al.*, “A comparison of affine region detectors,” *International journal of computer vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[19] K. Mikolajczyk and C. Schmid, “Indexing based on scale invariant interest points,” 2001.

[20] T. Lindeberg and J. Gårding, “Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure,” *Image and vision computing*, vol. 15, no. 6, pp. 415–434, 1997.

[21] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” 2005.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[23] L. Zheng, Y. Yang, and Q. Tian, “Sift meets cnn: A decade survey of instance retrieval,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1224–1244, 2017.

[24] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, “Deep image retrieval: Learning global representations for image search,” in *European conference on computer vision*, Springer, 2016, pp. 241–257.

[25] O. Morère *et al.*, “Group invariant deep representations for image instance retrieval,” in *2017 AAAI Spring Symposium Series*, 2017.

[26] V. D. Sachdeva *et al.*, “Performance evaluation of sift and convolutional neural network for image retrieval,” *Performance Evaluation*, vol. 8, no. 12, 2017.

[27] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[30] N. R. Chopde and M. Nichat, “Landmark based shortest path detection by using a* and haversine formula,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 298–302, 2013.

[31] F. He, T. Liu, and D. Tao, “Why resnet works? residuals generalize,” *arXiv preprint arXiv:1904.01367*, 2019.

[32] S. Targ, D. Almeida, and K. Lyman, “Resnet in resnet: Generalizing residual architectures,” *arXiv preprint arXiv:1603.08029*, 2016.

[33] Y. Bengio, P. Simard, P. Frasconi, *et al.*, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[34] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, *Gradient flow in recurrent nets: The difficulty of learning long-term dependencies*, 2001.

[35] G. Papandreou, I. Kokkinos, and P.-A. Savalle, “Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 390–399.

[36] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.

[37] E. Kauderer-Abrams, “Quantifying translation-invariance in convolutional neural networks,” *arXiv preprint arXiv:1801.01450*, 2017.

[38] S. M. Tooth and J. Dobelman, “A new look at generalized means,” *Applied Mathematics*, vol. 07, pp. 468–472, Jan. 2016. DOI: `10.4236/am.2016.76042`.

[39] P. Dollár, Z. Tu, P. Perona, and S. Belongie, “Integral channel features,” 2009.

[40] F. Radenović, G. Tolias, and O. Chum, “Fine-tuning cnn image retrieval with no human annotation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[41] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[42] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer, “Fracking deep convolutional image descriptors,” *arXiv preprint arXiv:1412.6537*, 2014.

[43] J. Deng *et al.*, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.

[44] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[45] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

[46] B. Kumar, G. Carneiro, I. Reid, *et al.*, “Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions,” in *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition, 2016, pp. 5385–5394.

ABBREVIATIONS

Adam	Adaptive Moment Estimation.
CBIR	Content-based Image Retrieval.
CNN	Convolutional Neural Network.
DoG	Difference of Gaussians.
ETSC	European Transport Safety Council.
GeM	Generalized Mean.
GPM	Grupo de Procesado Multimedia - Multimedia Processing Group.
GPS	Global Positioning System.
GPU	Graphics Processing Unit.
HOG	Histogram of Oriented Gradients.
ILSVRC	ImageNet Large Scale Visual Recognition Challenge.
LoG	Laplacian of Gaussian.
mACC	Mean Accuracy.
mAP	Mean Average Precision.
PIN	Road Safety Performance Index.
ReLU	Rectified Linear Unit.
RPN	Region Proposal Network.
SfM	Structure-from-Motion.
SGD	Stochastic Gradient Descent.
SIFT	Scale-Invariant Feature Transform.
UC3M	Universidad Carlos III de Madrid.
VOC	Visual Object Classes.
YOLO	You Only Look Once.