View-Invariant Intersection Recognition from Videos using Deep Network Ensembles

Anonymous WACV submission

Paper ID 432

Abstract

This paper strives to answer the following question: Is it possible to recognize an intersection when seen from different road segments that constitute the intersection? An intersection or a junction typically is a meeting point of three or four road segments. Its recognition from a road segment that is transverse to or 180 degrees apart from its previous sighting is an extremely challenging and yet a very relevant problem to be addressed from the point of view of both autonomous driving as well as loop detection. This paper formulates this as a problem of video recognition and proposes a novel LSTM based Siamese style deep network for video recognition. For what is indeed a challenging problem and the limited annotated dataset available we show competitive results of recognizing intersections when approached from diverse viewpoints or road segments. Specifically, we tabulate effective recognition accuracy even as the approaches to the intersection being compared are disparate both in terms of viewpoints and weather/illumination conditions. We show competitive results on both synthetic yet highly realistic data mined from the gaming platform GTA as well as on real world data made available through Mapillary.

1. INTRODUCTION

Recognizing an intersection from a different approach sequence or a sequence of viewpoints different from one seen before can be pivotal in various applications that include autonomous driving, driver assistance systems, loop detection for SLAM including multi-robot and multi-session SLAM frameworks. An immediate benefit could be that data can be exchanged between robots only around areas of intersection than across entire runs to detect and close loops, an idea that was put forth in an indoor robotic context earlier [20]. Retrieving previously seen intersections can also be advantageous from the point of view large-scale outdoor topological mapping frameworks.

Intersection recognition from disparate video streams or image sequences is an extremely challenging problem that stems due to large variation in viewpoint, weather and appearance. Complexity also emanates from varying levels of traffic and chaos at a junction[1] between two sequences, as well as due to changing levels of occlusion, illumination between two video sequences of a specific intersection. Lack of annotated datasets on intersections also poses a challenge.

Within the robotics community, loop detection methods have used diverse image recognition and retrieval techniques [16, 15, 6] that have not attempted to detect loops from very diverse viewpoints while they perform admirably in the presence of weather changes [23] or under the duress of varying traffic [17]. In the vision community, CNN [31] features have been used to efficiently compare scenes or structures which are viewed from varying depths, which give a zoom-in vis-a-vis zoom-out effect. Whereas in this paper, we argue that recognizing intersections has many potential benefits and therefore needs an attention and treatment of its own.

We propose a novel stacked deep network ensemble architecture that combines state-of-the-art CNN, Bidirectional LSTM and Siamese style distance function learning for the task of view-invariant intersection recognition in videos. While the CNN component of the ensemble primarily deals with image-level feature representation, the bidirectional LSTM is the key recurrent network component that enables learning of temporal evolution of visual features in videos followed by the Siamese network-based distance metric learning for comparing two input video streams. Our proposed network is conceptualized in Figure 1.

We contribute in the following ways:

1. Firstly we propose a novel and pertinent problem of recognizing intersections when approached from highly disparate viewpoints

[1] In this work, we use the words junctions and intersections interchangeably.

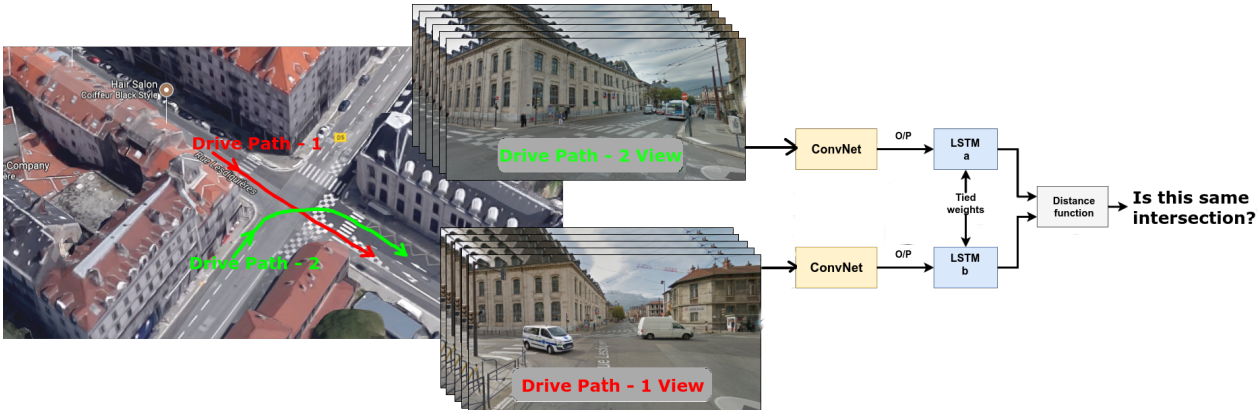

Figure 1. Trajectories shown with red and green are separate drive sessions taken at the same intersection. Corresponding videos are given to our proposed network to determine whether they are belonging to same intersection or not.

2. We propose an original deep network ensemble. The proposed architecture can handle videos of varying length and compare videos capturing reverse trajectories. To the best of our knowledge, this kind of Siamese network with recurrent LSTM component has been largely unexplored in the video domain. Furthermore, we use the hidden state of LSTM cells, instead of the traditionally used output state, for video representation, which has largely been unexplored too.

3. We showcase the efficacy of the proposed architecture through competitive results on challenging sequences. We use synthetic yet highly realistic data from the GTA [2] gaming platform and actual real world data of Mapillary [4] for this purpose.

To the best of our knowledge, this is the first attempt in this direction using the ensemble of state-of-the-art CNN, bidirectional LSTM and Siamese network architecture for recognizing intersection in input video pairs. Highly unpredictable nature of road intersections makes them vulnerable to accidents. Nevertheless, proposing robust solutions for autonomous vehicles aware of the intersections and the being able to recognize them based on a previously explored pathways is much needed.

2. Literature Review

In existing literature around robotic perception, there are limited efforts towards intersection detection and recognition. Some of these use sensors other than cameras such as laser and virtual cylindrical scanners [35, 22].

Recently, a framework for detecting intersections from videos was proposed in [11]. They used LSTM based architecture to observe the relative change in outdoor features to detect intersection. Nevertheless, their work did not focus on intersections recognition task. On the other hand, visual loop closure detection techniques in the literature [7, 16, 15] mainly focused on image level comparison of scenes. These techniques try to find re-occurring scene during the driving session based on fast KD tree based comparison of image features. Although these methods achieve admirable accuracy for loop detection they cannot be improvised for view-invariant recognition over videos.

In recent years, gaming environments have been created and/or used for generating and annotating datasets. [27] used Grand Theft Auto V (GTA) gaming-engine to create large-scale pixel-accurate ground truth data for training semantic segmentation systems. SYNTHIA [28] is a virtual world environment which was created to automatically generate realistic synthetic images with pixel-level annotations for the task of semantic segmentation. [27] and [28] show the added improvement in the real-world setting by using synthetic-datasets for training deep-network models. CARLA [13] is another open-source simulator developed to help in development, training and validation of Autonomous Driving. Visually, GTA environment is more realistic than SYNTHIA and CARLA environments.

Real-world datasets such as [25, 1] are designed with the idea of visual place recognition using images. These datasets are limited in the pathways covered and the number of intersections. Another real world street-level imagery platform [4] consist of images and image-sequences with variability in the pathways and intersections and thus enables sequence learning for intersection recognition. Concurrent works have used this platform to create Mapillary Vistas Dataset [26] which is a semantic segmentation dataset consisting of 25k images. We have used Mapillary dataset to showcase our result on real world scenarios.

Specifically there has been no prior effort towards recognizing intersections when approached from different road segments that constitute the intersection.

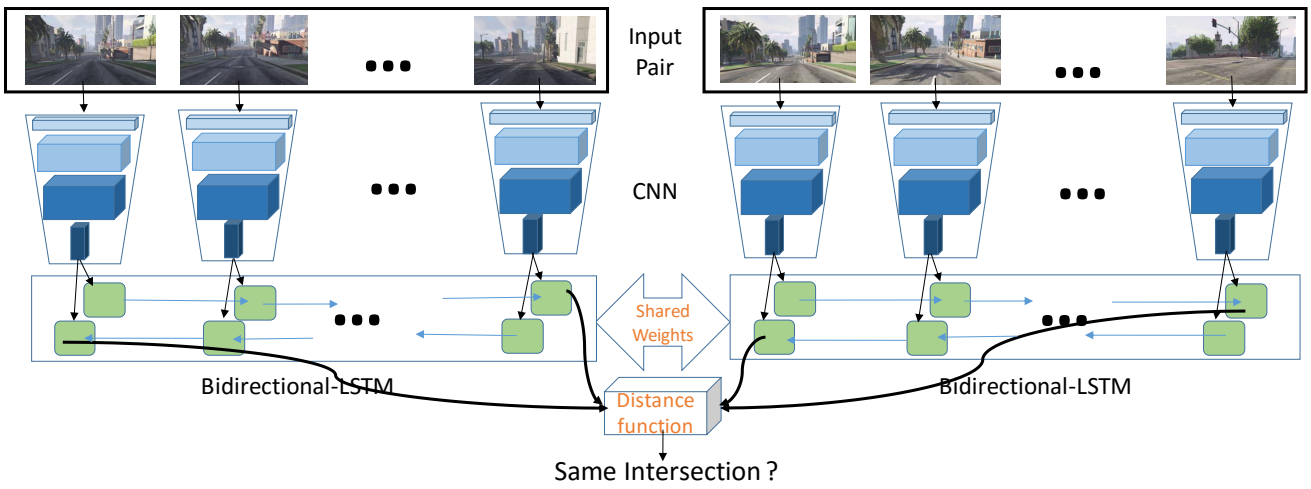

Figure 2. Proposed model is shown with video pairs as input and binary-classification as the output. Features from pretrained CNN network are fed into bidirectional LSTM with shared-weights as shown with hidden units in green (unfolded in time).

3. Method

In order to achieve view-invariant intersection recognition from video sequences, our overall approach follows the recent successes of deep network models in learning useful representations of visual data [9]. Our overall strategy comprises of three specific sub-objectives: (i) capture representations of individual video frames that can help identify the intersection in a view-invariant manner; (ii) leverage the temporal correlations between video frames to better capture the unique identity of an intersection across different views; and (iii) use an appropriate distance metric to compare the thus learned spatiotemporal representations of intersection video sequences. We use a stacked deep network ensemble architecture to realize the above strategy.

Our stacked ensemble consists of a state-of-the-art Convolutional Neural Network (CNN) to learn video frame representations, a Bidirectional Long Short-Term Memory (LSTM) Network (a variant of Recurrent Neural Networks, that addresses the vanishing gradient problem) network that captures temporal correlations across video frames, and a Siamese Network that learns a suitable distance metric that can uniquely identify an intersection from these video sequences across views. Figure 2 summarizes the proposed ensemble strategy.

3.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) based models have recently enjoyed tremendous success in a variety of tasks including image classification [21], image-segmentation [8], activity-recognition [33], etc. CNN is a biologically-inspired model which sequentially stacks a series of convolution operations on an image followed by non-linear activations and pooling operations, and are capable of capturing low-level and high-level features from the image. It has been shown that codes learned by the CNN can be used as off-the-shelf features for a variety of tasks [30], [32].

From a plethora of CNN pretrained models, we choose the model most relevant to our tasks which provide invariance in lighting and pose. In particular, we employ AmosNet [12] as our CNN architecture. AmosNet is a variation of CaffeNet [21] trained for Visual Places Recognition under varying weather conditions. We plug in AmosNet with pretrained weights and use the *pool6* outputs as input features for the LSTM.

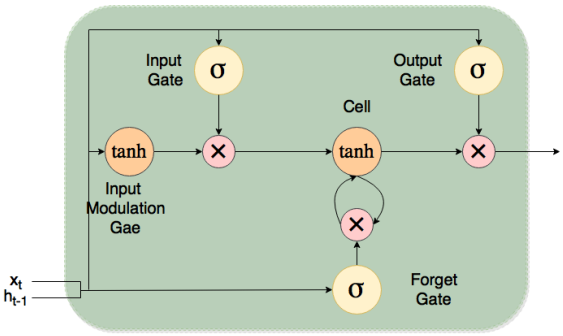

Figure 3. LSTM structure. Expansion of green block shown in Figure 2 (reproduced from [14])

3.2. Long Short-Term Memory Network

Recurrent Neural Networks (RNNs) are neural networks adapted for time-varying sequential data by incorporating a feedback-loop in the architecture (Figure 3). In theory, RNNs should be able to model temporal correlations in long-term sequences; however, they often fail in practice

due to the exploding/vanishing gradient problem [10].

Long Short-term Memory (LSTM), introduced by Hochreiter and Schmidhuber [18], is a variant of RNN with the ability to capture long-term dependencies [19]. Similar to RNN, it has a hidden representation which is updated at each time step, but it also consists of a memory state ($\mathbf{c}_t$), which holds the relevant information at each time step. The information entering and exiting out of cell states is controlled by gating mechanisms (sigmoid, and $\tanh$ functions). An input gate $\mathbf{i}_t$ decides what part of the information entering the LSTM through $\hat{g}_t$, should be held in the current cell state, $\mathbf{c}_t$. The forget gate $\mathbf{f}_t$ decides what to forget from the previous cell state, $\mathbf{c}_{t-1}$. The output gate $\mathbf{o}_t$ determines how much of c_t should be visible as output of the LSTM at time t. This system is captured by the set of equations below:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\hat{\mathbf{g}}_t &= \tanh(\mathbf{W}_g\mathbf{x}_t + \mathbf{U}_g\mathbf{h}_{t-1} + \mathbf{b}_g) \\
\mathbf{c}_t &= \mathbf{i}_t \odot \hat{\mathbf{g}}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh \mathbf{c}_t
\end{aligned}$$

Here $\sigma(\mathbf{x})$ is a sigmoid function which operates on each element of the vector $\mathbf{x}$ separately and outputs a value between 0 and 1, and $\odot$ represents the Hadamard product. LSTMs are capable of handling variable-length inputs and unfolding each input dynamically.

Bidirectional RNNs [29] incorporate future and past context by running the inverse of the input through a separate RNN. A similar extension for LSTMs is called Bidirectional LSTMs which have shown good performance in many applications that involve sequential data such as text [24]. The proposed model employed Bidirectional LSTMs to capture the temporal correlations between video frames. This enables our model to identify the intersection uniquely whether a vehicle approaches it in the forward or opposite direction.

Instead of using averaged/fused output of unfolded LSTM units or output from last unit we use the hidden-representation from the last unfolded time step as a feature vector for the videos as shown in Figure 2. This is because, we have empirically observed that the final results while using LSTM's hidden state outperform results with traditionally used regular output state.

3.3. Siamese Network

The third component of our deep network ensemble is a Siamese network, which is used to learn data-driven distance metrics. A Siamese Network [34] consists of two independent networks with same architecture and shared weights. Both these networks are merged to learn a distance metric, d, between the two inputs provided to the respective networks.

During training, Siamese networks work on triplets ($\mathbf{x}_1$, $\mathbf{x}_2$, y) where y is the ground truth similarity between $\mathbf{x}_1$ and $\mathbf{x}_2$, i.e., $y = 1$ if the videos denote the same intersection, else $y = 0$. The networks' weights are then updated by minimizing the loss function described below.

Contrastive Loss: The total loss function over a dataset $X = \{(\mathbf{x}_1^i, \mathbf{x}_2^i, y^i), \forall i = 1, \cdots, n\}$ is given by:

$$L_{\mathbf{W}}(X) = \sum_{i=1}^{n} L_{\mathbf{W}}^i((\phi(\mathbf{x}_1^i), \phi(\mathbf{x}_2^i)), y^i) \tag{1}$$

where $\mathbf{W}$ are the weights/parameters of the network, $\phi(\mathbf{x})$ denotes the output of the last layer of the shared network architecture for each individual input $\mathbf{x}$, and $L_{\mathbf{W}}^i$ is

$$L_{\mathbf{W}}^i = y^i L_{pos}(\phi(\mathbf{x}_1^i), \phi(\mathbf{x}_2^i)) + (1 - y^i) L_{neg}(\phi(\mathbf{x}_1^i), \phi(\mathbf{x}_2^i)) \tag{2}$$

where

$$L_{pos}(\phi(\mathbf{x}_1^i), \phi(\mathbf{x}_2^i)) = d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2))^2 \tag{3}$$

$$L_{neg}(\phi(\mathbf{x}_1^i), \phi(\mathbf{x}_2^i)) = \max(1 - d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)), 0)^2 \tag{4}$$

At test time, the learned distance function is used to predict the similarity in the videos using the expression in Eq. 5.

$$\hat{y} = \begin{cases} 0 : d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) > \theta \\ 1 : d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) \leq \theta \end{cases} \tag{5}$$

We set the value of θ to be 0.5 in our experiments. Since the number of possible negative pairs can be very high as compared to positive pairs, we scale the loss function for each positive by a constant factor (> 1), given by the ratio of negative pairs to positive pairs in the dataset.

3.4. Training and Network Parameters

Although our model can be trained in an end to end manner we train it in a greedy way based on empirical observation. We use pretrained weights from previous state-of-the-art CNN models which provide a meaningful weight-initialization to our model and reduces training time. We use the contrastive loss described in Eq. 1 to train the Bidirectional LSTM, as shown in Figure 2. The proposed model is trained using the ADAM optimizer which is a variant of Stochastic Gradient Descent (SGD).

In our initial experiments, we varied the number of Bidirectional LSTM-layers (upto 3), but the performance gain was negligible. Similarly, we experimented with different values (5, 10, 50, 80, 150, 250) for hidden-unit dimensions in the LSTM cell. Empirical results found 80 to be the best performing after which the performance saturates. Thus, we fix the number of Bidirectional layer to one and hidden layer dimension to be 80 in all further experiments.

Figure 4. Data Visualization: Random snapshots from GTA environment [2] (Row 1,2) and Mapillary [4] (Row 3,4). Images show different weather and day/night conditions as well as various outdoor scenes for urban scenarios. The game-environment snapshots look visibly similar to that of real world images and have significantly higher variations in weather and lighting. (Best seen in color)

A pair of videos x_1, x_2 are passed as a set of contiguous frames to the model. Each frame in the video is an RGB image with a fixed size ($227 \times 227 \times 3$). For each frame, the CNN feature ($6 \times 6 \times 256$ which is resized to 1×9216) is computed. These CNN features from each video are fed into two separate Bidirectional LSTM with tied-weights. The hidden unit representation ($\phi(\mathbf{x_1})$, $\phi(\mathbf{x_2})$) for respective input videos from the last cell in the Bidirectional LSTM network is then used as a feature for the video. Further, we compute the distance d between $\phi(x_1)$ and $\phi(x_2)$ using the Siamese network approach, and use this to predict $\hat{y}$ if the videos are of same intersection or not as mentioned before in Section 3.3.

4. Experiments & Results

In this section, we provide detailed description of datasets synthetically generated and real datasets for various experiments. Subsequently we describe the various experimental scenarios which is followed by the explanation of the quantitative and qualitative results.

4.1. Dataset Description

Synthetic data for this task is collected from GTA [2] gaming environment while real world data is mined from publicly available street images/sequences from Mapillary [4]. Figure 4 shows random snapshots captured from GTA environment (Row $1, 2$) and Mapillary (Row $3, 4$). One can infer from the figure that the snapshots from GTA environment are visually realistic as well as similar to real-world data from Mapillary and contain the key challenges associated with real environments (as listed in Section 1). However, the GTA environment offers more complexity in terms of weather and lighting variations as compared to Mapillary.

Each video/sequence consist of a series of frames collected around junction points. Each junction can be traversed in multiple pathways. We refer to the set of all such possible traversals as *trajectories*.

4.1.1 GTA

Here, we collected videos in two different scenarios. Firstly, we choose a set of intersections in the game-environment and then sample them in a *dense* manner, i.e., each possible trajectory in the junction is traversed. Secondly, we traversed a car from one point to another via means of an AI-car driving mode [3]. From such traversal, we break the captured video into small chunks involving intersections and non-intersections. We discard video chunks which have no intersection.

In the first scenario, we selected 27 unique junctions where 9 of them had arbitrary lighting conditions and the rest were in normal daylight condition. In the second scenario, we captured 12 traversals in two arbitrary different lighting variations.

4.1.2 Mapillary

We download images from [4], which is a community-led service for people who collaboratively want to visualize the world with street-level photos. It has more than 200 million images [5]. These are captured in various modes including walking, riding (either a bike or car), panorama and photo-spheres. The data is available under a CC-BY-SA license agreement.

Table 1. Number of video pairs in training, testing and the validation set for the different experimental scenarios

Dataset	Lighting Setting	Trajectory Relation	Training		Validation		Testing	
			Positive	Negative	Positive	Negative	Positive	Negative
GTA	day and night	All combinations	1509	1440	459	612	106	104
		Overlap in view and trajectory	640	1440	168	212	62	104
	day	All combinations	864	1166	300	653	72	82
Mapillary	day	Overlap in view and trajectory	3080	3328	421	428	400	400
		All combinations	6409	6976	350	318	425	420

We use mapillary's API to download images and construct *trajectories*. We first query for all the images in a bounding box defined by the longitudes and latitudes. For every image in the bounding box, the API provides a tuple consisting of image-key, latitude, longitude, orientation and the (video-)sequence it belongs to along with other metadata information. These images can be then downloaded using the image-key. Using two orthogonally overlapping sequences we get a location in the map which is used to identify the intersection. Subsequently, this is used to download a subset of continuous images from all the sequences passing through the intersection. We generate sequences representing multiple trajectories at an intersection from the provided tuples.

From the visual analysis of the extracted data, we found that the dataset lacks different lighting settings (predominantly capturing morning/day hour images). This can be attributed to the limitations of data collection in real world settings.

We mined around $300,000$ images which consisted of around 1700 usable sequences from around 500 junctions. For each junction, we get 3 to 6 sequences which implies that the junctions are not sampled in a *dense* manner.

4.2. Evaluation Metric

We report percentage accuracy as the metric in our experiments. Accuracy is defined as the ratio of the number of correctly classified samples (both positive and negative pairs combined) to the total predictions made.

4.3. Experimental Setup

From the extracted dataset, we first annotate positive and negative video pairs. Positive (videos involving the same intersection) pairs are mined from the above datasets exhaustively, i.e., we select all possible positive pairs. Negative pairs generated from the datasets can be very high as any two trajectories from different junction are treated as a negative pair. Hence we limit the number of negative pairs by randomly fixing a subset.

To make sure that test, train and validation sets are mutually exclusive, we sampled these from different junctions and/or non-overlapping traversals. Specifically, in GTA for the test set, we maintain an exclusive set of 3 intersections which are only sampled from junctions while for training and validation, we use both scenarios as detailed in Section 4.1. Similarly, in Mapillary we maintain the exclusivity in train, validation and test data by extracting sequences from mutually exclusive bounding boxes.

Table 2. Accuracy of our method on different datasets for the different experimental scenarios

Dataset	Lighting setting	Trajectory Relation	Accuracy
GTA	day and night	All combinations	70.95
		Overlap in view and trajectory	78.2
	day	All combinations	76.72
Mapillary	day	Overlap in view and trajectory	81.0
		All combinations	72.1

For a pair of videos involving the same junction there exists a trajectory-relation between them, which can be categorized in terms of overlap in view and/or trajectory. Overlap in trajectory refers to the proximal relation in the two trajectories while overlap in view refers to trajectories, viewing the larger part of the same area. For example trajectories moving in opposite direction can be proximal in distance but can have minimal view overlap.

For the purpose of our experiments we categorize the trajectory-relations into two primary setups: *overlap in view and trajectory* and *All combinations*. The former refers to parallel and overlapping trajectory-relations while the latter refers to all the possible trajectory-relations. Similarly, we categorize the lighting setting into two categories: *day* vs *day and night*. The names are self-explanatory.

We test the robustness and generalizing capability of our model under various trajectory-relation and lighting (day/night) settings. The details of the experiment-wise distribution of samples can be found in Table 1. For GTA, we sampled the video-pairs in three categories based on trajectory-relations and lighting conditions. We discarded the *day* and *Overlap in view and trajectory* setting due to

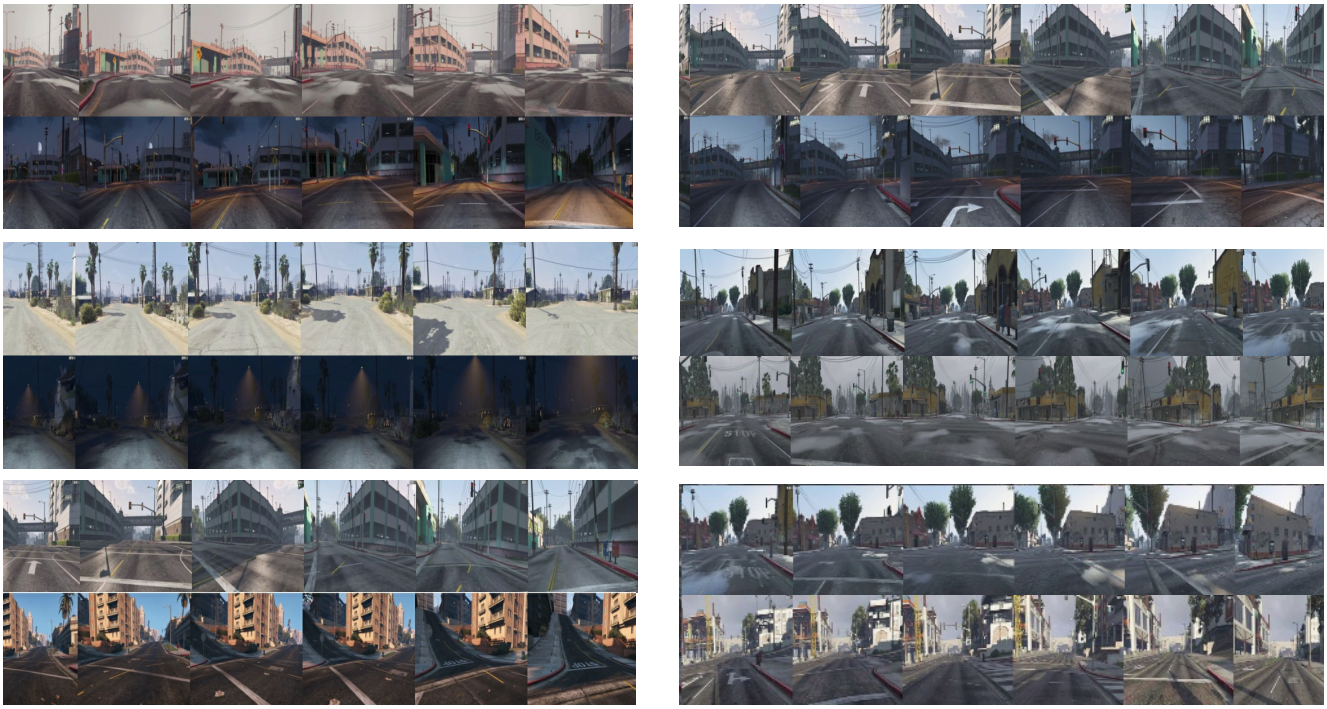

Figure 5. Qualitative results on GTA: Six randomly selected video-pairs (each video is represented as a set of contiguous frames). Row-1,2 show 4 correctly classified positive video-pairs. Row-3 shows 2 correctly classified negative video-pairs. These examples show that our model is able to capture view and lighting variations. Here, we just show 6 similar continuous frames per video, corresponding frames in the two videos need not be temporally aligned (Section 4.6). (Best seen in color)

insufficient training data. For Mapillary, we decided on the two categories as the dataset predominantly consisted of day/morning images.

4.4. Quantitative Results

Table 2 depicts results on GTA and Mapillary dataset under varying scenarios using our proposed model. For GTA, under all possible lighting conditions (*day and night*) the model performs better by 7% when video-pairs have an *overlap in view and trajectory* as compared to *all combinations*. This points to the fact that the generic setting involving all trajectory relations is harder to model as the view variations increase significantly. When the lighting condition is set to *day* only, for all trajectory-relations the model gives an accuracy of 76.72% which is 6.5% greater than *day and night* scenario. This can be attributed to the reduced complexity in lighting variations.

Similarly, for mapillary the model performs better when the trajectory-relations are limited to *overlap in view and trajectory* as compared to *All combinations* by 8.9%.

4.5. Qualitative Results

In this subsection, we fix our model to the most generic scenario (all possible lighting variations and trajectory relations). Figure 5 shows the qualitative result on a subset of 4 positive (Row-1, 2) and 2 negative video pairs (Row-3) from the test-set of GTA using this model. All these video pairs are correctly classified by the model. It is easy to see that our learned model is able to generalize to varying weather, lighting and view conditions. For example: video-pair in Row-2, Column-2 capture the different view of the same structure in rainy weather. Similarly, in Row-1, Column-1 the model is able to predict similar videos in presence of view and lighting variations. The shown examples are in coherence with the challenges in intersection-detection as mentioned in Section 1. In Figure 5 we show 6 contiguous frames from each sequence, where we manually crop the relevant contiguous frames for illustration purposes (see Supplementary material for full videos).

Similarly, Figure 7 show the qualitative result on 3 correctly classified positive video pairs. Each video-pair has a different trajectory-relation. Row-1 video-pair are similar trajectories, Row-2 video-pair are partially-overlapping trajectories and Row-3 video-pair are orthogonal trajectories.

4.6. Discussion

In addition to qualitative and quantitative results it is important to understand the challenges associated with the problem. Here, we explicitly discuss this in the context of positively labeled video-pair from GTA data shown in Figure 6. On careful observation, it becomes evident that only a few corresponding frames in the two videos are similar

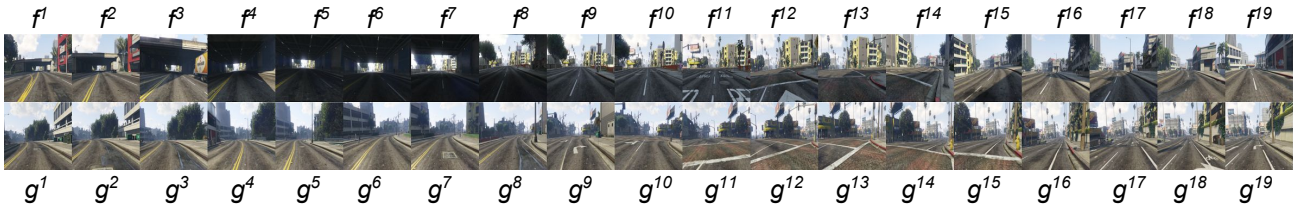

Figure 6. A positively annotated video-pair consisting of 19 frames each. Here, f^i and g^i represent the frames in the temporal order for the two videos respectively. The videos have an overlapping trajectory in opposite directions. It is non trivial to observe that a partial structural overlap exists between a few frames (Observe f^{19} and g^1, f^{10} and g^{11}, f^{11} and g^{12}). (Best seen in color)

Figure 7. Qualitative results on Mapillary: . (Best seen in color)

due to the presence of partial overlapping structure in them. Furthermore these similar frames are not in the same order in the two videos. This reflects one should not interpret the quantum of the results in this context.

To show the limitation of our model, we present a failure case shown in Figure 8, where a negative pair is classified as a positive pair by our network. However visual inspection reveals that in this case, due to presence of large illumination change and less discriminative structure, it is challenging even for human beings to correctly classify the video pair.

Thus, in our view the proposed architecture not stellar but performs significantly impressive for such complex and ill posed problem of view-invariant intersection.

Figure 8. A failure case for the model. Row-1, 2 are contiguous frames from the two videos. The video-pairs belong to different junctions but is classified as same intersection. Images in Row-1 are structurally similar to that of Row-2. (Best seen in color)

5. CONCLUSIONS

This paper proposed a novel stacked deep network ensemble architecture that combines state-of-the-art CNN, bidirectional LSTM and Siamese style distance function learning for the task of view-invariant intersection recognition in videos. The proposed architecture enables recognizing the same intersection across two videos of variable length having large view variations, inverted trajectory, lightning and weather variations. We have collected annotated data (more than 2000 videos) from GTA [2] and Mapillary [4] and have computed results on this data with varying parameter choices and have reported competitive results.

References

[1] Gardens Point Dataset. https://wiki.qut.edu.au/display/cyphy/Day+and+Night+with+Lateral+Pose+Change+Datasets.

[2] Grand Theft Auto V. https://en.wikipedia.org/wiki/Development_of_Grand_Theft_Auto_V.

[3] Grand Theft Auto V Auto-drive Mod. https://www.gta5-mods.com/scripts/vautodrive.

[4] Mapillary. https://www.mapillary.com/app.

[5] Mapillary Wikipedia. https://en.wikipedia.org/wiki/Mapillary.

[6] S. Achar, C. V. Jawahar, and K. M. Krishna. Large scale visual localization in urban environments. In *ICRA*, 2011.

[7] A. Angeli, D. Filliat, S. Doncieux, and J. Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *TRO*, 2008.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[9] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *TPAMI*, 2013.

[10] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Transactions on neural networks*, 1994.

[11] D. Bhatt, D. Sodhi, A. Pal, V. Balasubramanian, and K. M. Krishna. Have I reached the intersection: A deep learning-based approach for intersection detection from monocular cameras. In *IROS*, 2017.

[12] Z. Chen, A. Jacobson, N. Sunderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford. Deep learning features at scale for visual place recognition. *arXiv preprint arXiv:1701.05105*, 2017.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017.

[14] M. Fayyaz, M. H. Saffar, M. Sabokrou, M. Fathy, R. Klette, and F. Huang. STFCN: Spatio-temporal FCN for semantic video segmentation. *arXiv preprint arXiv:1608.05971*, 2016.

[15] D. Gálvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *TRO*, 2012.

[16] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In *ICRA*, 2012.

[17] A. A. Hafez, M. Singh, K. M. Krishna, and C. V. Jawahar. Visual localization in highly crowded urban environments. In *IROS*, 2013.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[19] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[20] A. K. Krishnan and K. M. Krishna. A visual exploration algorithm using semantic cues that constructs image based hybrid maps. In *IROS*, 2010.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[22] Q. Li, L. Chen, Q. Zhu, M. Li, Q. Zhang, and S. S. Ge. Intersection detection and recognition for autonomous urban driving using a virtual cylindrical scanner. *IET Intelligent Transport Systems*, 2013.

[23] C. Linegar, W. Churchill, and P. Newman. Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera. In *ICRA*, 2016.

[24] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

[25] M. J. Milford and G. F. Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *ICRA*, 2012.

[26] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017.

[27] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[28] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[29] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *Transactions on Signal Processing*, 1997.

[30] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf: an astounding baseline for recognition. In *CVPR workshops*, 2014.

[31] N. Sunderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. In *RSS*, 2015.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[33] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[34] W. Yih, K. Toutanova, J. C. Platt, and C. Meek. Learning discriminative projections for text similarity measures. In *CoNLL*, 2011.

[35] Q. Zhu, Q. Mao, L. Chen, M. Li, and Q. Li. Veloregistration based intersection detection for autonomous driving in challenging urban scenarios. In *ITSC*, 2012.

Supplementary Material: View-Invariant Intersection Recognition from Videos using Deep Network Ensembles

Anonymous WACV submission

Paper ID 432

Abstract

This supplement document is organized as follows:

- *Section 1 has additional qualitative results on Mapillary [2] dataset where we discuss performance of the proposed network in the context of success and failuar cases.*
- *We show additional quantitative results for the five different experimental scenarios (as discussed in the main paper) in Section 2.*
- *Additional qualitative results are included in the video-file submitted along with this document. In addition to GTA [1] and Mapillary [2] data, we also show results on publicly available UQ St Lucia Dataset [3]. This data consists of calibrated stereo data from road traversals on a 9.5km circuit around the University of Queensland's St Lucia campus. This dataset is generally used for testing visual SLAM algorithms.*

1. Qualitative Results

In this section, we show additional qualitative results on Mapillary dataset. We present and discuss success as well as failure cases and attempt to understand the labyrinthine challenges associated with problem/data and how proposed model deals with them.

Figure 1 shows a correctly predicted positive video pair[1] from Mapillary where both trajectories pass through same intersection. The pair of trajectories are depicted using two sequences of frames. We zoom into regions from both the trajectories to show corresponding common structures. We also show an *intersection-map* which represent the *top-view* of the trajectories on the intersection. As depicted by the *Intersection-map* in Figure 1, the trajectories are perpendicular and hence minimal view overlap exists between them.

[1] We use the terms sample and video pair interchangeably

Figure 3 shows 5 positive samples from Mapillary that are successfully predicted by the network. Each sample depict a unique trajectory-relation and view-overlap as shown by the *intersection-map* (bottom-most row). We color encode (green, cyan, red, brown) the visually perceptible common structures (using ellipses) in the trajectories. We follow the same encoding while plotting the respective *intersection-map* in the last row. In video pair $1, 5$, we observe that the overlapping part of trajectories are in the opposite direction but our network managed to exploit the common structures seen from different views. In video pair 2, though the partially overlapping trajectories are in the same direction, we can observe that *shadows* pose a challenge that is successfully handled by the network. This indicate the ability of proposed network to generalize on varying lighting conditions. Video pair 3 shows a trivial scenario where both trajectories are overlapping and network can easily predict them to same. On the other hand, video pair 4 shows the tougher scenario with perpendicular trajectories where view overlap is minimal among all the depicted cases and our network successfully predicted this to be the same intersection. Moreover, we also observe occlusion in a few trajectories (video pair 4 trajectory 2, video pair 5 trajectory 2) due to vehicles.

Figure 2 shows some wrong predictions made by the network. Video pair 1 consist of trajectories from different intersections (negative sample) but is predicted as the same intersection. We believe that the model got mislead here due to lack of unique structures in the scene as most of it has trees. Additionally, the variance in lighting conditions (as the network was trained on *day* conditions in Mapillary) can also pose challenge if not trained on other conditions.

In contrast, video pair $2, 3$ are trajectories from the same-intersection but are predicted as different intersection. In video pair $2, 3$ we observe a complete lack of visual features in the overlapping part of trajectories. In case of video pair 2, overlapping structure is completely occluded in the first trajectory due the presence of *truck*. In video pair 3, trajectory 1 has the camera capturing *side-view* instead of the *front-view* while also being partially occluded by the mir-

Figure 1. A correctly classified sample from Mapillary. First row show frames (arranged in the temporal order) from the two trajectories respectively where we highlight the overlapping structural content (green and cyan). Second row zooms in for overlapping structures in the sequences. Rightmost figure in the row shows trajectory of both the videos in an *intersection-map* which acts as a *top-view* of the intersection. We color encode the trajectories and overlapping content in the *intersection-map*. (Best seen in color)

Figure 2. Failure Cases: 3 wrong predictions by the network. Video pair 1 consist of trajectories from different intersections but is predicted as the same intersection. In contrast, video pair $2, 3$ are trajectories from the same-intersection but are predicted as different intersection.

ror in the *car*. Hence, using purely the visual content for all these faiuar cases, even human might easily fail without some extra context (ground truth labels were generated using GPS coordinates).

2. Quantitative Results

We report precision (P), recall (R), F1-score (F1) and true negative rates (TNR) for all the experimental scenarios in Table 1.

For Mapillary, we observe that the model performs better on all metrics when the trajectory-relations are limited to *Overlap in view and trajectory* as compared to *All combinations*. The decrease in precision (by 0.249) and recall (by 0.115) is much higher then the decline in true negative rates (by 0.040). These results are in coherence with the increasing complexity added in trajectory-relations in *All combi-*

Table 1. Precision (P), recall (R), F1 score (F1) and true negative rates (TNR) of our method on different datasets for the different experimental scenarios.

Dataset	Lighting Setting	Trajectory Relation	P	R	F1	TNR
GTA	day and night	All combinations	0.49	0.858	0.623	0.528
		Overlap in view and trajectory	0.588	0.613	0.60	0.83
	day	All combinations	0.69	0.77	0.727	0.736
Mapillary	day	Overlap in view and trajectory	0.759	0.903	0.824	0.716
		All combinations	0.51	0.788	0.619	0.676

nations scenario. Figure 3 shows examples with large view variations and minimal overlap, indicating difficultly level of the *All combinations* setup.

Similarly, for GTA, in *day and night* scenario, we observe that the model performs better on precision (by 0.098) and true negative rates (by 0.30) when the trajectory-relations are limited to *Overlap in view and trajectory* as compared to *All combinations*, whereas it performs poorer on recall (by 0.245).

In the common scenario of *day* and *All combinations*, we notice that the performance on GTA is better than Mapillary on all metrics. This can be explained due to complexity of data annotation in real-world where visually dis-similar videos can be annotated as same based on their GPS coordinates as shown by an example (Video Pair 2) in Figure 2. Such data-complexity is slightly lower in GTA due to the controlled environment settings.

References

[1] Grand Theft Auto V. https://en.wikipedia.org/wiki/Development_of_Grand_Theft_Auto_V. 1

[2] Mapillary. https://www.mapillary.com/app. 1

[3] M. Warren, D. McKinnon, H. He, and B. Upcroft. Unaided stereo vision based pose estimation. In *ACRA*, Brisbane, 2010. 1

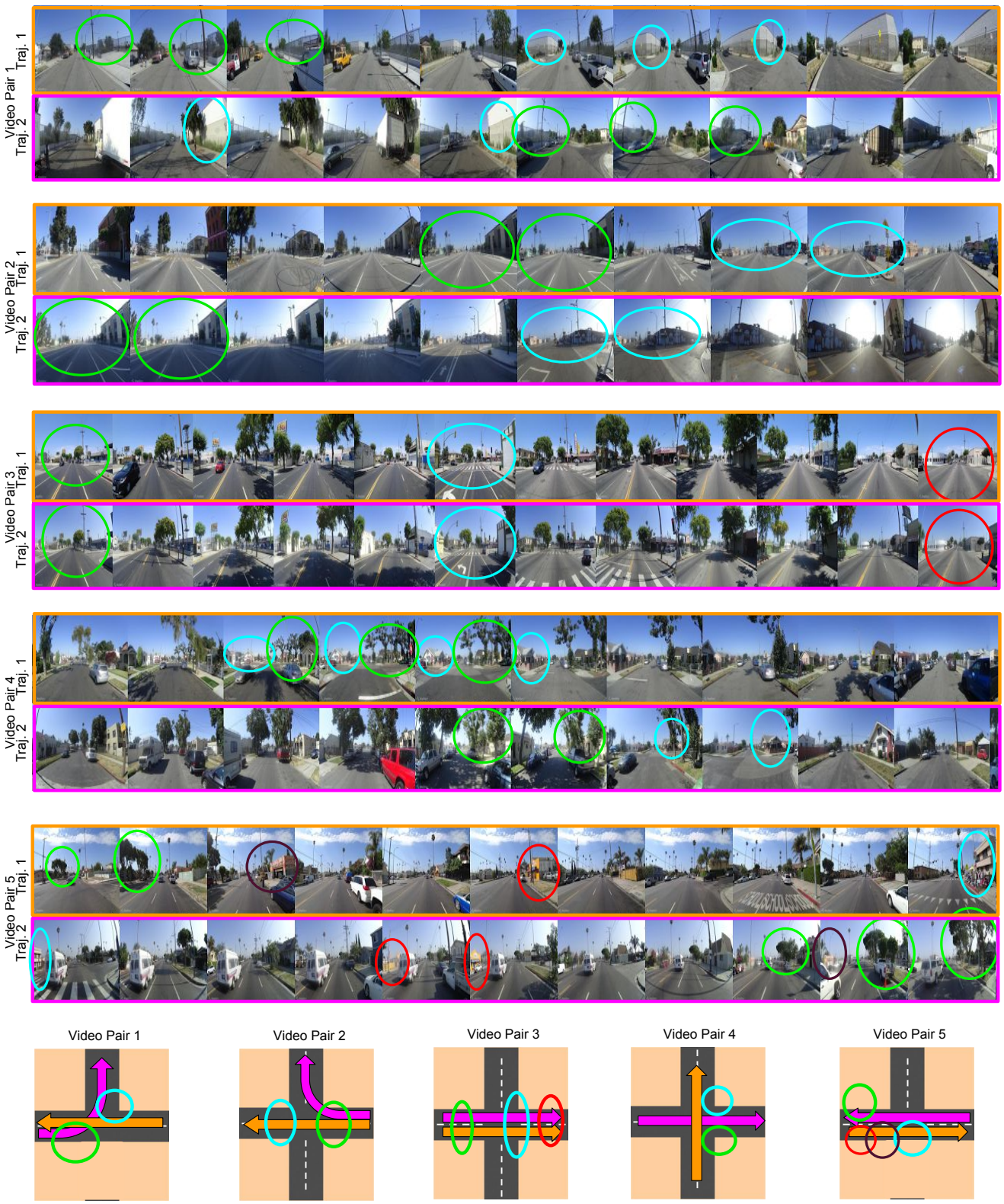

Figure 3. 5 correctly classified samples from Mapillary. Each sample depicts unique trajectory-relation and view-overlap between trajectories as shown in the *intersection-map* (bottom-most row) which represents the *top-view* of the intersection. We color encode the visually perceptible common structures (cyan, green, red, brown) in the Video pairs. Video pair $2, 3$ show examples where trajectory direction and view overlap. Video pair $1, 4, 5$ depict cases when the overlap in trajectories and/or the view is minimal. (Best seen in color)